Latent Sector Faults and Reliability of Disk Arrays

HANNU H. KARI

Kullervonkuja 9B9
FIN-02880, Veikkola, Finland

Thesis for the degree of Doctor of Technology to be presented with due permission for public examination and debate in Auditorium E at Helsinki University of Technology, Espoo, Finland, on the 19th of May, 1997, at 12 o’clock noon.

ESPOO 1997

ABSTRACT

This thesis studies the effects of latent sector faults on reliability, performance, and combined performability of disk arrays. This is done by developing two novel reliability models that include two fault types: disk unit faults and sector faults. The new reliability models study both hot swap and hot spare approaches in repairing disk unit faults. A traditional analytical method and also a novel approximation method are used for analyzing these reliability models. Two RAID (Redundant Array of Inexpensive Disks) configurations are used in the analysis: RAID-1 and RAID-5. Significant drop in reliability and performability is resulted when latent sector faults are not quickly detected. A sector fault may stay a long time undetected if user's disk access pattern is unevenly distributed and the fault resides on a seldom accessed area. To reduce the risk of latent sector faults, this thesis proposes four novel disk scanning algorithms that utilize the idle time of disks to read periodically the entire disk surface in small segments. The main idea of the disk scanning algorithms is the same as in the memory scrubbing algorithm, but this is the first time when this approach is used with the secondary memory. The disk scanning algorithms are analyzed in this thesis and dramatic improvement in reliability and performability is achieved while there is only a minor effect on performance.

Key words: Latent fault detection, latent sector faults, reliability of disk arrays, redundant arrays of inexpensive disks, RAID, performability

> `Well, in our country,' said Alice, still panting a little, `you'd generally get to somewhere else -- if you ran very fast for a long time, as we've been doing.'
>
> `A slow sort of country!' said the Queen. `Now, here, you see, it takes all the running you can do, to keep in the same place. If you want to get somewhere else, you must run at least twice as fast as that!'
>
> Lewis Carroll: Through the Looking-glass

FOREWORD

The original idea for the disk scanning algorithms was invented when I was in Texas A&M University in College Station, Texas, USA. Since I received scholarships from ASLA-Fulbright, Suomen Kulttuurirahasto, and Jenny ja Antti Wihurin säätiö, I got an opportunity to do academic research for two sabbatical years, in 1990-1992. During those days, I was working in Computer Science department in professor Fabrizio Lombardi's research group. After returning back to Finland in September 1992, this work has been continued at Helsinki University of Technology, in Espoo, Finland under supervision of professor Heikki Saikkonen also in Computer Science department.

The initial purpose of this research was to improve the performability of disk arrays in general. It was noticed already at the early phase of the research that the dominant factor of the performability is the latent faults. Hence, this research concentrated on latent fault detection and prevention as well as the analysis of the effects of the latent faults.

I would here express my gratitude to all professors who have helped and encouraged me in my academic career: Fabrizio Lombardi, Kauko Rahko, and Heikki Saikkonen. Thanks to my old colleagues in ICL Personal Systems during the time when I was working in the disk array project: Tapio Hill, Jarmo Hillo, Jouni Isola, Jarkko Kallio, Kari Kamunen, Jorma Manner, Sisko Pihlajamäki, Olli-Pekka Räsänen, and Ville Ylinen. Special thanks to Risto Kari, professor Fabrizio Lombardi, and Lisa Patterson for several helpful suggestions for language and contents of the thesis.[*]

Finally, I would like to express my gratitude to my current employer, Nokia Telecommunications, and its management that have arranged me an opportunity to spend all my available spare time with this research.[†]

Kirkkonummi, April 1997 Hannu H. Kari

[*] All names listed in alphabetical order.

[†] This research has not been supported by Nokia Telecommunications. Neither does this research anyhow indicate the interests of Nokia Telecommunication or its past, present, or future projects, products, or research activities.

Table Of Contents

APPENDICES

List Of Figures

List Of Tables

-

List Of Symbols, Abbreviations, And Terms

LIST OF SYMBOLS

ε_s	error in the approximation of a Markov model
$\varepsilon(D,\lambda,\mu)$	error function in the approximation of a Markov model
λ	disk failure rate
λ_d	disk unit failure rate
λ_{df}	disk unit failure rate after the first disk unit fault
$\lambda_{fail,s}$	disk array failure rate approximation in TMM
$\lambda_{fail,I,A}$	disk array failure rate approximation in EMM1A
$\lambda_{fail,II,A}$	disk array failure rate approximation in EMM2A
λ_s	sector failure rate
λ_{sd}	spare disk failure rate
μ	disk repair rate
μ_d	disk unit repair rate
μ_{dr}	disk repair rate when spare disk is missing
μ_s	sector fault detection and repair rate
$\mu_{s,scan}$	sector fault detection and repair rate by scanning disk requests
$\mu_{s,user}$	sector fault detection and repair rate by user disk requests
μ_{sd}	spare disk repair rate
ρ	utilization (of a disk)
ξ	factor in TMM
ζ	factor in TMM
a_i	instantaneous activity of the scanning algorithm
a_{repair}	activity of the repair process
a_{scan}	activity of the scanning algorithm
$A(t_i)$	adjusted activity of the scanning algorithm at time t_i
$adjust()$	a function to adjust parameters of the scanning algorithm
b_i	percentage of user disk requests falling into specified area c_i
c_i	specified area into which b_i percentage of user disk requests hits
$C_{Double-80/20}(S_a,S)$	coverage of the user access pattern as a function of the number of accessed sectors for Double-80/20 distribution
$C_{Single-80/20}(S_a,S)$	coverage of the user access pattern as a function of the number of accessed sectors for Single-80/20 distribution
$C_{Triple-80/20}(S_a,S)$	coverage of the user access pattern as a function of the number of accessed sectors for Triple-80/20 distribution
$C_{Uniform}(S_a,S)$	coverage of the user access pattern as function of the number of accessed sectors for Uniform distribution
CA	steady state performability of the system
$CA(t)$	total performability of the system at time *t*
$CA_i(t)$	performability of the system at state *i* and time *t*
CCA	cumulative performability of the system over its entire lifetime

CCA_{TMM}	cumulative performability of TMM over its entire lifetime
$CCA_{TMM,A}$	approximation of cumulative performability of TMM over its entire lifetime
CCA_{EMM1}	cumulative performability of EMM1 over its entire lifetime
CP_0	cumulative reliability of TMM at state *0*
CP_{00}	cumulative reliability of EMM1 at state *00*
CP_{01}	cumulative reliability of EMM1 at state *01*
CP_1	cumulative reliability of TMM at state *1*
CP_{10}	cumulative reliability of EMM1 at state *10*
CP_2	cumulative reliability of TMM at state *2*
CP_i	cumulative reliability of the system at state *i*
CP_f	cumulative reliability of EMM1 at state *f*
cr	check region of the scanning algorithm
D	number of data disks in the array
ds	disk size in bytes
$E_{Double\text{-}80/20}$	estimated number of accessed sectors to detect a sector fault with Double-80/20 access pattern (absolute value)
$\overset{\square}{E}_{Double\text{-}80/20}$	estimated number of accessed sectors to detect a sector fault with Double-80/20 access pattern (relative to the number of sectors in the disk)
E_{Scan}	estimated number of accessed sectors to detect a sector fault with the scanning algorithm (absolute value)
$\overset{\square}{E}_{Scan}$	estimated number of accessed sectors to detect a sector fault with the scanning algorithm (relative to the number of sectors in the disk)
$E_{Single\text{-}80/20}$	estimated number of accessed sectors to detect a sector fault with Single-80/20 access pattern (absolute value)
$\overset{\square}{E}_{Single\text{-}80/20}$	estimated number of accessed sectors to detect a sector fault with Single-80/20 access pattern (relative to the number of sectors in the disk)
$E_{Double\text{-}80/20}$	estimated number of accessed sectors to detect a sector fault with Triple-80/20 access pattern (absolute value)
$\overset{\square}{E}_{Double\text{-}80/20}$	estimated number of accessed sectors to detect a sector fault with Triple-80/20 access pattern (relative to the number of sectors in the disk)
$E_{Uniform}$	estimated number of accessed sectors to detect a sector fault with Uniform access pattern (absolute value)
$\overset{\square}{E}_{Uniform}$	estimated number of accessed sectors to detect a sector fault with Uniform access pattern (relative to the number of sectors in the disk)
$f(RA_s, RD_s, p_{fd})$	function to define the sector fault detection rate with a scanning algorithm
$f(RA_u, RD_u, p_{fd})$	function to define the sector fault detection rate with a user disk access pattern
h	history factor of the scanning algorithm
H	number of Hamming coded parity disks
$M1$	mission success probability of TMM for one year mission
$M1_I$	mission success probability of EMM1 for one year mission
$M1_{I,A}$	mission success probability of EMM1A for one year mission
$M1_{II,A}$	mission success probability of EMM2A for one year mission
$M10$	mission success probability of TMM for ten years mission
$M10_I$	mission success probability of EMM1for ten years mission
$M10_{I,A}$	mission success probability of EMM1A for ten years mission
$M10_{II,A}$	mission success probability of EMM2A for ten years mission

$M3$	mission success probability of TMM for three years mission
$M3_I$	mission success probability of EMM1 for three years mission
$M3_{I,A}$	mission success probability of EMM1A for three years mission
$M3_{II,A}$	mission success probability of EMM2A for three years mission
$MTTDL_{Indep}$	Mean Time To Data Loss in Gibson’s approximation
$MTTDL_I$	Mean Time To Data Loss of EMM1
$MTTDL_{I,A}$	Mean Time To Data Loss of EMM1A
$MTTDL_{II,A}$	Mean Time To Data Loss of EMM2A
$MTTDL_s$	Mean Time To Data Loss of simplified TMM
$MTTDL_{TMM}$	Mean Time To Data Loss of TMM
$MTTF_{disk}$	Mean Time To Failure of a disk
$MTTR_{disk}$	Mean Time To Repair of a disk
$p_0(t)$	probability of TMM to be in state *0* at time *t*
p_0	steady state probability of TMM to be in state *0*
$p_{0,s}$	approximation of the steady state probability of TMM to be in state *0*
$p_{00}(t)$	probability of EMM1 to be in state *00* at time *t*
$p_{00,A}$	approximation of the steady state probability of EMM1 to be in state *00*
$p_{000,A}$	approximation of the steady state probability of EMM2 to be in state *000*
$p_{001,A}$	approximation of the steady state probability of EMM2 to be in state *001*
$p_{01}(t)$	probability of EMM1 to be in state *01* at time *t*
$p_{01,A}$	approximation of the steady state probability of EMM1 to be in state *01*
$p_{010,A}$	approximation of the steady state probability of EMM2 to be in state *010*
$p_1(t)$	probability of TMM to be in state *1* at time *t*
p_1	steady state probability of TMM to be in state *1*
$p_{1,s}$	approximation of the steady state probability of TMM to be in state *1*
$p_{10}(t)$	probability of EMM1 to be in state *10* at time *t*
$p_{10,A}$	approximation of the steady state probability of EMM1 to be in state *10*
$p_{100,A}$	approximation of the steady state probability of EMM2 to be in state *100*
$p_{101,A}$	approximation of the steady state probability of EMM2 to be in state *101*
$p_{110,A}$	approximation of the steady state probability of EMM2 to be in state *110*
$p_2(t)$	probability of TMM to be in state *2* at time *t*
p_2	steady state probability of TMM to be in state *2*
$p_{2,s}$	approximation of the steady state probability of TMM to be in state *2*
p_f	probability of EMM1 to be in state *f*
$p_f(t)$	probability of EMM1 to be in state *f* at time *t*
p_{fd}	probability to detect a sector fault with a single disk access (assumed to be equal to one)
p_i	steady state probability for a system to be in state *i*
pfa	potentially faulty area of the scanning algorithm
Q_I	divisor in EMM1
$Q_{I,A}$	divisor in EMM1A
$Q_{II,A}$	divisor in EMM2A

$R_{Indep}(t)$	reliability of the disk array in Gibson's approximation at time *t*
$R(t)$	reliability of disk array at time *t*
$R_I(t)$	reliability of EMM1 at time *t*
r_i	root in EMM1
RA_s	average activity of the scanning disk read requests
RA_u	average activity of the user disk read requests
RD_s	distribution of the scanning disk read requests
RD_u	distribution of the user disk read requests
rs	request size of the scanning algorithm
S	number of sectors in a disk
S_a	total number of accessed sectors
S_i	dividend in EMM1
sa	start address of the scanning algorithm
sao	start address offset of the scanning algorithm
T_{scan}	time to scan a disk
$w_i(t)$	reward function of state *i* at time *t* in the performability model
W_0	average reward function of TMM at state *0*
W_{00}	average reward function of EMM1 at state *00*
W_{01}	average reward function of EMM1 at state *01*
W_1	average reward function of TMM at state *1*
W_{10}	average reward function of EMM1 at state *10*
W_2	average reward function of TMM at state *2*
W_{01}	average reward function of EMM1 at state *01*
W_i	average reward of the disk array at state *i*
W_f	average reward function of EMM1 at state *f*
wt	wait time of the scanning algorithm
$wt_{\max}$	maximum wait time of the scanning algorithm
$wt_{\min}$	minimum wait time of the scanning algorithm

LIST OF ABBREVIATIONS

byte	eight bits of information
CD ROM	Compact Disk Read Only Memory, optical storage media
CPU	Central Processing Unit
ECC	Error-Checking-and-Correcting memory or mechanism
EMM1	Enhanced Markov Model, version 1
EMM1A	approximation for Enhanced Markov Model, version 1
EMM2	Enhanced Markov Model, version 2
EMM2A	approximation for Enhanced Markov Model, version 2
FIM	Finnish Markka
HDA	Hardware Disk Array
I/O	Input/Output
IDE	Intelligent Drive Electronics
GB	giga byte
kB	kilo byte
MB	mega byte
Mbps	mega bits per second
MHz	mega Hertz
MTBDF	Mean Time Between Disk unit Failures

MTBF	Mean Time Between Failures
MTBSDF	Mean Time Between Second Disk unit Failures
MTBSF	Mean Time Between Sector Faults
MTTDL	Mean Time To Data Loss
MTTF	Mean Time To Failure
MTTOSD	Mean Time To Order and replace Spare Disk
MTTR	Mean Time To Repair
MTTRDF	Mean Time To Repair Disk unit Failure
MTTRSF	Mean Time To Repair Sector Fault
ms	millisecond
OLTP	On-Line Transaction Processing
PC	personal computer
RAID	Redundant Array of Inexpensive Disks, a disk array concept
RAID-0	striped disk array configuration
RAID-1xRAID-1	striped disk array configuration that is mirrored
RAID-1	mirrored disk array configuration
RAID-2	Hamming coded disk array configuration
RAID-3	disk array with byte/bit oriented parity
RAID-4	disk array with non-distribute block oriented parity
RAID-5	disk array with distribute block oriented parity
RAID-5+	non-standard RAID configuration
RAID-6	non-standard RAID configuration
RAID-7	non-standard RAID configuration
RAM	Random Access Memory
rpm	rounds per minute
SCSI	Small Computer System Interface
SCSI-1	SCSI standard version 1
SCSI-2	SCSI standard version 2
SCSI-3	SCSI standard version 3
SDA	Software Disk Array
SLED	Single Large Expensive Disk
tps	transactions per second
TMM	Traditional Markov Model

LIST OF TERMS

active disk	a disk that is actively used for storing data in the array (as opposite to a spare disk)
bad sector	a faulty sector that has lost its data
bathtub curve	a typical reliability model with high infant and old age failure rates and otherwise almost constant failure rate
check region	area that is scanned and after which the disk statistic information is read
cold format	disk format procedure where deteriorated areas are omitted
cold region	seldom accessed region in a disk
corresponding sector	same logical sector address as in an other disk
cost-performability	combined factor of cost, performance, and reliability of a system
crippled array	a disk array in which one of the disks has failed and it is not yet fully recovered
Curie point	the point of temperature where a disk loses its magnetic storage capacity
delayed repair	repair process that is started some time after detecting a fault
disk group	a group of disks that are used together for forming one disk array entity, for example a disk and its mirror in RAID-1
disk unit fault	a fault in a disk that results a total inoperability of the disk
ENABLE EARLY RECOVERY	SCSI command to expedite REASSIGN BLOCK -command activation after first problems in the disk
ERROR COUNTER PAGE	SCSI error diagnostic information
field return statistics	statistics that is gathered on failed hard disks that have been returned back to the manufacturer
head switch	time to wait when accessed data is spun over more than one disk surface and the disk must

	change from one head to another
high data locality	probability that the next disk access hits nearby the previous disk access is high
hot spare	a spare disk that is always available in the disk array
hot spot	commonly accessed area in disk
hot swap	a mechanism where a spare disk can be inserted into the array on fly
immediate repair	a repair process that is started immediately after detecting a fault
LAST N ERROR EVENT PAGE	a SCSI command to get more information for the last *n* error events
latent sector fault	a sector fault that is not yet detected
Markov model	a model for reliability transitions
masking effect	an effect of overlapping errors where the second error masks the detailed information of the previous error
memory scrubbing	an algorithm that scans primary memory in order to detect faults in RAM
mission success probability	probability that the system remains consistent over a given period of time
obstructed repair	a repair process that is started immediately after the fault detection but the repair process is not running at full speed
performability	combined factor of performance and reliability of a system
READ	a SCSI command to retrieve information from a disk
read-ahead	an algorithm to read in advance next blocks of data from a disk
REASSING BLOCK	a SCSI command to initiate sector fault repair process
reward	a function that describes the performability reward
rotation delay	average time needed to wait for the disk to rotate to right position
rotation speed	speed to rotate a hard disk
scanning cycle	a cycle during in which all sectors of the disks have been accessed by the scanning process
scanning request	a request that is used for reading a block of disk to detect latent faults
sector	smallest storage entity in a disk, typically 512 bytes
sector fault	fault that is caused by media deterioration in one location of a disk causing a sector to be incapable of storing data
seek time	time to move disk head from one track to another
spare disk	an extra disk to be used for disk repair process after a disk unit fault
stripe unit	unit of data interleaving in a RAID array
sustained transfer rate	continuous data transfer capacity to/from a hard disk
transfer rate	data transfer capacity to/from a hard disk
user	a generic entity that accesses disks for context of the data
VERIFY	a SCSI command to verify data in a disk
write-behind	an algorithm to store data into a disk cache and write it afterwards to disk
WRITE AND VERIFY	a SCSI command to write and immediately verify the written data

0

1. INTRODUCTION

Performance and reliability of computer systems have improved rapidly in recent years. However, these improvements have not been equal among all components. Some components, such as CPU (as measured by its clock speed or processing capacity), primary memory (as measured by its storage capacity and access time), and secondary memory (as measured by the storage capacity and reliability of a hard disk), have improved faster (increasing by 40-100% per year) while other components, such as I/O systems (as measured by the number of operations per second) or overall reliability (as measured by mean time between software or hardware faults), have improved at a much lower rate (e.g., the improvement in the average disk seek time has been only 7% in the same time span) [Lee 1991, Chen 1990, Chen 1990a].

The number of instructions executed per second has radically increased due to rapid development of the microprocessor technology. This is attributed to new manufacturing technologies, materials, and architectures. For example, by changing from 5V operating voltage to 3.3V, the processor clock can be doubled without increasing CPU power consumption. Also, several parallel architectures have been introduced to distribute processing among distinct processing units. Hence, the processing capacity (as measured by the number of operations per second) has increased significantly [Intel 1996a].

Similarly, the capacity of the primary memory has increased in recent years. However, this has mainly been required because the increase of the average size of application programs that has increased by 25-50% per year.

Requests for a faster I/O subsystem have emerged to satisfy the improvements in the other parts of the computer system. A faster disk subsystem is needed not only to match rapidly increasing performance of the other components but also to match larger programs and data sets.

The performance discrepancy of the various components in a computer system has increased continuously in recent years. The performance of some components is illustrated in Figure 1 relative to their values in 1985 (note the logarithmic scale) [IBM 1996a, IBM 1996b, IBM 1996c, Intel 1996, Intel 1996a, Seagate 1996c, Milligan 1994, Gray 1993]. These performance improvements have clearly been very uneven: the best improvement is in the disk capacity and the CPU speed while the components of the disk subsystem have improved significantly less. Especially, the properties that are related to physical disk constraints have improved only slightly. The I/O subsystem has unquestionably become the bottleneck of a computer system. Thus, the benefit of faster CPUs is easily diminished due to the slow I/O subsystem.

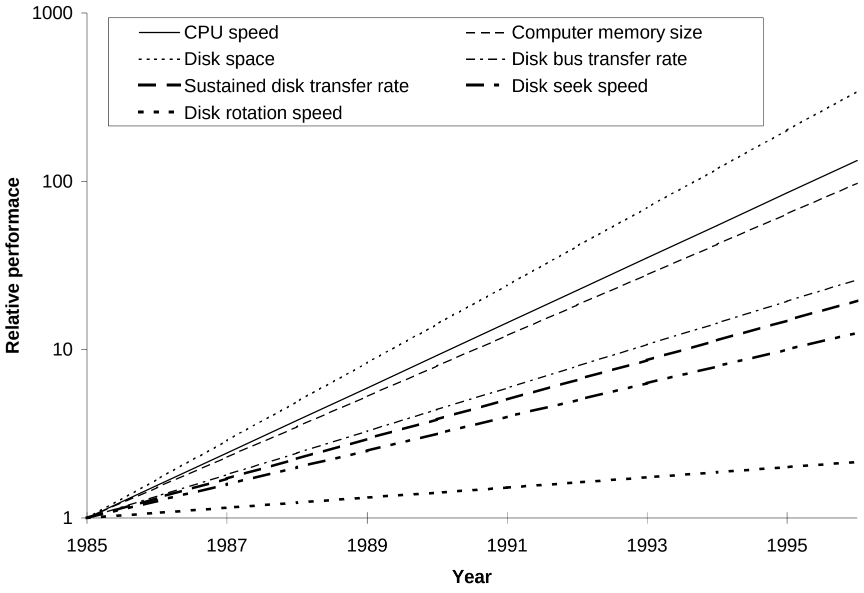

Figure 1. Trend of relative performance of various computer components

1.1 Improving disk subsystem

Several concepts of disk arrays have been introduced to reduce the increased gap of the performance improvement between CPU and disk I/O subsystem. The first ideas to overcome the performance limitations of a physical disk were based on mirroring and interleaving techniques [Kim 1987, Livny 1987, Kim 1986, Kim 1985]. Currently, one of the most popular concepts is the *redundant array of inexpensive disks* (RAID) [DPT 1993, RAB 1993, Gibson 1991, Lee 1991, Chen 1990, Chen 1990a, Lee 1990, Chen 1989, Katz 1989, Chen 1988, Patterson 1988, Patterson 1987, Salem 1986]. In the RAID concept, several disks are used in parallel to improve throughput, transaction rate, and/or reliability.

Better throughput in a disk array is achieved by utilizing several disks for one large user I/O request. When a large contiguous data block is accessed, the bottleneck of a conventional disk subsystem is the sustained transfer rate from the disk surface [Reddy 1990a, Katz 1989, Ousterhout 1988]. By accessing several disks in parallel, it is possible to achieve higher transfer throughput as data can be fetched/stored simultaneously from/to several disks. The disk bus transfer capacity is typically much higher (in order of five to ten times) than that of a single disk. In large arrays, several disk buses are used for providing high bandwidth [Seagate 1996a, Seagate 1996b, Seagate 1996c, Milligan 1994, Gray 1993, Hillo 1993]. In addition, modern disks contain large internal buffers to store data temporarily if the disk bus is momentarily used by other disks for data transfer. Hence, the operations of different disks can be overlapped so that while one disk is transferring data over the disk bus, the others can do seeks and gather data into their buffers or write data from their buffers into the disk.

When many disks are used in parallel, the number of faults increases significantly and reliability will be an important factor. In this thesis, the interest is focused on the question how high reliability can be offered when also high performance is required. Especially, the effect of latent faults is studied.

Disk array types

The most common RAID array configurations are listed in Table 1. The second column briefly illustrates the structure of each array configuration as listed in the technical literature [DPT 1993, Hillo 1993, RAB 1993, Gibson 1991, Lee 1991, Chen 1990, Chen 1990a, Lee 1990, Chen 1989, Katz 1989, Reddy 1989, Chen 1988, Patterson 1988, Patterson 1987, Salem 1986]. The number of disks needed to implement an appropriate array configuration is then listed in the third and fourth columns. Here, D indicates the number of data disks (used for storing user data) and H is the number of parity disks (if the number of parity disks depends on the number of data disks as in the case of Hamming coding where $D \le 2^H - H - 1$) [Gibson 1991, Gibson 1989a, MacWilliams 1977, Peterson 1972]. The last column in the table illustrates the storage efficiency of the disk array architectures using an example of ten data disks. Beside the RAID types listed here, there are also other array types such as two dimensional parity and arrays with non-binary symbol codes [Stevens

Table 1. Various RAID configurations

Array type	Array structure	Number of data disks	Number of disks used for redundancy	Typical data storage efficiency (for 10 data disks)
RAID-0	Striped, no redundancy	D	0	100%
RAID-1	Mirrored*	D	D	50%
RAID-2	Hamming coded	D	H	71%
RAID-3	Bit/Byte oriented parity	D	1	91%
RAID-4	Striped with non-distributed, block oriented parity	D	1	91%
RAID-5	Striped with distributed, block oriented parity	D	1	91%

* The disk mirroring technique was not invented in the RAID concept. For example, Tandem computers used mirrored disks already in 1970’s. However, RAID-1 is commonly used for describing mirrored disks as one alternative of disk array configurations.

1995, Schwarz 1994, Katz 1993, Mourad 1993a, RAB 1993, Lee 1992, Gibson 1991, Stonebraker 1989].

Performance effect

The performance of a disk array can be roughly expressed with two interrelated parameters: the maximum throughput of the system and the maximum number of simultaneous disk requests [Hou 1994, DPT 1993a, Geist 1993, Hillo 1993, Mourad 1993, Reddy 1991a, Chervenak 1990, Seltzer 1990, Seltzer 1990a, Olson 1989, Gibson 1989, Reddy 1989, Stonebraker 1988, Cox 1986, Kim 1986, Kim 1985]. These parameters are illustrated in Table 2. The different array configurations provide a variety of maximum throughput and maximum number of simultaneous read/write operations.

Importance of reliability

Originally, the fault tolerant disk subsystems were used only in expensive mainframe architectures, such as *online transaction processing* (OLTP) databases in banking applications, while smaller computers, such as PCs, had no fault tolerant features. Nowadays, even desktop PCs may have several disks containing several gigabytes of data. As the number of disks in modern computer systems increases and the user has more and more precious data stored onto them, the requirement for reliable data storage becomes imminent even in smaller computers. The RAID concept provides an answer also for this question allowing a wide range of architectures, with respect to cost, performance, and reliability, to be used.

Disk faults

Traditionally, disks in a disk array are considered to have only one type of fault that disables the

Table 2. Performance of RAID configurations

Array type	Number of disks in the array	Maximum total throughput (relative to a single disk)	Maximum number of simultaneous read operations	Maximum number of simultaneous write operations
RAID-0	D	D	D	D
RAID-1	$2D$	$2D$	$2D$	D
RAID-2	$D+H$	D	1	1
RAID-3	$D+1$	D	D	1
RAID-4	$D+1$	D	D	1
RAID-5	$D+1$	$D+1$	$D+1$	$(D+1)/2$

entire disk [Hillo 1993, Gibson 1991, Cioffi 1990, Muntz 1990, Sierra 1990, Schulze 1988, Williams 1988]. However, there is in practice a variety of possible faults in a disk unit (such as head, actuator, surface, motor, connector, controller, and bearings faults) [Haseltine 1996, ANSI 1994, ANSI 1986]. The two most common faults are the above mentioned disk unit fault and a sector fault that effects only small part of the disk (just one sector). The actual reliability statistics are not revealed by disk or computer manufacturers [Hillo 1994, Gibson 1991].

Fault detection

Faults in a disk array are detected sooner or later either by normal user disk requests or by some diagnostic activity [IBM 1996d, Ottem 1996, Scritsmier 1996, Kari 1994, Kari 1993, Kari 1993a, Kari 1993b]. In the case of a disk unit fault, the fault detection is fast [ANSI 1997, ANSI 1994, ANSI 1986], but sector faults may remain undetected for a long time [Kari 1994, Kari 1993, Kari 1993a, Kari 1993b]. The latent faults have a significant effect on the reliability of the disk array as a conventional disk array can tolerate only one simultaneous fault in a disk group[*]. According to practical experience in the industry, significant part of the cases when a disk array has lost its consistency were caused by the latent sector faults[†] [Scritsmier 1996].

Array repairs

A disk array is conventionally repaired using a spare disk into which data of the faulty disk is reconstructed using the redundant information of the data stored in the remaining disks. Depending on the array type, one or more disks in the array are involved with the repair process.

Sector faults are usually ignored by conventional reliability models and repair processes. This is partly because the modern disks can automatically repair faulty sectors by themselves during a write operation [ANSI 1997, Räsänen 1996, Scritsmier 1996, ANSI 1994, Räsänen 1994, Platt 1992, ANSI 1986]. Practical disk array implementations, on the other hand, can do automatically data recovery of the lost data during the read operation using the redundant information stored in the other disks. Because a disk array is generally accessed unevenly [Hou 1994, Ylinen 1994, Seltzer 1992, Miller 1991, Reddy 1991a, Chen 1990, Seltzer 1990a, Olson 1989, Reddy 1990a, Bhide 1988, Ousterhout 1988], it is not possible to detect all sector faults using only normal user disk requests.

[*] A *disk group* forms one disk array entity, for example a disk and its mirror in RAID-1.

[†] If a faulty sector remains undetected for a long time, it is called a *latent sector fault*. The sector fault may remain latent because it is only detected when the sector is accessed what may take a long time.

Reliability of disk arrays

As most of the disk array configurations are built to tolerate only one fault before repair (in order to simplify the data update on the disk array), the reliability of the disk array is dominated by the repair time of the array [Hou 1994, Schwarz 1994, Burkhard 1993, Chandy 1993, Geist 1993, Gray 1993, Hillo 1993, Gibson 1991, Reddy 1991, Sierra 1990, Gibson 1989, Gibson 1989a, Chen 1988, Garcia-Molina 1988, Schulze 1988, Stonebraker 1988]. This is the case when the sector faults are ignored [Kari 1994, Kari 1993, Kari 1993a, Kari 1993b]. If the sector faults are included, the latent sector faults are as significant as the repair time. When latent sector faults remain undetected for a long time, the reliability will be dominated by the latent sector faults.

Reliability (as expressed with *Mean Time To Data Loss*, MTTDL) is illustrated as a function of disk array configuration in Table 3 [Hillo 1993, Gibson 1991, Gibson 1989] with the same disk repair and failure times. This example assumes that the number of data disks is 10, *Mean Time To Failure*, MTTF, of a disk and *Mean Time To Repair*, MTTR, of a disk array are 100 000 and 24 hours, respectively. A second method used for expressing the reliability of a disk array is the *mission success probability* that gives the probability of an array to survive over a given time without losing its consistency [Gibson91].

Trade-offs

There will always be a trade-off between cost, performance, and reliability as it is shown in Tables 1, 2, and 3. Typically, the better performance is required, the less reliable service a disk array can provide, and vice versa [Hou94, Hillo93, Gibson91, Muntz90]. This is especially true with the write operation. Thus, it is not possible to have one array architecture that is the best for all environments and user needs. For example, the following issues must be addressed when selecting a disk array architecture:

Table 3. Reliability of sample RAID configurations (MTTF=100 000h, MTTR=24h)

Array type	Number of disks in the array	MTTDL of the disk array [million hours]
RAID-0	10	0.01
RAID-1	10+10	20.8
RAID-2	10+4	2.3
RAID-3	10+1	3.8
RAID-4	10+1	3.8
RAID-5	10+1	3.8

1. **Cost** issues:
    - cost of redundancy
    - cost of increased overall complexity
    - cost of system failure or data loss
    - cost of repair process
    - scaleability
2. **Performance** issues:
    - high I/O rate or high transfer capacity
    - I/O patterns
    - size of I/O requests
    - ratio of read to write operations
    - crippled* array performance
    - performance degradation due to repair process
    - performance degradation due to preventive maintenance or fault detection
3. **Reliability** issues:
    - mean time to data loss
    - data availability
    - recovery speed
    - on-line or off-line recovery
    - detection of latent faults
    - effect of interrelated faults

Two terms have been introduced to express combination of the above mentioned parameters: *performability* and *cost-performability* [Trivedi 1994, Catania 1993, Pattipati 1993, Smith 1988, Furchgott 1984, Meyer 1980, Beaudry 1978].

Performability defines the combined performance and reliability of the system. The main problem in defining the performability formula is the different scales of the performance and reliability factors. The scale of the reliability (as expressed with MTTDL) can vary by several orders of magnitude while the scale of the performance (as expressed with the number of disk operations per second) is almost linearly related to the number of disks in the array.

Cost-performability expresses the trade-off between performability and how much it costs to

* A *crippled array* is an array in which one of the disks has failed and it is not yet fully recovered. When the failed disk is accessed, the data is reconstructed using other disks in the array which may require an access of all other disks in the array.

operate such a disk array system. On the cost side of the equation, installation, running, and failure costs are typically included. An additional problem of the cost-performability is the ambiguity of the cost definition. In contrast to reliability and performance that can be expressed with exact, commonly agreed terms, the cost has a subjective nature: for example, each user may have one's own opinion about the cost of data loss.

Practical disk array implementations and theoretical models

Analysis of practical implementations of the disk array architectures can differ significantly from the theoretical models. This is mainly because several simplifications are made in the modeling process of the disk array architectures. For example, simple fault models, uniform access patterns, and independent failure rates are commonly assumed in theoretical models, but they are seldom the case in the practical implementations.

Simple fault models, such as the assumption that there is only one type of fault in a disk, are often used for modeling a disk array [Chandy 1993, Geist 1993, Gibson 1991, Gibson 1989a]. Such models also assume fast fault detection. Thus, it is possible to have an analytical approach to the reliability problem as the number of states in a Markov model is reduced, especially in the non-steady state analysis. These simple models ignore sector faults that commonly occur, but are more difficult to detect. As the sector faults can remain latent for a long time, the recovery process will start much later causing higher probability of having another fault in the same array before the first one is repaired. Hence, the reliability of the practical disk array system may be much lower than what is theoretically estimated.

In a practical disk array system, disk requests are spread quite unevenly over the disk space. The way the disk accesses are spread depends on, among other things, the type of the application, the operating system, and the number of simultaneous users. The main effect of this is that disk accesses tend to concentrate on certain areas (so called *hot spots*) of the disk space while other areas are rarely accessed (so called *cold regions*) [Hou 1994, Ylinen 1994, Seltzer 1992, Miller 1991, Reddy 1991a, Chen 1990, Reddy 1990, Seltzer 1990a, Olson 1989, Bhide 1988, Ousterhout 1988]. Therefore, fault detection due to normal disk accesses is limited. In the case of a fault in the seldom accessed area, the reliability can decrease significantly due to this latent fault. The probability of latent faults (especially sector faults) increases when larger and larger disk arrays are used. The uneven access patterns have also some effect on the performance of the disk array.

Faults in a disk array are often assumed to be independent. However, there are several reasons why the faults can actually be interrelated. First, the operating environment is usually the same for all disks (e.g., they use the same cooling system, power supply, or disk controller) [Räsänen 1996, Räsänen 1994, Seagate 1992, Seagate 1992a, Gibson 1991]. Second, disks in a disk array come very likely from the same manufacturing batch and therefore they are all prone to same manufacturing quality problems in both hardware and software wise [Hillo 1994, Gray 1993, Hillo 1992]. Besides, due to the complexity of the disk arrays, they are also prone to all kind of software errors that occur especially during heavy load [Kamunen 1994, Räsänen 1994]. Hence, the failure rate of a disk array is in practice much higher than theoretically estimated. Especially, the probability of a second fault just after the first one is significantly higher in a practical system than in theory (when the faults are independent).

In summary, the theoretical reliability and performance figures of a disk array are never achieved in practice, but are limited by the actual implementation. Also, simplification of the system modeling reduces accuracy of the theoretical results.

1.2 Objectives of this thesis

In this thesis, the main objective is to study the effect of latent sector faults in reliability of disk arrays. This is done by introducing and analyzing detailed reliability models for a disk array that have two types of faults: disk unit faults and sector faults. Two repair models are analyzed: an array with *hot spare* and an array with *hot swap.*[*] Both analytical and approximation methods are used for analyzing the reliability. Also, the effect of the related faults is studied.

Two reliability metrics are used in this thesis: *data availability* and Mean Time To Data Loss (MTTDL). The data availability is used for illustrating the probability of the system of maintaining the data for a given time. This can be used, for example, for expressing the 1-year, 3-years, and 10-years mission success probabilities [Gibson 1991]. The second term, MTTDL, is also used for expressing the quality of the disk array with respect to reliability and availability by expressing the estimated time the system maintains data integrity [Gibson 1991, Pages 1986, Siewiorek 1982, Shooman 1968].

The second objective of this thesis is to study the effect of latent fault detection techniques in disk arrays. A novel idea is to scan a disk while it is otherwise idle and to detect deteriorated areas in the disk before it loses its capability to maintain data storage. Besides, this scanning process can also be used for expediting the detection of the latent sector faults. This is especially important in systems that tolerate only one fault before repair and where latent faults are probable due to uneven

[*] A *hot spare* array has a spare disk already installed and can start the repair immediately after a fault detection. The *hot swap* array requires human intervention and, therefore, has a slower recovery process.

access patterns and large number of disks.

The third objective of this thesis is to study the effect of the latent fault detection technique on performability. Latent fault detection increases the reliability of the disk array as the probability of data loss is significantly reduced when the latent faults are eliminated faster. On the other hand, latent fault detection reduces the performance of the disk array as part of its time is not available for user requests.

1.3 Structure of this thesis

This thesis is divided into six parts: introduction, basic principles of latent fault detection, reliability modeling and analysis, performability analysis, a disk subsystem as a part of a computer system, and conclusions.

The first part of the thesis contains an introduction to disk arrays and some background information for this research. The first chapter contains an overview on the background of computer systems with disk arrays. In Chapter 2, previous studies are briefly reviewed in the area of performance and reliability evaluation of disks and disk arrays. Motivations of this thesis are then discussed in Chapter 3. Models of disks and disk arrays are reviewed in Chapter 4.

The second part of this thesis discusses the idea of detecting latent faults. In Chapter 5, the main principles of disk scanning algorithms are presented.

The main part of this thesis is the analysis of the disk arrays. Here, the reliability of two different disk array configurations (RAID-1 and RAID-5) is studied. In Chapter 6, the prerequisites of the reliability analysis are discussed. The assumptions are divided into three categories: reliability metrics, reliability measurements, and basic assumptions of the components in the reliability model. Then, two reliability models are formulated in Chapter 7. Both exact analytical and approximation approaches are used. The reliability models are analyzed in Chapter 8. The new reliability models are compared with results in the technical literature. Several aspects on reliability are discussed in this chapter comparing and analyzing the reliability models. Nine reliability scenarios are also analyzed in this chapter.

The fourth part studies the performability of a disk array. Chapter 9 addresses the performability issues. A performability model is built and analyzed for RAID-5 arrays. Also, a simple performability approximation is derived for RAID-1 and RAID-5 arrays.

The fifth part discusses briefly the disk array as a part of a computer system. Reliability of an entire computer system including a disk array is reviewed in Chapter 10.

Finally, the conclusions are presented in Chapter 11.

1.4 Contributions of this thesis

The contributions of this thesis can be divided into the following three main areas:

- new studies on the effects of latent faults on the reliability of a disk array,
- new reliability models for disk arrays taking into account both disk unit faults and sector faults, and
- new scanning algorithms to detect latent sector faults.

Effect of latent faults on disk array reliability

The main contribution of this thesis is its emphasis on the adverse reliability effect of the latent faults. In the technical literature, reliability analysis has concentrated on disk unit faults ignoring the possible effects of latent sector faults. As common disk array architectures are capable of handling only one fault in a disk group before repair, latent faults decrease the reliability of the disk array dramatically. As shown in this thesis, user initiated disk requests do not efficiently detect latent sector faults because of their uneven access patterns. This thesis illustrates the importance of latent fault detection by showing how significantly the reliability will decrease if the latent faults are not detected early enough. Latent fault detection can be done using simple algorithms as shown in this thesis thus dramatically increasing the reliability. Detailed analysis of the reliability of a disk array with various aspects related to latent faults is also included into this thesis.

Also, the possibility of delaying or slowing down the repair process is studied. This is important in systems where high data availability is needed, but performance degradation during the repair state must still be limited. In this thesis, the speed of the repair processes is studied in order to improve performability. The repair process can be delayed to reduce the performance degradation effect of the repair process on the user disk requests at the expense of reduced reliability.

New reliability models

This thesis presents also two novel reliability models for a disk array that include both disk unit and sector faults: one model for an array with hot swap and another for hot spare disks. The hot swap model is simpler as it assumes that the spare disks are fault-free while the hot spare model allows faults to occur also in the spare disk. The former model is analyzed using both an exact analytical method and an approximation method while the latter one, due to its complexity, is only analyzed using approximation. These reliability models are also used in the performability analysis.

Methods to detect and remove latent faults

New methods to detect and remove latent faults in order to maintain high reliability and data availability are introduced and analyzed. Three conference publications on disk scanning algorithms by the author are referred to. These algorithms utilize a similar approach as the *memory scrubbing* algorithms in detecting latent faults in the primary memory [Saleh 1990]. However, the publications were the first ones proposing such algorithms to be used for scanning secondary storage (i.e., hard disks). As disks behave differently from the primary memory, the new disk scanning algorithms are adapted for the disk usage.

2. PREVIOUS STUDIES

Performance and reliability of the disk subsystems have been widely studied in the technical literature [Hou 1994, Schwarz 1994, Burkhard 1993, Chandy 1993, Geist 1993, Hillo 1993, Mourad 1993, Gibson 1991, Kim 1991, Lee 1991, Ng 1991, Reddy 1991, Reddy 1991a, Chen 1990, Chen 1990a, Chervenak 1990, Lee 1990, Seltzer 1990, Seltzer 1990a, Chen 1989, Gibson 1989, Gibson 1989a, Olson 1989, Reddy 1989, Chen 1988, Garcia-Molina 1988, Schulze 1988, Stonebraker 1988, King 1987, Livny 1987, Cox 1986, Kim 1986, Kim 1985]. Some studies have concentrated on improving disk subsystem performance while some others have dealt with the reliability issues, but also combined performance and reliability has been analyzed. In addition, disk array repair methods, fault prevention mechanisms, performability, and cost-performability have been studied.

A short overview of various branches of performance and reliability analysis of disks and disk arrays is given in this chapter.

2.1 Improving disk subsystem performance

The performance of a disk subsystem has been traditionally improved using one of three approaches: improving the software, improving the disk hardware, or using an array of disks.

2.1.1 Software approach

The performance of the disk subsystem can be improved without changing the hardware. This can be done, for example, by optimizing the disk accesses, their order, or time when they are executed.

Caching

One of the easiest methods to improve disk subsystem performance is to store the previously used disk requests for further use. As the same disk locations are very likely to be accessed again, this disk caching can significantly reduce the number of actual disk requests. This reduces the average response time as significant portion of the requests is completed in practice immediately while the disk load is also reduced thus shortening the response times of the remaining disk requests. There have been several studies of caching algorithms, such as [Thiebaut 1992, Jhingran 1989, Nelson 1988, Ousterhout 1988, Koch 1987, Grossman 1985, Coffman 1973].

Read-ahead

It is not always enough to cache the previous disk requests to speed up the current disk requests.

The performance can be further improved by using the idle time of the disk subsystem to read in advance those disk areas that are most likely to be accessed next. It has been shown that it is very probable that a user will access locations nearby the previous location [Hou 1994, Bhide 1988]. Sequential disk reads especially benefit from this read-ahead feature.

Write-behind

In the write-behind scheme, the main idea is to store the disk write requests, rearrange them, and optimize the disk usage [Seltzer 1990a]. By rearranging the disk write requests, it is possible to shorten the average disk seek length. If a user writes several times to the same location, it is possible to reduce the number of actual disk writes by writing only the last update. With a battery backed memory, this can allow a long delay before a physical write operation if most requests are reads.

File system enhancements

Improving the disk I/O is not the only method of improving the performance. Performance can be further improved by arranging the files in the file system so that the files of the same directory are located nearby each other as it is very likely that they are accessed at the same time [Seltzer 1993, Rosenblum 1992, Seltzer 1992, Dibble 1989, King 1987, Koch 1987, Ousterhout 1985]. Hence, the operating system can optimize the file location by making the files contiguous and rearranging the files by the directories.

2.1.2 Hardware approach

The main problem with these software enhancements is that the disk subsystem can have only limited improvements in the disk performance as the disks have physical constraints such as rotation and seek delays, and disk transfer rate. Although disks have improved in these matters significantly in recent years, the improvement has not been as rapid as the development in the other parts of the computer architecture [IBM 1996a, Lee 1991, Chen 1990, Chen 1990a].

Beside the software enhancements, hardware improvements on disk subsystem have also been proposed as listed below. According to these ideas, it is possible to improve the architecture of a single disk or to use a set of physical disks as one logical disk.

Improving disk architecture

One of the main limitations of a conventional hard disk is that it can serve only one request at a time. This is due to the fact that it has only one read/write head that accesses the physical storage media.

Two types of proposals to enhance the disk architecture in this approach are: to add dependent or independent heads [Orji 1991, Cioffi 1990, Sierra 1990, Cox 1986, Coffman 1973].

In the former case, the heads are using the same physical arm and the distance between the heads is fixed (e.g., half of the distance between the first and the last tracks of the disk). The main benefit of this approach is that the average disk seek distance can be reduced by half as the first head handles the first half of the disk and the other one handles the second half. The main disadvantage of this approach is that only one request can be in process at any time.

In the latter case, the disk is equipped with two (or more) independent disk arms that can all access the entire disk space. The main benefit of such arrangement is that it is possible to serve several disk requests at the same time. The disk itself or the disk controller can optimize the disk accesses (both seek and rotation delays) by using that head that is the closest to the requested area. This also improves slightly the reliability as the disk can still operate even if one of its heads is faulty.

Arrays of disks

An alternative method to improve the physical properties of a single disk is to use several physical disks as one logical entity. The operating system or the device driver of the disk subsystem divides the incoming requests to appropriate disks depending on the arrangement of disks. A typical example of this approach is disk mirroring.

2.1.3 Redundant array of inexpensive disks

The problem of dedicated hardware approach is slow development and vendor dependency. As the special disks do not generally follow any standards, the implementation depends on a single manufacturer. Also, the single disk approach provides no protection against disk faults. In addition, the probability of error free media goes down as the area of the disk increases. Therefore, in contrast to the *SLED* (Single Large Expensive Disk) approach an alternative approach: RAID (Redundant Array of Inexpensive Disks) is used. In RAID, the disk I/O performance is improved by enhancing the disk subsystems by combining a set of disks to work together as one logical disk. Several studies (as noted below) have been reported in this area.

The concept of redundant array of inexpensive disks (RAID) is one of the most popular approaches for disk arrays [DPT 1993, RAB 1993, Gibson 1991, Lee 1991, Chen 1990, Chen 1990a, Lee 1990, Chen 1989, Katz 1989, Chen 1988, Patterson 1988, Patterson 1987, Salem 1986]. The RAID concept was introduced to improve the performance and/or the reliability of the disk subsystem. This concept utilizes models for different disk array algorithms, like mirroring, striping, and striping with parity.

Several RAID models (e.g., RAID-0, RAID-1, RAID-2, RAID-3, RAID-4, RAID-5) are developed for different purposes.* More detailed description of the different RAID models can be found in Chapter 4 and in [Hillo 1993, RAB 1993, Gibson 1991, Patterson 1987].

Beside these simple arrays, there are also more complex arrays such as RAID-0xRAID-1 (i.e., mirrored RAID-0 array) or RAID-5+ [Stevens 1995, Schwarz 1994, Hillo 1993, Katz 1993, Mourad 1993a, RAB 1993, Lee 1992, Stonebraker 1989]. At University of California in Berkeley, a second generation RAID concept has been developed to enhance the array architecture [Katz 1993, Lee 1992].

The performance of disk arrays is mainly optimized to store and retrieve data when no error has occurred [Hou 1994, Geist 1993, Hillo 1993, Mourad 1993, Chen 1992, Kemppainen 1991, Lee 1991, Olson 1989, Reddy 1991a, Chen 1990, Chervenak 1990, Lee 1990]. The performance of the arrays in a degraded state is considered to be of lesser importance because this is assumed to be an infrequent state.

2.2 Improving disk subsystem reliability

Reliability is typically maintained passively [Hillo 1993, Gibson 1991]. Disk faults are detected by the normal disk requests, i.e., no special disk activity is done to detect faults. The reliability of the disk subsystem has been improved using two approaches: improving the reliability of a single disk and using redundant disks.

2.2.1 Improved disk reliability

The reliability of a single disk has improved significantly in recent years [Quantum 1996a, Seagate 1996c, Nilsson 1993, Faulkner 1991, Gibson 1991]. This improvement has been due to the enhancements in the disk mechanics (reduced physical size and improved materials) as well as the new algorithms to estimate the disk reliability from the field returns [Quantum 1996a, Nilsson 1993, Gibson 1991].

Two major problems with the reliability of a single disk are: uneven reliability and high reliability requirements. It has been shown that the reliability of the same type of disks varies much (even several orders of magnitude) among the individual disks and manufacturing batches [Hillo 1994, Hillo 1992, Gibson 1991]. On the other hand, the reliability of the disk subsystem becomes

* There are also other RAID models, but the above mentioned are the ones that have a fixed meaning. For example, there are a few alternative interpretations of RAID-6 or RAID-7 models [RAB 1993].

more and more important as the amount of data stored on the disks increases.

2.2.2 Redundant disk arrays

Reliability of the disk subsystem has mainly been improved by introducing disk array concepts [RAB 1993, Lee 1991, Lee 1990, Katz 1989, Chen 1988, Patterson 1988, Salem 1986]. The main idea for the array is that the number of disk faults increases quite linearly with the number of disks in the array. Hence, to survive the increased fault rate, the array should have some redundancy to tolerate at least one disk fault.

A typical disk array tolerates only one fault in a disk group [Hou 1994, Schwarz 1994, Burkhard 1993, Chandy 1993, Holland 1993, RAB 1993, Gibson 1991, Reddy 1991, Muntz 1990]. The main reason for this is that higher level redundancy would cause lower performance.[*] The reliability of such systems is kept high by expediting the repair process in order to minimize the time when the system has no available redundancy.

The repair process can be expedited by one of three methods: speeding up the repair process, starting the repair process earlier or detecting the faults earlier. Typically, the first two methods are used. The repair process is typically expedited by giving it priority over the normal user requests [Hillo 1993, Gibson 1991]. The repair process can be started earlier if the system has hot spare units that can be used for the recovery immediately after the fault is detected [Gibson 1991, Pages 1986, Siewiorek 1982, Shooman 1968]. If the repair is started a long time after the fault detection (e.g., the faulty unit must be replaced by a serviceman or the spare part must be ordered after the fault detection), the reliability decreases dramatically [Gibson 1991].

The same RAID concepts and RAID configurations (except RAID-0) can also be used for improving the reliability as well as the performance [Hillo 1993, RAB 1993, Gibson 1991]. However, the different RAID concepts have quite different behavior in terms of performance and reliability. For example, the RAID-1 array with two disks has the best reliability figures of all arrays that tolerate a single fault, but the performance or the cost may not be acceptable (e.g., the write throughput for a large disk I/O is not better than with a single disk). On the other hand, RAID-4 and RAID-5 arrays have significantly better performance (especially for read operations), but the reliability is much worse than that of a RAID-1 array as the large number of parallel data disks is secured with only one parity disk.

There are also proposals for systems that can survive two or more faults in the same group of

[*] When an array tolerates at most one fault, each user write operation requires a minimum of two disk writes. If two faults are tolerated, then a minimum of three disk writes are needed for every user write operation.

disks [Stevens 1995, Schwarz 1994, Mourad 1993a, Gibson 1991]. However, they are usually considered only for extremely highly reliable systems where the reliability concerns override the performance.

2.3 Reliability analysis

The reliability analysis of a disk array is quite widely studied [Hou 1994, Schwarz 1994, Burkhard 1993, Chandy 1993, Geist 1993, Gray 1993, Hillo 1993, Gibson 1991, Reddy 1991, Sierra 1990, Gibson 1989, Gibson 1989a, Chen 1988, Garcia-Molina 1988, Schulze 1988, Stonebraker 1988]. Three main approaches are: exact analytical, measurements, and simulation.

In the exact analytical approach, the reliability analysis is based on a Markov model of the disk array [Schwarz 1994, Hillo 1993, Gibson 1991]. The main problem with this analytical approach is that the analysis quickly gets complicated when the reliability model is made more accurate. Typically, the disk array is modeled with a non-steady state Markov model where the system has at least one sink (i.e., data loss state). Hence, the equations become complex even with simple models. It is also possible to obtain estimates of the reliability by using an approximation approach as presented in [Gibson 1991].

The second alternative is to measure existing disk array systems, but this is typically considered to be unfeasible as the number of arrays is small and the mean time between faults is very long.

The third alternative is to use a simulation approach like in [Sahner 1987, Sahner 1986]. Here, the flexibility of the simulation programs makes it possible to create more complex models that have various behavior patterns (such as non-constant failure rates or dependent faults).

2.4 Disk array repair algorithms

The repair time of a disk array is typically considered to be so short that its effect on the performance is quite insignificant as stated for example in [Hillo 1993, Gibson 1991]. This is true in practice when the long term average response time or the number of requests per second are studied as the repair time is typically a few hours while the average time between faults in a disk array is tens or even hundreds of thousands of hours [Quantum 1996a, Seagate 1996c, Faulkner 1991, Gibson 1991].

Typically, the main principle of the disk array repair algorithms is “repair as fast as possible to minimize the risk of having the second fault before the repair”. This is a reasonable approach when only reliability is considered, but the higher reliability is achieved at the expense of worse performance during the repair time [Muntz 1990].

If the repair time can be selected so that the repair can be done during the low load period of the system, the performance degradation due to the repair process can be significantly lowered. Unfortunately, this is not always possible. First, the reliability may suffer too much if the repair process is delayed several hours or days because of the current heavy load in the system [Gibson 1991]. Second, it is not always possible to postpone the repair for a more suitable time as the system may be loaded continuously with the same load, i.e., with no idle period [TPC 1992, TPC 1992a]. Besides, the performance of the crippled array is often significantly worse than that of a fully working array.

2.5 Fault prevention algorithms

The third method, that was mentioned above, to improve reliability is to expedite the fault detection. Typically, the fault detection is not done actively in disk arrays as it is considered that there are only disk faults and they are rapidly detected by the normal user disk requests. This is not, unfortunately, the case when sector faults are also considered [Scritsmier 1996, Cioffi 1990, Sierra 1990, Schulze 1988, Williams 1988]. In this case, the sector fault can remain undetected for a long time [Kari 1994, Kari 1993, Kari 1993a, Kari 1993b].

Fault detection can be improved by using the idle time of the system to diagnose the system status. When there is nothing else to do, the disk subsystem can be gradually read in small blocks so that the user disk requests are not disturbed too much if a user disk request comes while the scanning request is still being processed [Kari 1994, Kari 1993, Kari 1993a, Kari 1993b].

The disk scanning algorithm uses the same basic principle as the memory scrubbing algorithm [Saleh 1990]. In the memory scrubbing algorithm, the idle time of the computer is used for scanning the primary memory of the computer to find defected areas.

2.6 Performability analysis

A new term, performability, has been introduced in order to combine the metrics of performance and reliability of a computer system [Trivedi 1994, Catania 1993, Pattipati 1993, Smith 1988, Furchgott 1984, Meyer 1980, Beaudry 1978]. The main idea for the combination is to allow comparisons of different configurations of alternative models when both the performance and the reliability are important.

The basic idea of performability is to use a Markov reward model. In the Markov reward model, the system is given a *reward* for every state of a normal Markov state model. The reward can be, for example, the performance of the system in that state as expressed with the number of operations per second. When the reward in each state and the probability of being in that state are known, the performability can be formed as the sum of rewards weighted with the probabilities of the conventional Markov model.

The combined analysis of performance and reliability (performability) has become one of the key issues in modern disk subsystems. The higher reliability usually causes performance degradation on disk write requests as the same data must be stored in multiple locations [RAB 1993]. On the other hand, redundancy in the system is essential for a disk subsystem to survive media faults.

Cost-performability analysis

Cost-performability is analyzed in a similar way as performability. Here, the cost of the system is also recognized. The costs of the system will then include factors such as initial installation cost, cost to run the system, cost for a system failure, and cost to reestablish the system after a failure.

2.7 Other related studies

The performance research and analysis of disk subsystems has generally been performed under steady state conditions. However, performance during a fault recovery process is also important, especially when the disk subsystem is used in a real-time environment where strict response time requirements must be met. For example, the mean response time is not sufficient to ensure that a disk subsystem can continue operations to store or retrieve data for a multimedia application such as audio or video playback and recording. Usually, performance is guaranteed only under either a steady or a degraded state, but not while the system is under repair of a disk fault [Gibson 1991, Muntz 1990].

One of the disk array performance analysis studies made for the repair process of a disk array is [Muntz 1990]. In this analysis, the performance of the disk array is analyzed not only during the normal operation of the array, but also during its repair phase.

3. MOTIVATION OF THIS RESEARCH

The main objective of this thesis is to study the effect of latent sector faults in disk array reliability. The motivation behind this objective is to improve the performability of a disk array subsystem without increasing the cost of the system significantly. This is done in order to minimize the hardware investments while the performance and the data availability are maximized. Typically, a user is unwilling to invest in unnecessary equipment, but, on the other hand, the system reliability and the data availability should be as high as possible. Also, when the high performance requirement is included, optimization becomes important.

The optimization problem generally has two alternative approaches. First, the optimization concerns only the reliability and performance factors (or as combined performability). This can be interpreted as "the best combined reliability and performance at any cost". Usually, it is not feasible to optimize only either the performance or the reliability as the other would suffer too much. The second alternative is to optimize the performability and the system cost together (expressed as cost-performability). This resembles a more practical situation where, due to economical reasons, it not possible to add as much redundancy as desired.

Different users have various desired values for reliability, performance, and cost. Hence, it should be possible to express performability or cost-performability using common equations for a given array configuration as a function of these three parameters.

One of the main factors that limits the system reliability (and therefore also performability and cost-performability) is latent faults. Typically, a disk array can survive at most one fault in a disk group at any time. As a fault may remain undetected in a disk array for a long time, the importance of latent fault detection increases significantly in order to maintain reliability.

The reliability of a conventional disk array is typically estimated to be of the order of millions of hours as expressed with the mean time to data loss (MTTDL) like stated in [Gibson 1991]. These figures consider only disk unit faults ignoring both sector faults and latent faults in general. If these faults are included, the reliability drops significantly [Kari 1994, Kari 1993, Kari 1993a, Kari 1993b]. Hence, there should be a mechanism to detect those faults in order to regain the reliability. However, the reliability will very likely have an upper bound of the conventional estimations (with only disk unit faults considered).

3.1 Performability and cost-performability

For disk subsystems, the performability and cost-performability terms can be used for expressing the quality of the architecture [Trivedi 1994, Catania 1993, Pattipati 1993, Smith 1988, Furchgott 1984, Meyer 1980, Beaudry 1978]. Performability can be used for analyzing the behavior of a disk subsystem and finding the optimum combination of performance and reliability as a function of the number of disks in the array as shown in Figure 2.[*] Here, a hypothetical system is measured with two parameters: performance (as measured with the number of I/O operations per second) and reliability (as expressed with MTTDL). The performance of a disk array improves with the number of disks as the system can serve more simultaneous disk requests. On the other hand, the reliability decreases with the number of disks as there is a higher number of parallel disks that can become faulty. The optimum performability can be defined as a function of performance and reliability.

In cost-performability, the cost (as expressed with the cost of the system installation, running costs, and possible damages due to data loss) of the system is taken into consideration. In Figure 3, the cost-performability of the same system is illustrated. As the cost increases with the number of parallel disks, the optimum cost-performability point is not necessarily at the same location as the optimum performability point. Cost-performability is not discussed further in this thesis since the term “cost” has no unambiguous definition like performance and reliability.

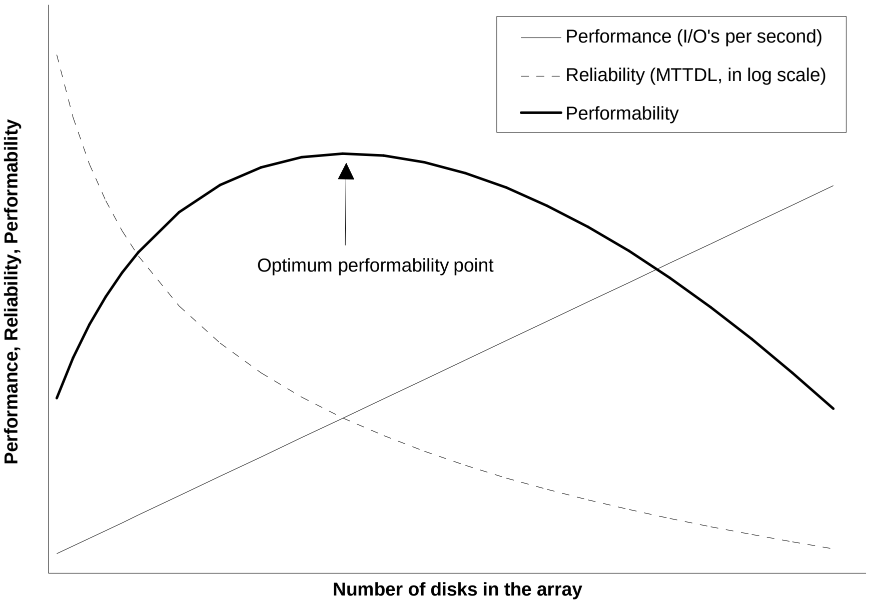

Figure 2. Principle of optimizing combined performance and reliability

[*] A detailed model for performability is discussed in Chapter 9.

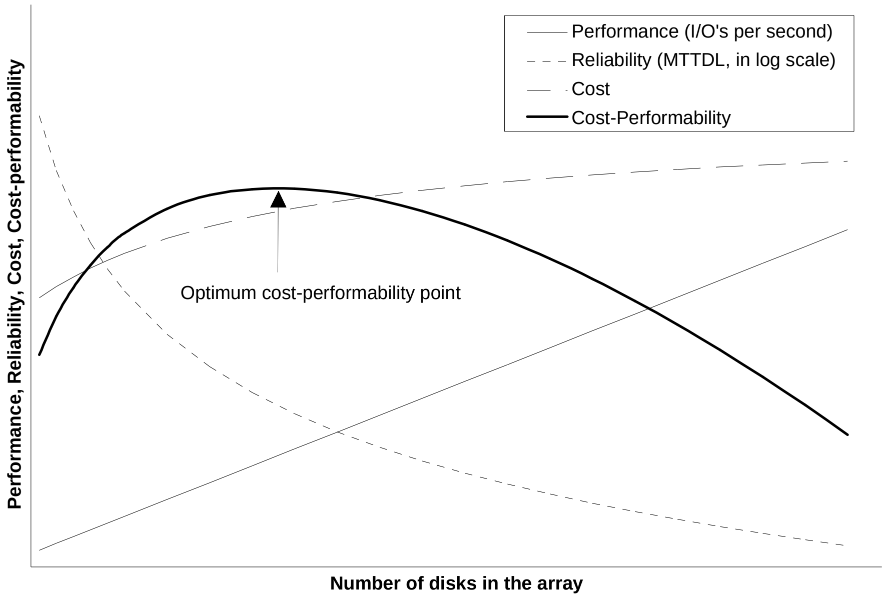

Figure 3. Principle of optimizing combined performance, reliability, and cost

3.2 Performance and reliability development

The performance and reliability of computer systems have improved radically in recent years. It has been compared that if the development in the automobile industry had been as rapid as in the computer industry, the current cars would cost only one dollar and they would run one million miles per gallon. However, the reliability of an average car is typically much higher than that of a conventional computer (especially when the software problems are included).

Performance development

Unfortunately, performance developments have not been as rapid for all components of a computer system. The slowest development in the performance area has been amongst I/O subsystems such as network, mass storage, and user I/O. These systems have been overrun by the rapid progress of the central processing unit (CPU). For example, the network transfer rate has improved by ten fold in last five years (e.g., from 10 Mbps to 100 Mbps). The user I/O rate has increased only for output devices, but most of the performance gain has been drained by the higher display resolution. On the contrary, the user input rate has not improved significantly since the early days of computing. The capacity of hard disks has increased significantly in recent years, but otherwise the hard disk performance has shown only slight improvement. In Table 4, typical top of the line personal computers of 1986 and 1996 are compared [Fujitsu 1996, Intel 1996, Intel 1996a, Nokia 1986].

Table 4. Comparison of a typical PC in 1986 and 1996

	PC in 1986	PC in 1996
Processor	Intel 80286, 8 MHz	Intel Pentium Pro, 200 MHz
• iCOMP 2.0 index [Intel 1996]	1.9[*]	220
Display	VGA, 640x480, 16 colors	Super VGA, 1600x1280, 16 million colors
Memory	640 kB, 16 bit wide	64 MB, 64 bit wide
Hard disk	ST506, 20 MB	SCSI/IDE, 4 GB
• average seek time	100 ms	10 ms
• average rotation delay	8.3 ms	4.2 ms
• sustained transfer rate of disk	0.5 MB/s	5 MB/s
• average reliability (MTBF)	20 000 h	500 000 h
Disk bus speed	2 MB/s	20 MB/s
Network speed	500 kb/s	100 Mb/s
Size of a normal word processing program	500 kB	4 MB
Average street price	100 000 FIM	25 000 FIM

Figure 4 illustrates how the hard disk capacity has developed for non-removable hard disks as a function of time and the size of the disk [IBM 1996a]. The capacity of the hard disks has increased steadily, about 40% per year, while the form factor (i.e., the physical size of the disk) has reduced significantly.

The performance of hard disks has not improved as fast as their capacity. The performance improvement has been restricted by the physical limitations (such as rotation speed, seek delays, storage density, and head mass).

Three parameters are used for measuring and expressing the performance of a hard disk: *rotation speed, seek time,* and *transfer rate*.

The enhancements of combined seek and rotation delays of a hard disk are illustrated in Figure 5 [IBM 1996b]. In mid 1980’s, the average seek time was still in order of 100 milliseconds, but, due to smaller disk form factors and lighter materials, the average seek time is nowadays less than 10 milliseconds. This means that the seek delay is now only 10% of that of ten years ago. The rotation speed has increased in last ten years from about 1800 rpm up to 7200 rpm. Thus, the reduction of rotational latency is 75% over the same time span.

[*] Intel prefers to use iCOMP in measuring the performance of its new CPUs instead of commonly used SPECint and SPECfloat benchmarks. Intel does not measure iCOMP index for old processors (such as 80286). This is estimated using Pentium 75MHz as reference (iCOMP 2.0 value is 67) and CPUBENCH figures for 75 MHz Pentium (4805 units) and 8 MHz 80286 (139 units).

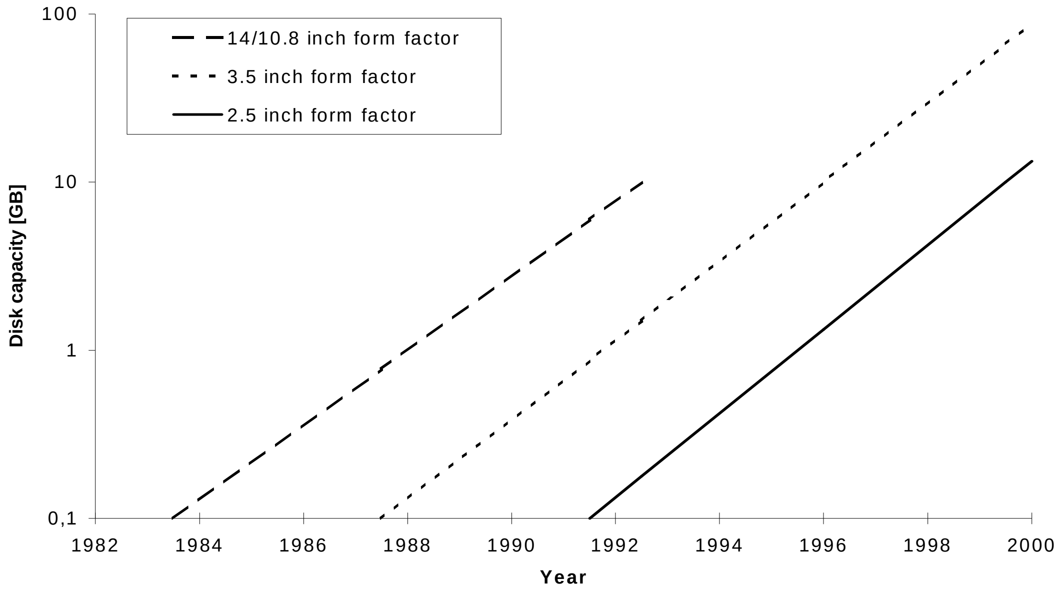

Figure 4. Development trends of the disk capacity and size

The third parameter of a hard disk is the data transfer rate. The data transfer rate is limited by two factors: internal transfer rate in a disk and disk bus transfer rate. From late 1980’s through mid 1990’s, the internal data rate of disk drives has increased about 40% per year [IBM 1996c]. The current internal data rate is about 10 MB/s while the external data rate depends on the disk bus type varying from 10 to 40 or up to 100 MB/s [Seagate 1996a, Seagate 1996b]. Modern hard disks can

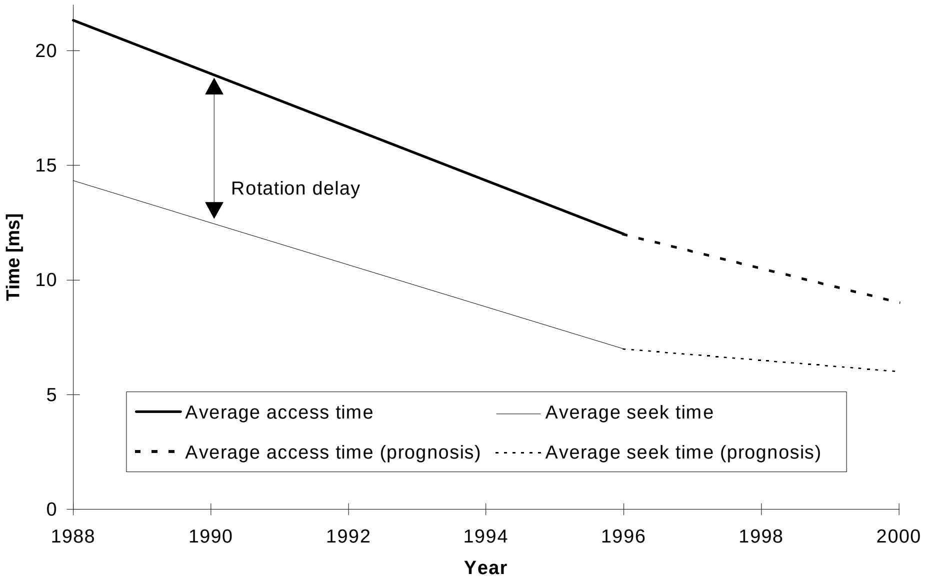

Figure 5. Progress of the seek and rotation delays in recent years

utilize significantly higher bus transfer rates by buffering the data and disconnecting themselves from the bus when they are performing internal disk I/O operations. Hence, several hard disks can be connected onto the same bus sharing the high speed transfer channel.

Reliability development

The reliability of the hard disks has been enhanced significantly in the last ten years. In the mid 1980’s the average MTBF for a hard disk was in order of 20-40 000 hours while the current MTBF figures are around 500 000 to 1 million hours [Seagate 1996c, Quantum 1996a, Hillo 1993, Nilsson 1993, Faulkner 1991, Gibson 1991]. The main reasons for this are the improved disk technology, reduced size of the disks and new methods to predict the MTBF figures (based on field returns). One million hours (about 100 years) for MTBF of a disk is a quite theoretical figure. The actual figure greatly depends on the usage of the disks. For example, a set of 50 heavily loaded disks had 13 faults in three months leading to less than 10 000 hours MTBF while the “official” MTBF for these drives was around 300 000 hours [Hillo 1996, Räsänen 1996, Hillo 1994, Räsänen 1994].

3.3 Economical effects

The improved reliability and data availability have no value by themselves. On the contrary, most of the users are price-conscious and will not like to invest in unnecessary pieces of equipment unless there is a real benefit in the investment.

Eventually, the question of performability is money and risk management for the desired performance and reliability of the system. As a disk array is generally purchased in the first place to protect valuable user data and preferably to provide nonstop operation, the cost of a data loss can be assumed to be high. Thus, the probability of data loss should be minimized but not at any price. It is not wise to increase the level of complexity too high in the system because the practical reliability may be only a fraction of the theoretical estimation. The practical reliability is decreased, for example, by improper human operation and software errors (caused by too complex system software).

The financial effects of the disk array concept are two fold. First, the initial cost of the disk array subsystem is significantly higher as more hard disks are needed and the disk controller (and its software) is more complex. Second, the probability of data loss is smaller and therefore the expected damage due to a data loss is significantly less than in a non-fault tolerant disk subsystem.

At a certain point, there is no need to improve the data availability in the disk array level as other components are relatively less reliable than the disk array.

3.3.1 Two views on reliability

There are two points of view to computer reliability: user's and manufacturer's.

User’s view to reliability

From the user's point of view, the system is or is not operable. Therefore, there is only marginal (if any) benefit of improving MTTDL value of a system, for example, from 1 million hours to 10 million hours. This is because the user is only observing (typically) one disk array and none of the normal computer systems are designed to operate for a so long period of time (100 to 1000 years). Most of the computers become obsolete in a few years and will be replaced with a new model before the reliability figures have decreased even a bit. Hence, the reliability issues, when inspecting only one machine, lose their significance as the availability of the system remains high (in many practical cases being almost one) over the entire useful lifetime of the system.

Manufacturer’s view on reliability

On the contrary, a manufacturer sees a completely different view of reliability. For example, if there are 100 000 installations in the field each containing one disk array subsystem, there is a significant difference in user complaints if MTTDL increases from one to ten million hours. In the former case, there will be about 880 cases of data loss per year (systems are assumed to run 24 hours per day) but, in the latter case, only about 88 cases. Hence, this may have a dramatic effect on the profit and reputation of a company.

3.4 Benefits of improved performability

There are benefits for both good performance and reliability (as well as low cost). However, the combined performability is a compromise for both. The benefits can be divided into three categories: improved data reliability (or data availability), improved performance, and reduced cost to operate the system (fewer data losses).

Improved data availability

Enhancements in the reliability of a disk array improve data availability as well as nonstop operation of the system. A good example for a system that can benefit from the improved data availability is a database server that supports OLTP. In such a system, continuous operation is important and the cost of a data loss is typically extremely high.

Improved performance

The improved performance (especially during the degraded state) is a valuable property for systems that must provide good performance at all time. A disk array should be usable even during the repair process and there should be no need to shut down the system or disable user request processing while the repair process is active. In this way, the system can provide nonstop service for users even during exceptional situations.

Cost optimization

The final benefit of improved performability is the reduced cost to run the system. The user will experience higher reliability and/or better performance of the disk array for the same money when the total life span and all possible costs of the system are considered. Alternatively, the same reliability and/or performance can be achieved with reduced cost.

3.5 Methods to improve performability

Performance improvement

This is a widely studied aspect of the disk arrays and there are a large variety of reports presenting how to improve performance in disk arrays [Hou 1994, Burkhard 1993, Holland 1993, Mourad 1993a, Reddy 1991, Muntz 1990]. Hence, this subject is not studied further in this thesis.

Reliability improvement

Data availability can be improved in three ways: using more reliable components, using higher levels of redundancy, or expediting the repair process. It is difficult to improve component reliability beyond a certain level. Therefore, it is not possible to improve the data availability by only enhancing the components. Alternatively, better availability can be achieved by utilizing higher levels of redundancy. Unfortunately, this usually also means performance degradation as updating data on a disk array gets slower as the redundancy increases. Thus, data availability can be improved up to a certain level when the lower limit of performance is set. The only remaining method to improve data availability is to expedite the repair process.

Most modern disk array architectures tolerate only one fault in a disk group. Therefore, it is vital for the data availability to minimize the time when the array has a fault in it. By reducing the duration of a fault (i.e., this is done by expediting the fault detection process and/or the fault repair process), the probability of having a second fault in the same disk group can be reduced radically.

It is typically quite difficult to expedite a repair process without effecting the performance. This is because when the time of the repair process is minimized, the utilization of the disks increases causing user disk requests to be delayed significantly. Thus, performance requirements limit the usage of the speed-up of the repair process in improving the reliability.

The remaining method to improve the reliability is therefore to reduce the time when faults are present in the system. As the repair process is hard to expedite, the interest should be focused on detecting existing faults thus eliminating the faults as quickly as possible. The fault detection can be done either by conventional user disk requests or by a special diagnostic procedure. The former case has the problem that it is related to user access patterns and therefore does not provide full coverage as not all areas of the disk are accessed by user requests. Hence, an active scanning program is needed to detect faults also in the rarely accessed areas.

The active scanning program inserts disk scanning requests among user disk requests thus increasing delays in user disk requests. If the parameters are set properly, the performance degradation will be reasonable. However, even a slight increase in the load in a congested system can lead into significant delays.

The best option would be if the system would detect faults even before their occurrence. This can be done using the increased number of retries as early warning signs of degradation [ANSI 1994, Räsänen 1994].

4. MODERN DISKS AND DISK ARRAYS

This chapter consists of two parts: a quick overview of the properties of modern SCSI disks and a review of disk array configurations.

4.1 SCSI disk properties

The two most commonly used disk types in the modern computers are based on IDE (*Intelligent Drive Electronics*) and SCSI (*Small Computer System Interface*) interfaces. In many cases, disk manufacturers offer similar disks with both disk interfaces. The main difference between these two interfaces is the larger variety of functions in the SCSI interface. In this thesis, the interest is focused on SCSI disks as they are widely used in disk arrays.

There are currently two SCSI standards: SCSI-1 and SCSI-2 [ANSI 1994, ANSI 1986]. The third generation of the SCSI standard (SCSI-3) is under development [ANSI 1997, T10 1997, ANSI 1996, ANSI 1995]. These three standards specify the interfaces not only for disks but also for other devices (such as CD ROMs, tape streamers, printers, and local area networks).

The SCSI commands are divided into two categories: mandatory and optional commands. The mandatory commands are required to be recognized by all SCSI devices while a manufacturer may or may not implement the optional commands. Some of the commands are device specific thus used only with certain devices or they behave differently with different devices. In addition, there are some vendor specific SCSI commands or fields in the SCSI commands [ANSI 1994, Seagate 1992a]. For example, statistical information can be obtained from a disk with a standard command, but the information is manufacturer specific.

Significant part of the thesis is attributed to the enhanced properties of modern SCSI disk standards. As a normal SCSI disk by itself keeps track of its operation and logs events during its normal operations, it is possible to implement the scanning algorithms that are discussed in this thesis.

4.1.1 Logical data representation in SCSI disks

The main operating principles of disks can be found in [ANSI 1996, Seagate 1996a, Seagate 1996b, ANSI 1995, ANSI 1994, Seagate 1992, Seagate 1992a, Conner 1992, Sierra 1990, ANSI 1986, Cox 1986]. In a disk that complies with the SCSI-2 standard, the disk storage is represented as a continuous set of blocks, usually sectors [ANSI 1994, Seagate 1992a]. This is different from some older disk standards where a disk was categorized with three parameters: number of heads, number of tracks, and number of sectors per track. In a SCSI disk, the actual structure of the disk is hidden from normal users, but there is a SCSI command to query detailed information about physical disk geometry.

There are two major benefits of having a linear storage architecture. First, it hides the physical structure of the disk simplifying the handling of the disk by the operating system. Second, the linear data representation allows the operating system or the disk to repair sector faults without modifying the logical representation.

The major disadvantage of the linear storage architecture is found with the performance optimization. As the operating system does not generally know the physical structure of a disk, it is not able to adjust its disk requests based on the mechanical limitations. For example, read-ahead algorithm may suffer additional head switching and seek delays if the last requested sectors fall into a different track or surface.

4.1.2 Sector repair process

A SCSI disk tries to minimize the risk of media deterioration. During a low level format operation, the disk scans the surface and ignores those areas that have been diagnosed to be faulty [ANSI 1994, Seagate 1992a]. The actual procedure is vendor specific. For example, not only the actual faulty area is omitted, but also its nearby regions can be rejected in order to minimize the probability of encountering media deterioration in the future.

A SCSI disk may also encounter defective blocks after a low level format operation. This can happen, for example, during a read or a write operation or while performing a diagnostic operation. In such an event, the defective block can be recovered with a special SCSI command: *REASSIGN BLOCK* [ANSI 1997, Kamunen 1996, Räsänen 1996, ANSI 1994, Räsänen 1994, Platt 1992, Seagate 1992a, ANSI 1986]. This command replaces the specified defected area (i.e., sector) with a spare one while maintaining the same logical representation. Again, the actual repair procedure is vendor specific.

4.1.3 Advanced features

Modern SCSI disks gather various statistical information during their normal operation. This information is divided into two categories: counters and detailed statistics. For example, the following kind of information is provided in *ERROR COUNTER PAGES* information elements of SCSI disks [ANSI 1994, Seagate 1992a]:

- Errors corrected without substantial delay
- Errors corrected with possible delays

- Total errors (e.g., rewrites or rereads)
- Total errors corrected
- Total times correction algorithm processed
- Total bytes processed
- Total uncorrected errors

This information is typically used for observing the behavior of the disk and its operational quality. The error counters can be used for early warning signs of coming disk faults.

Most recent error information

There is also more detailed information on the *LAST N ERROR EVENTS PAGE* that is vendor specific [ANSI 1994, Seagate92a]. This information can be used for obtaining more detailed information on the most recent errors. The number of those reported errors depends on the manufacturer.

Masking effect

A disadvantage of the *LAST N ERROR EVENTS PAGE* function is the masking effect. As the disk has limited capacity to store detailed error information, some of the older errors may be masked by new ones. In practice, the masking effect does not significantly reduce the fault detection capability. Typically, the frequency of reading this information is much higher than what it is updated. The information can be read in every few seconds while there are normally only a few major errors in a month or a year [ANSI 1994, Räsänen 1994, Gibson 1991].* Hence, the system can detect most of the errors reported with LAST N ERROR EVENTS PAGE without any major risk of missing error reports.

Automatic error reporting and recovery

The second method to get more information about the internal problems is to configure the disk to report all internal errors and recoveries. For example, it is possible to define, that the disk should report if read operation was successful, but [ANSI 1994, Räsänen 1994, Seagate 1992a]:

- error correction was needed,
- negative/positive head offset was needed,
- reassignment is recommended, or

* If the fault rate of a disk increases and a large number of faulty sectors is detected in a short time, the disk is considered to have reached the end of its useful lifetime and it should be replaced.

- rewrite is recommended.

It is also possible to configure the SCSI disk to perform recovery of the possibly defective area given the slightest sign of a problem (by setting *ENABLE EARLY RECOVERY* mode). This informs the disk to use the most expedited form of error recovery at the expense of higher risk of error mis-detection and mis-correction (i.e., the disk may consider that the sector is faulty even when it is actually not).

Media verification

The third alternative that is useful especially with a scanning algorithm is to use the *VERIFY* command [Scritsmier 1996, ANSI 1994, Seagate 1992a]. This command reads the information from the medium, but the data is not transferred from the disk (i.e., the data is read from the disk surface into the internal buffers, the consistency of the data is checked and errors reported). This can speed up the latent fault detection as the disk controller is not loaded with the scanning data as the disk performs most of the work internally.

4.2 Modern disk arrays

The disk subsystem architectures that are used as examples in this thesis are based on the practical implementation of the RAID-concept such as reported in [Hillo 1993, Kemppainen 1991]. Figure 6 depicts a sample configuration of such an approach.[*] This approach is a typical *hardware disk array* (HDA) where a disk controller implements the RAID algorithms and, from the point of view of an operating system, it resembles a logical disk.

A HDA is considered to be superior to a *software disk array* (SDA) [Kamunen 1994, Räsänen 1994, Hillo 1993]. In a SDA, there are three major disadvantages. First, the SDA depends strongly on the operating system (i.e., the array software is implemented as a part of the main operating system hence requiring a special driver for every new operating system). Second, the SDA is inferior to the HDA in handling faults in the array (e.g., booting from the SDA with one disk failed may be impossible). Besides, the SDA suffers more from higher interrupt load caused by disk I/O, especially in more complex arrays such as RAID-5. Typically, a general purpose operating system is not as efficient in handling a large number of interrupts as a dedicated real-time operating system on

[*] The configuration of the disk array that is used here as an example describes a medium size disk array that is typically used in mid-range computers (such as PC-class servers). The reliability analysis in this thesis is not, however, limited only to such systems, but the analysis is valid for any number of disks.

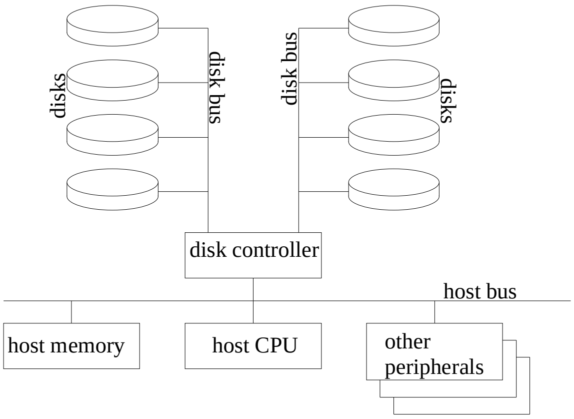

Figure 6. An example of a disk array configuration

a controller board [Räsänen 1994, Kamunen 1994].

Most of the disk array controllers are based on SCSI buses and SCSI disks. The transfer rate of a SCSI bus is sufficient to serve a large set of disks, especially in an OLTP environment where the average size of disk requests is typically small [IBM 1996c, Seagate 1996a, Ylinen 1994, Hillo 1993, Kari 1992, Miller 1991]. The number of disk buses is increased mainly to allow larger disk array configurations and higher number of parallel I/O's, not to increase the data transfer capacity.

The main application environments that have been kept in mind while doing this research are database and file servers. First, the database servers are typical examples of systems with nonstop operation and strict response time requirements that must be met even during a degraded state. Second, both database and file servers have similar access patterns where disk accesses are spread unevenly over the disk space.

4.2.1 Hierarchical fault tolerant architecture

A hierarchical fault tolerant architecture supports several parallel disk array controllers in one or several computers. This provides a three-level fault tolerant architecture as illustrated in Figure 7 [Novell 1997]. The lowest level of fault tolerance is the devices, disks. The second level consists of mirrored disk controllers.* Finally, the third level fault tolerance consists of mirroring the entire servers. For example, dual disk controllers can be used for surviving controller faults or mirrored computer systems can be used for surviving faults in any other critical part.

In this research, the focus is on the device level fault tolerance (level I). Hence, the controller

* Typically, the second and third level fault tolerant architectures are based on mirroring while the first level can use various array configurations.

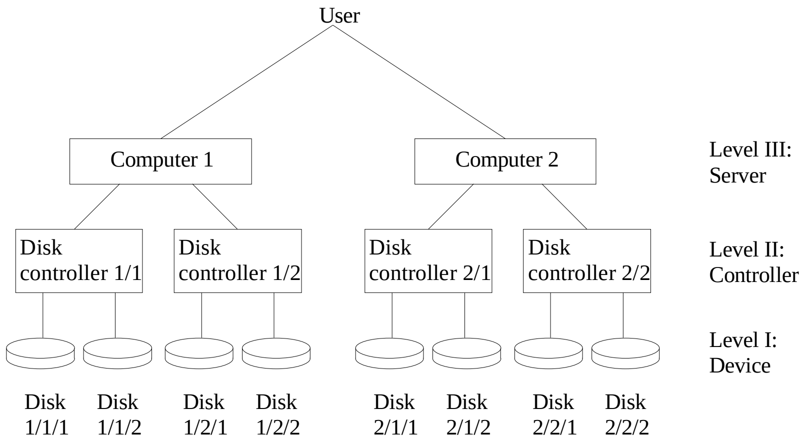

Figure 7. Three level hierarchical fault tolerant architecture

mirroring (level II) and server mirroring (level III) techniques are not discussed further in this thesis.

4.2.2 Disk array architectures

Six main RAID architectures are already listed in Chapter 1. Here, those architectures are briefly described. Beside those "standard" RAID architectures (RAID-0, RAID-1, RAID-2, RAID-3, RAID-4, and RAID-5), there are several RAID variants, but they are out of the scope of this thesis. It is not an intention here to give a complete description of all RAID architectures but just to have a short overview of the main principles. More detailed description of various RAID architectures can be found in [DPT 1993, RAB 1993, Lee 1991, Lee 1990, Katz 1989, Chen 1988, Patterson 1988, Salem 1986]

Single disk

A single disk (sometimes denoted as RAID-0/1, striped array with only one disk) is often used in the models for comparison purposes. Some disk array controllers also support single disks for compatibility reasons [Hillo 1993, Kemppainen 1991].

RAID-0

Disk striping (RAID-0) provides no additional reliability improvement, but it is very often used in systems where reliability is not so vital (e.g., recovery after a disk failure can be done using backups and log files). Reliability of a disk subsystem decreases dramatically as the number of disks increases [Hillo 1993, Gibson 1991].

Figure 8 illustrates the basic principle of the RAID-0 array. It has two main parameters: the

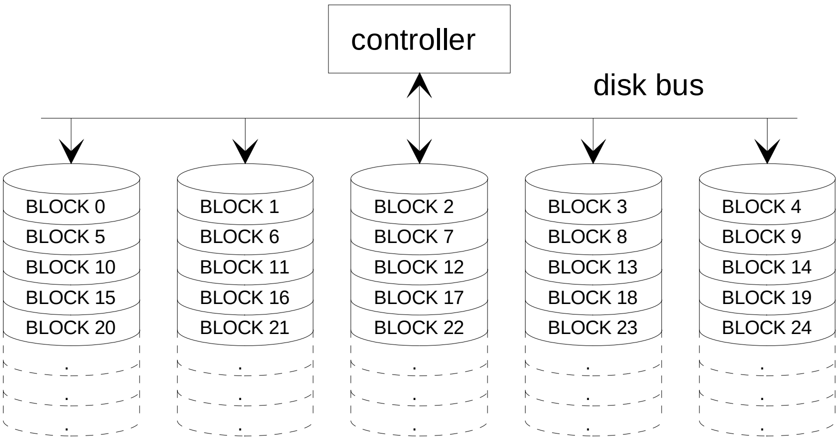

Figure 8. RAID-0 array with five disks

number of disks (*D*) and the size of the stripe unit*. Depending on the size of a request, one or several disks are accessed. The optimum size of the stripe unit depends on the user disk requests, the average disk access time and the data transfer rate [Hillo 1993, Chen 1990a].

RAID-1

Disk mirroring (RAID-1) is a traditional method of achieving fault tolerance in a disk subsystem as described for example in [RAB 1993, Gibson 1991]. Due to two identical copies of the data, the disk mirroring algorithm suffers from high cost of redundancy (i.e., 50% of the disk space is spent for redundancy). One of the benefits of the mirroring concept is found in its way of handling write requests. Mirroring has significantly smaller overhead in write operations than, for example, the RAID-5 array. In a sustained write load, the RAID-1 array performs as well as a single disk [Hillo 1993, Kemppainen 1991]. The RAID-1 array has also twice as high bandwidth for read operations as a single disk. In Figure 9, the basic principle of the RAID-1 array is illustrated.

It is possible to combine the mirrored and the striped array architectures (i.e., RAID-0xRAID-1). The main idea is to have a RAID-0 array mirrored with an identical one. This allows achieving similar capacities (e.g., 2x50 disks) with the RAID-1 array as with the other RAID arrays.

RAID-2

RAID-2 arrays are designed for environments where high data transfer rates are required. As illustrated in Figure 10, the data is striped across multiple disks while some of the drives are

* The *stripe unit* is defined to be the unit of data interleaving, i.e., the amount of contiguous data (the size of BLOCKx) that is placed on each disk. Depending on the disk array configuration, the stripe unit can be bit, byte, sector, or several sectors.

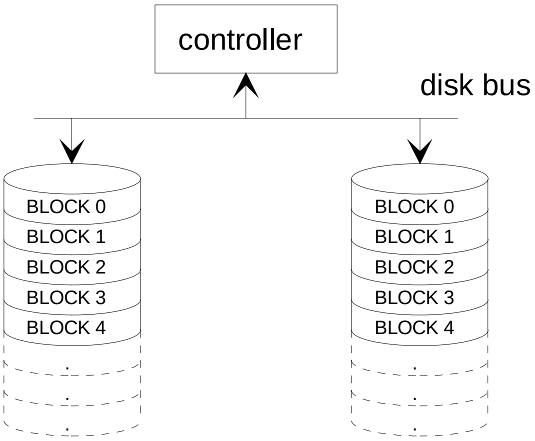

Figure 9. RAID-1 array with two disks

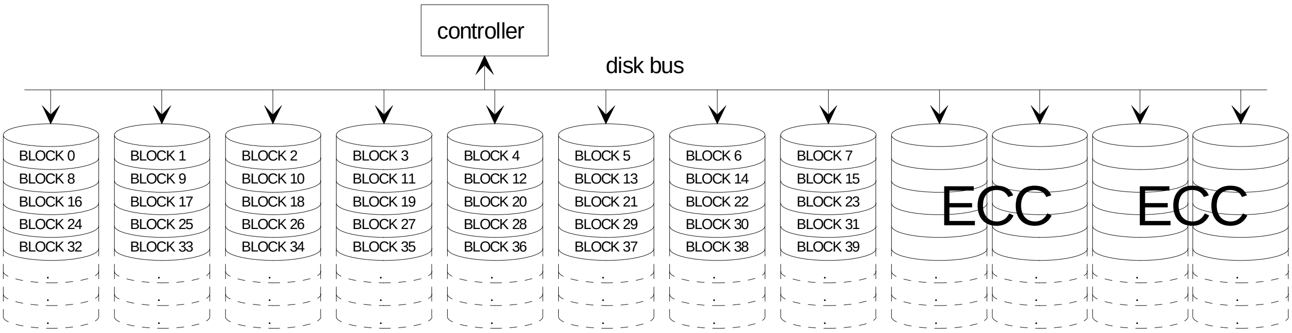

Figure 10. RAID-2 array with eight data disks and four parity disks

dedicated to store additional ECC information that is calculated over the data disks.

RAID-3

As with RAID-2, the data is striped across multiple disks in the RAID-3 array. In this case, only one parity disk is used as illustrated in Figure 11. The size of the stripe unit can be either one bit or one byte. The error detection in the RAID-3 relies on the ECC embedded in each of the disks.*

The parity is computed horizontally over the recorded bits BIT(n) … BIT(n+4) [RAB 1993]. For example, for the first parity row, PARITY0 is calculated as

$$PARITY0 = BIT0 \otimes BIT1 \otimes BIT2 \otimes BIT3 \otimes BIT4 \qquad (1)$$

* When a data block is read from a disk, the disk checks the internal ECC and reports an error if it does not match. Hence, the upper layer (e.g., the controller) can use information in the other data disks and the parity disk to reconstruct the missing information. The

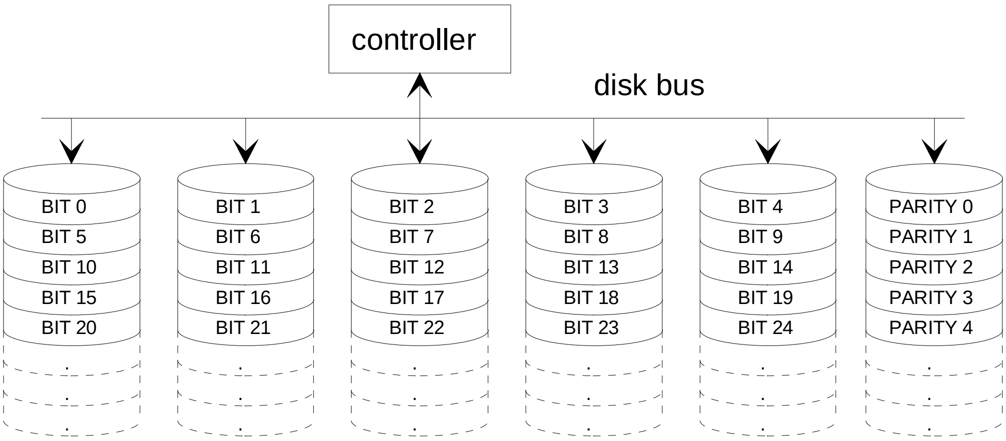

Figure 11. RAID-3 array with five data disks and one parity disk

where $\otimes$ indicates exclusive-or function. If, for example, the second data disk (storing BIT1) is faulty, the data is recovered as follows

$$BIT1 = BIT0 \otimes BIT2 \otimes BIT3 \otimes BIT4 \otimes PARITY0\,. \tag{2}$$

The same principles are applicable also for byte oriented RAID-3 as well as with RAID-4 and RAID-5 arrays.

This example identifies also the main problem of the RAID-3, RAID-4, and RAID-5 arrays: If there are more than one missing bit/byte/block in a row, the algorithm is no longer capable of reconstructing the data.

RAID-4

The RAID-4 array uses the same principle as the RAID-3 array with the exception that, instead of bit or byte as the stripe unit, the RAID-4 array uses a larger stripe unit size (i.e., typically multiple of sectors). This allows the controller to issue several simultaneous read operations and one write operation to the RAID-4 array. Figure 12 illustrates the RAID-4 array configuration.

RAID-5

The main bottleneck of the RAID-4 array is the parity disk as all write requests must update the same disk. This problem has been solved in the RAID-5 array where the parity is distributed over all disks evenly as shown in Figure 13.

Both RAID-4 and RAID-5 arrays can optimize the write operations. According to the RAID-4

same principle is also used in RAID-4 and RAID-5 arrays.

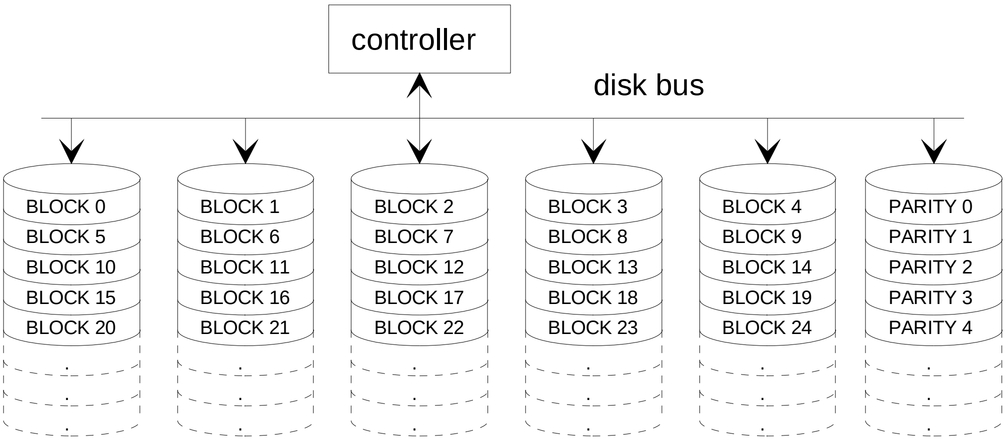

Figure 12. RAID-4 array with five data disks and one parity disk

and RAID-5 definitions, the parity information is calculated over all disks of the disk group as shown in Equation (1). If only one block is modified, this would require reading data from all other disks and then calculating the new parity. Instead, the parity can be calculated using old data, old parity and new data as the following example illustrates [Hillo 1993, RAB 1993]. If BLOCK2 is to be updated, new PARITY0 can be calculated by first reading old BLOCK2 and old PARITY0, then calculating

$$PARITY0_{new} = BLOCK2_{new} \otimes BLOCK2_{old} \otimes PARITY_{old} \tag{3}$$

and finally writing new BLOCK2 and new PARITY0 into the disks. Hence, a user write operation

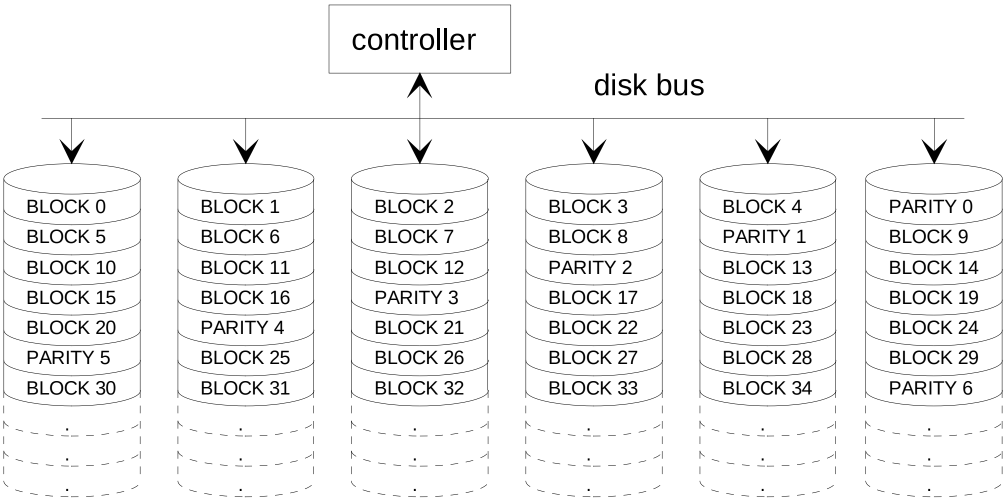

Figure 13. RAID-5 array with six combined data and parity disks

can very often be reduced to two disk read and two disk write operations for RAID-4 and RAID-5 arrays. An additional benefit is that the remaining disks can serve other disk requests at the same time.

5. DISK SCANNING ALGORITHMS

Disk scanning algorithms that are discussed in this thesis are based on the same basic principle as the memory scrubbing algorithm [Saleh 1990]. The memory scrubbing algorithm scans primary memory during the idle time of processor in order to find memory defects. Similarly, the disk scanning algorithms use the idle time of disks in detecting latent sector faults as reported in [Kari 1994, Kari 1993, Kari 1993a, Kari 1993b].

The disk scanning algorithms differ significantly from the memory scrubbing algorithm. For the memory scrubbing algorithm, time to access any memory location is almost constant while a disk access depends on the previous disk location. Especially with high data locality (where the probability of accessing nearby the previous request is high), the scanning requests tend to move the heads away from their previous location causing increased seek times.

In this chapter, four different scanning algorithms are presented. The first two algorithms were published in [Kari 1994, Kari 1993, Kari 1993a, Kari 1993b] while the two later ones are only presented here.

5.1 Original disk scanning algorithm

The original scanning algorithm was presented in [Kari 1993]. In this algorithm, the idle time of the system is used. As the disk driver is typically interrupt driven and no polling is allowed, the scanning algorithm is a timer based process that checks whether a disk is idle. If the disk is already in use, no action is taken. If the disk is idle, then the algorithm issues a read request to the disk.

This method reduces significantly the longest delays that any user disk request may experience. The maximum time that a user disk request needs to wait (beside the other user requests) is one scanning request. This is because the scanning algorithm waits for an idle slot thus each subsequent user request will in the worst case have one scanning request ahead of it. Thus, it is not possible to have two scanning requests on a disk as the other request would have been established while the previous one was still pending and this would violate the idle test of the algorithm.

The scanning algorithm utilizes the advanced statistics of a modern SCSI disk as described in the previous chapter [ANSI 1994, Seagate 1992a]. The parameters of the scanning algorithms are fixed and they depend on the estimated maximum load of the system.

The original (non-adaptive) scanning algorithm is as follows:

Algorithm 1:

1. Reset initial parameters (waiting time *wt*, request size *rs*, start address *sa*, and size of check region *cr*).
2. Reset the start address offset, *sao=0*.
3. Wait for a fixed amount of time *wt*.
4. If there is at least one user request waiting for the disk (in process or waiting in a queue), go to step 3.
5. Issue a disk read request at location *sa+sao* with size *rs*.
6. When the read request has been completed, increment *sao* by *rs*.
7. If *sao < cr*, go to step 3.
8. Read the extended disk statistical information.
9. If no faults were encountered, go to step 13.
10. Test the potentially faulty address (*pfa*) that was reported by the statistics.
11. If the sector in *pfa* is faulty (or the test reads cause more retries), start the REASSIGN BLOCK command to repair that sector.
12. If *pfa<sa* or *pfa>(sa+sao)*, go to step 2.
13. Increment *sa* by *sao*,
14. If the entire disk is not yet scanned, go to step 2.
15. Reset the start address *sa=0* and go to step 2.

The main benefit of Algorithm 1 is that it needs to access only once a deteriorated sector to locate it while scanning a large number of sectors. However, there are three minor disadvantages. First, the advanced statistics are available only in SCSI-2 (and SCSI-3) disks and it is partly vendor specific [ANSI 1994]. Second, the masking effect may hide some errors as described in the previous chapter. Third, the early warning signs of media deterioration can cause an extensive usage of spare sectors unless a thorough test is performed to check each potentially faulty sector. However, it is still a better alternative to replace entire disk (without data loss) than to lose critical data due to latent sector faults.

The time to scan the entire disk depends strongly on the wait time and the system load. For example, if the constant load in the array is 80%, the wait time is 10 seconds, and the scanning request size is 64 kB, then the time to scan a 1 GB hard disk is

$$T_{scan} = \frac{ds \times wt}{rs \times (1-\rho)} = \frac{1GB \times 10s}{64kB \times (1-0.8)} = 227 hours \approx 9.5 days \qquad (4)$$

where ds is the disk size and ρ is the disk load. If the load in the array is not constant (e.g., the array is used 12 hours per day with 80% load and rest of the day it is totally idle), the scanning time would be only about 3.2 days.

5.2 Adaptive disk scanning algorithm

The main problem of Algorithm 1 is that it is not able to utilize uneven load patterns very efficiently. Especially, the low load period is underutilized. The situation can be improved by an adaptive algorithm that adjusts its activity based on the current disk load. When the disk load caused by normal user requests goes down, the scanning algorithm can increase its activity. Respectively, when the disk load increases, the scanning algorithm should reduce its activity to ensure that the delays caused by the scanning process are limited. The size of the request block is usually kept constant as it is optimized with respect to the rotation delays.

Beside the scanning process, this algorithm utilizes also a timer-based process. This process checks the disk subsystem activity at regular intervals. Based on the current instantaneous disk activity and the activity history, an estimate of the system activity is calculated. The system activity is then used for determining proper parameters in the scanning algorithm.

The proposed enhanced scanning algorithm is as follows [Kari 1994].

Algorithm 2:

1. Reset the initial parameters (waiting time *wt*, request size *rs*, start address *sa*, and size of check region *cr*).
2. Reset the start address offset, *sao=0*.
3. Wait for an amount of time given by *wt*.
4. If there are no user requests in the disk (in process or waiting in a queue), go to step 7.
5. Call function *adjust(wt)*.
6. Go to step 3.
7. Call function *adjust(wt)*.
8. Issue a disk read request at location *sa+sao* with size *rs*.
9. When the read request has been completed, increment *sao* by *rs*.
10. If *sao* < *cr*, go to step 3.
11. Read the extended disk statistics information.
12. If no faults were encountered, go to step 16.
13. Test the potentially faulty address (*pfa*) that was reported by the statistics.
14. If the sector in *pfa* is faulty (or the test reads cause more retries), start the REASSIGN BLOCK

command to repair that sector.

15. If *pfa<sa* or *pfa>(sa+sao)*, go to step 2.
16. Increment *sa* by *sao*.
17. If the entire disk is not yet scanned, go to step 2.
18. Reset the start address *sa=0* and go to step 2.

The scanning parameters are adjusted using the function *adjust(wt)* and the estimation of the system activity. The system activity is given by

$$A(t_i) = A(t_{i-1})(1-h) + a_i h \tag{5}$$

where $A(t_{i-1})$ is the previous estimation of the system activity, a_i is the instantaneous activity (either 0 or 1) at time i, and h is the history factor. The history factor h specifies how strongly the most recent measurements influence on the estimation. This allows fast adjustment to changes.

Based on the activity estimation, the *adjust* function is given by

$$wt = A(t_i)^2 (wt_{\max} - wt_{\min}) + wt_{\min} \tag{6}$$

where $wt_{\min}$ is the minimum wait time for the scanning algorithm (to limit the maximum system load increase caused by the scanning process) and $wt_{\max}$ is the maximum wait time for the scanning algorithm (to limit the maximum scanning time).

In Equation (6), the quadratic term of the estimated activity is used for compensating the increase of the queue length with higher system load. If a linear function ($wt = A(t_i)(wt_{\max} - wt_{\min}) + wt_{\max}$) were used, the wait time would increase too much even with a moderate disk load (0.25 < ρ < 0.75) so that a significant part of the scanning capacity would be wasted.

Using the parameters of the above example, this scanning algorithm would scan the entire 1 GB hard disk in 12 hours during idle period of night time, if the minimum wait time ($wt_{\min}$) is set to 2.5 seconds. Actually, if the minimum wait time were set to zero, the scanning time would be less than an hour.

5.3 Simplified disk scanning algorithm

The proposed scanning algorithms can be further improved when modifications on the disk device drivers are allowed. Instead of having a timer based operation, the intelligence can be built into the disk device driver. First, the device driver will have two priority queues: one for user requests and one for scanning requests. When there is at least one user disk request in the queue (or in the disk processing), the scanning requests are suspended. When the last user request is completed, a timer is started (e.g., length of one second). If during that time no new user request has arrived, then the scanning queue is resumed and the next scanning request can be issued. When the scanning request completes and if there are still no user disk requests, then the next scanning request can be issued immediately after the previous one.

Waiting a short time after the latest user disk request ensures that if the user issues another disk command right after the previous one was completed, the scanning algorithm does not delay that request. On the other hand, if there have not been any user requests coming for a certain period, it is quite unlikely that there will be a new one in a near future. Especially, the probability of having a new user disk request drops significantly if no user requests have been received for several seconds [Kamunen 1996, Räsänen 1996]. Thus, the scanning algorithm only needs to monitor the time since the completion of the latest user request. This can be a self learning procedure where the scanning algorithm learns the user behavior and adjusts its timer accordingly.

The main concern on this new algorithm is to keep the total load on the disk moderate. As this new algorithm would basically load the disk with 100% load soon after the user disks requests have stopped, there must be a mechanism that slows down the scanning algorithm. The simplest way to do this is to let the scanning algorithm run only once (or a few times) per day. Hence, as soon as the disk is scanned once per given period of time, the scanning algorithm is suspended. The scanning algorithm is again resumed when a new period starts.

It is also possible to simplify the scanning algorithm by setting the SCSI disk parameters properly. Instead of reading the disk statistics every now and then, the disk can be configured to report all faults immediately [Räsänen 1996, Antony 1992]. Then, it is possible to get automatically an indication of all read errors in the disk. Then, both user and scanning disk requests can be used for detecting the latent faults.

5.4 Disk scanning algorithm using VERIFY command

The disk scanning can be expedited even more using the VERIFY command [Räsänen 1996, Scritsmier 1996, ANSI 1994, Conner 1992, Seagate 1992a]. Instead of transferring the data from the disk to the disk controller using a normal READ command, a VERIFY command is used. With this command, the data is read from the disk surface into the internal buffers, the consistency of the data is checked, and errors reported. This reduces even further the extra load caused by the scanning algorithm.

5.5 Performance effect

With the above described enhancements on the scanning algorithms, it is possible to scan the entire hard disk efficiently (in a matter of hours or even minutes) with minimum effect on user requests [Scritsmier 1996]. If the load in the disk array is uneven during the day, the disks can be scanned once (or a few times) per day during the low load periods. Otherwise, scanning can be done in the background with minimum impact on performance. Hence, it can be safely assumed that a disk scanning algorithm efficiently detects latent sector faults with no or only a marginal effect on performance.

6. ASSUMPTIONS FOR RELIABILITY MODELS

The prerequisites for the reliability analysis are discussed in this chapter. Three categories are listed: reliability metrics, reliability measurement, and assumptions made for a novel reliability model.

6.1 Reliability metrics

Two reliability metrics are used in this thesis: data availability and MTTDL. The data availability is used for illustrating the probability of the system of maintaining the data for a given time. This can be used, for example, for expressing the 1-year, 3-years, and 10-years mission success probabilities [Gibson 1991]. The second term, MTTDL, is also used for expressing the quality of the disk array with respect to reliability and availability by expressing the estimated time the system maintains data integrity [Gibson 1991, Pages 1986, Siewiorek 1982, Shooman 1968].

Both data availability and MTTDL are functions of disk array configuration, Mean Time Between Failures (MTBF) or Mean Time To Failure (MTTF) of the disks, and Mean Time To Repair (MTTR) of the disk array. When the mission success probabilities or MTTDL are used, the array configurations can be compared using simple metrics.

6.2 Methods to evaluate reliability

Three alternative methods to evaluate the reliability of a disk array system are: analytical approach, measurements of the existing system, and reliability simulations.

6.2.1 Analytical approach

The analytical approach for the reliability analysis is based on Markov models [Shooman 1968]. With a Markov state transition diagram, it is possible to present and analyze both steady and transient states of a disk array. The steady state analysis is significantly simpler than the transient state analysis as, in the former case, the problem can be solved using the balance equations while, in the latter case, a set of differential equations must be solved.

In a simple Markov model, the group of differential equations can be easily solved using, for example, Laplace transformation, but, as the number of states in the model increases, the inverse Laplace transformation becomes more and more complex. This is shown in the next chapter. Even the complex inverse Laplace transformations can be done numerically, but the problem is to solve the equations in closed form. It may be infeasible in practice to use the numerical approach if a large number of parameter combinations is used, for example, for studying the sensitivity of different parameters. Hence, complex Markov models may require some approximation or simplification in order to be solved analytically.

6.2.1.1 Traditional Markov model

A *traditional Markov model* (TMM) for a disk array reliability is illustrated in Figure 14 [Schwarz 1994, Hillo 1993, Gibson 1991]. As the model has just one fault type and at maximum one simultaneous fault in an array is tolerated, the model has only three states and it can be easily solved analytically. In the model, $p_x(t)$ indicates the probability of the system being in state *x* (where *x* is the number of faulty disks) at time instance *t*. There are totally $D+1$ disks and at least D disks must remain fault-free to maintain data consistency. The system moves from the fault-free state (p_0) to the single fault state (p_1) if any of the $D+1$ disks fails with total rate of $(D+1)\lambda$. The system consistency is lost if a second disk fault occurs with rate $D\lambda$ before the first one is repaired. The failed disk can be any of the remaining D disks. The system returns back from the single fault state to the fault-free state when the faulty disk has been replaced with a spare disk and the data has been reconstructed onto that disk. The repair rate of the faulty disk is μ.

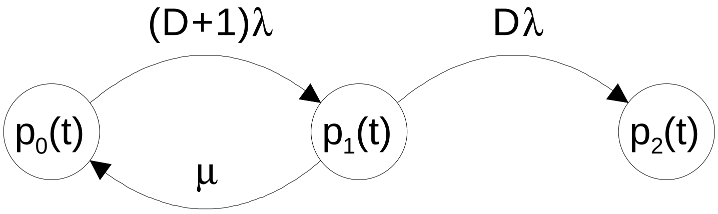

Figure 14. Traditional Markov model for a disk array (D is the number of disks, λ is the disk failure rate, μ is the repair rate, and $p_x(t)$ defines the probability of the system being at state x at time t where x defines the number of faulty disks in the disk array)

No meaningful steady state solution for the Markov model exists in Figure 14 since one sink is included in the model. However, the transition state equations are simple:

$$p_0'(t) = -(D+1)\lambda p_0(t) + \mu p_1(t)\,, \tag{7}$$

$$p_1'(t) = (D+1)\lambda p_0(t) - (\mu + D\lambda)\,p_1(t)\,, \tag{8}$$

and

$$p_2'(t) = D\lambda p_1(t) \tag{9}$$

where the parameters are shown in Figure 14. Also, it is generally assumed for the initial conditions that

$$p_0(0) = 1, \quad p_1(0) = p_2(0) = 0 \tag{10}$$

whereas

$$p_0(t) + p_1(t) + p_2(t) = 1, \quad \forall t. \tag{11}$$

Equations (7)-(9) can be then solved using Laplace transformation and with the help of equations (10) and (11). Hence, the following results will be achieved [Gibson 1991]:

$$p_0(t) = \frac{(D\lambda + \mu + \xi)e^{\xi t} - (D\lambda + \mu + \zeta)e^{\zeta t}}{\xi - \zeta}, \tag{12}$$

$$p_1(t) = \frac{(D+1)\lambda}{\xi - \zeta}(e^{\xi t} - e^{\zeta t}), \tag{13}$$

$$p_2(t) = 1 - p_0(t) - p_1(t) = 1 - \frac{\xi e^{\zeta t} - \zeta e^{\xi t}}{\xi - \zeta}, \tag{14}$$

and

$$R(t) = p_0(t) + p_1(t) = \frac{\xi e^{\zeta t} - \zeta e^{\xi t}}{\xi - \zeta} \tag{15}$$

where

$$\xi = \frac{-((2D+1)\lambda + \mu) + \sqrt{\lambda^2 + \mu^2 + 2(2D+1)\lambda\mu}}{2} \tag{16}$$

and

$$\zeta = \frac{-((2D+1)\lambda + \mu) - \sqrt{\lambda^2 + \mu^2 + 2(2D+1)\lambda\mu}}{2}. \tag{17}$$

The reliability (as measured with MTTDL) can be expressed as

$$MTTDL_{TMM} = \int_0^\infty R(t)dt = \int_0^\infty (p_0(t) + p_1(t))dt = \frac{(2D+1)\lambda + \mu}{D(D+1)\lambda^2}. \tag{18}$$

Similarly, the mission success probabilities are achieved using the following equations:

$$M1 = R(1\,year) = \frac{\xi e^{\zeta 1 year} - \zeta e^{\xi 1 year}}{\xi - \zeta}, \tag{19}$$

$$M3 = R(3\,years) = \frac{\xi e^{\zeta 3 years} - \zeta e^{\xi 3 years}}{\xi - \zeta}, \tag{20}$$

and

$$M10 = R(10\,years) = \frac{\xi e^{\zeta 10 years} - \zeta e^{\xi 10 years}}{\xi - \zeta} \tag{21}$$

where $M1$, $M3$, and $M10$ are the mission success probabilities for one, three, and ten years, respectively.

In the above equations (7)-(21), the following assumptions are valid [Gibson 1991]:

- faults in the disk array are independent,
- lifetime of disks is exponentially distributed,
- repair time is exponentially distributed,
- system tolerates only one fault before repair, and
- there are *D+1* disks in a fault-free system.

6.2.1.2 Approximation of Markov models

Approximation of a Markov model can be done either by reducing the state diagram, simplifying the equations, or using iterative methods [Gibson 1991].

Iterative method

The second method to solve the Markov model is based on the iterative method [Gibson 1991]. In this case, only MTTDL is achieved, but no *R(t)*. The iterative method is as follows:

Beginning in a given state *i*, the expected time until the first transition into a different state *j* can be expressed as

$$E(state\, i\; to\; state\; j) = E(time\, in\; state\, i\; per\; visit) + \sum_{k\neq i} P(transition\; state\, i\; to\; state\, k)E(state\, k\; to\; state\; j) \tag{22}$$

where

$$E(time\, in\; state\, i\; per\; visit) = \frac{1}{\sum rates\, out\; of\;\; state\, i} \tag{23}$$

and

$$P(transition\, state\, i\, to\, state\, k) = \frac{rate\, of\; transition\, to\, state\, k}{\sum rates\, out\; of\, state\, i}. \tag{24}$$

The solution to this system of linear equations includes an expression for the expected time beginning in state *0* and ending on the transition into state *2*, that is for MTTDL. For the Markov model in Figure 14, this system of equations is:

$$E(state\, 0\, to\; state\, 2) = \frac{1}{(D+1)\lambda} + \frac{(D+1)\lambda}{(D+1)\lambda} E(state\, 1\, to\; state\, 2)\,, \tag{25}$$

$$\begin{aligned} E(state\, 1\, to\; state\, 2) = \frac{1}{\mu + D\lambda} + \frac{\mu}{\mu + D\lambda} E(state\, 0\, to\; state\, 2) \\ + \frac{D\lambda}{\mu + D\lambda} E(state\, 2\; to\; state\, 2) \end{aligned}, \tag{26}$$

and

$$E(state\, 2\, to\, state\, 2) = 0\,. \tag{27}$$

Solving the above equations (25)-(27) for MTTDL leads to

$$MTTDL = E(state\, 0\, to\, state\, 2) = \frac{(2D+1)\lambda + \mu}{D(D+1)\lambda^2}\,. \tag{28}$$

Simple approximation

Gibson [Gibson 1991] has also proposed other approximation and simplification methods. These simplifications take advantage of the knowledge that the average disk failure rate is significantly lower than the average repair rate ($\lambda << \mu$). In that case, the system reliability can be approximated with the exponential function

$$R_{Indep}(t) = e^{-t/MTTDL_{Indep}} \tag{29}$$

where

$$MTTDL_{Indep} = \frac{\mu}{D(D+1)\lambda^2} = \frac{MTTF_{disk}{}^2}{D(D+1)\,MTTR_{disk}}, \tag{30}$$

$MTTF_{disk}$ is the mean time to failure of a disk, and $MTTR_{disk}$ is the mean time to repair a disk fault in the disk array.

6.2.2 Reliability measurements

The analytical results of the reliability model are sometimes verified using measurements in a real environment [Shooman 1968]. In many cases, such measurements are not possible as the number of systems may be low and the period of time to observe the system should be very long (MTTDL can be millions of hours) as stated in [Gibson 1991]. It is practically infeasible to wait tens of years to gather statistics while the number of installations is typically small because of the high price. In addition, the practical reliability figures may differ significantly from the analytical models, as reliability of a disk array is also strongly related to the software implementation of the disk array [Hillo 1996, Kamunen 1994, Räsänen 1994, Hillo 1993]. Therefore, it was decided not to use measurements from the existing disk array systems for comparison.

6.2.3 Reliability simulation

A simulation program is an alternative approach to solve a Markov model. For this, several simulation programs are available and some of them have already been used for disk array analysis [Gibson 1991, Lee 1991, Orji 1991, Chen 1990, Lee 1990, Reddy 1990a, Pawlikowski 1990, Comdisco 1989, Sahner 1987, Sahner 1986]. As the transition rates from state to state varies much (several orders of magnitude), the accuracy requirements for the simulation program become significant (e.g., accuracy of the random number generator or the precision of the arithmetic). Also, the time needed for the reliability simulations may become infeasibly long, especially, when the number of different combinations is large. Therefore, the simulation approach was also rejected for use in this thesis.

6.3 Assumptions for novel reliability models

Basic principles for novel approximation models are illustrated here.[*] The following three issues that are related with the disk reliability models are also discussed: disk fault models, fault detection efficiency, and disk access patterns.

6.3.1 Novel approximation method

The Markov model can be simplified with a novel two step approach. First, the Markov model is solved in a steady state case where the faulty state is ignored as depicted in Figure 15.

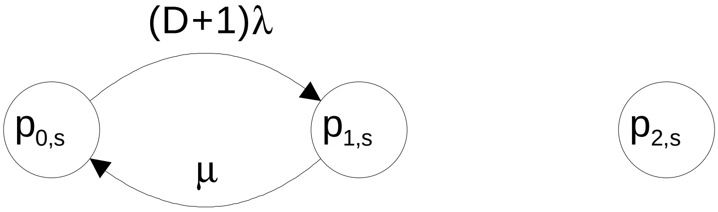

Figure 15. Steady state approximation of the traditional Markov model (D is the number of disks, λ is the disk failure rate, μ is the repair rate, and $p_{x,s}$ defines the approximation of the probability of the system being at state x where x defines the number of faulty disks in the disk array)

From the steady state model, the probabilities of being in states $p_{0,s}$ and $p_{1,s}$ are achieved as follows:

$$p_{0,s} = \frac{\mu}{(D+1)\lambda + \mu} \tag{31}$$

and

$$p_{1,s} = \frac{(D+1)\lambda}{(D+1)\lambda + \mu}. \tag{32}$$

Then, it is possible to approximate the failure rate using the transition that is illustrated in Figure 16 [Laininen 1995].

The failure rate in the approximation can be expressed as

[*] After this point, all ideas, reliability models, and analysis that are presented in this thesis have been done by the author of this thesis.

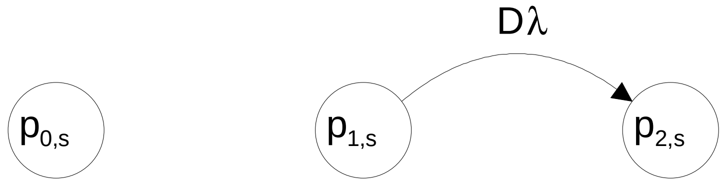

Figure 16. Approximation of the failure rate (D is the number of disks, λ is the disk failure rate, and $p_{x,s}$ defines the approximation of the probability of the system being at state x where x defines the number of faulty disks in the disk array)

$$\lambda_{fail,s} = p_{1,s} D\lambda = \frac{D(D+1)\lambda^2}{(D+1)\lambda + \mu} \tag{33}$$

from which MTTDL for the simplified model can be achieved as

$$MTTDL_s = \frac{1}{\lambda_{fail,s}} = \frac{(D+1)\lambda + \mu}{D(D+1)\lambda^2}\,. \tag{34}$$

This approximation is valid only when the disk repair rate is significantly higher than the total disk failure rate (i.e., $\mu >> (D+1)\lambda$) when $p_{0,s} >> p_{1,s}$.

When the exact and the simplified models are compared, the MTTDL ratio is

$$\frac{MTTDL_s}{MTTDL_{TMM}} = \frac{\frac{(D+1)\lambda + \mu}{D(D+1)\lambda^2}}{\frac{(2D+1)\lambda + \mu}{D(D+1)\lambda^2}} = \frac{(D+1)\lambda + \mu}{(2D+1)\lambda + \mu} \tag{35}$$

while the error is

$$\varepsilon_s = 1 - \frac{MTTDL_s}{MTTDL_{TMM}} = 1 - \frac{(D+1)\lambda + \mu}{(2D+1)\lambda + \mu} = \frac{D\lambda}{(2D+1)\lambda + \mu} = \varepsilon(D, \lambda, \mu)\,. \tag{36}$$

In a practical case, $1/\mu$ is of the order of 100 hours, $1/\lambda$ is of the order of 100 000 hours and D is order of tens. Hence, the error is around

$$\varepsilon(D = 10, \lambda = 1/100000h, \mu = 1/100h) = \frac{10 \times \frac{1}{100000h}}{(2 \times 10 + 1) \times \frac{1}{100000h} + \frac{1}{100h}} \approx 0.98\%\,. \tag{37}$$

The approximation underestimates the value of MTTDL because the error is always positive

($\varepsilon(D,\lambda,\mu) > 0$). Thus, the approximation gives a slightly pessimistic estimation for the system reliability.

6.3.2 Disk array fault models

The faults in a disk array system can be divided into two major categories: disk related faults and other faults [Hillo 1994, Räsänen 1994, Cioffi 1990, Sierra 1990, Schulze 1988, Williams 1988]. In a disk related fault, part of the stored data is lost due to a fault in a disk unit. The other faults do not necessarily cause loss of data, but may cause unavailability of it. For example, power failure or data bus fault may not effect the actual data, but the data is momentarily unavailable.

In highly reliable systems, such as modern disk arrays, continuous availability of data is a critical factor of reliability. Hence, unavailability of data can be considered as fatal as the actual data loss. On the other hand, a system, that accepts short down times, can also tolerate data unavailability. For example, if data of a database can be restored by using a backup and a log file together with a recovery procedure, it is then acceptable that the disk array may be unavailable for a short time because of software or hardware faults.

In the next three chapters, the main emphasis is on the reliability aspects of disks omitting the effects of other components (such as disk controller, cables, power supply and software). In Chapter 10, also those components are briefly discussed.

Transient and permanent disk faults

The first disk fault category divides the faults based on the duration of the fault. The duration of the fault can be short (i.e., transient or temporary fault) or long (i.e., permanent fault).

A transient fault has two alternative reasons of occurrence. First, a transient fault may be caused by a change of the system state (e.g., a temperature change may cause a disk to recalibrate itself resulting in a disk access failure) [Kamunen 1996, Räsänen 1996, ANSI 1994, Räsänen 1994, Koolen 1992, McGregor 1992]. Typically, just by retrying the same request, it is possible to complete the request with no errors. Second, the fault may have a more permanent nature, but by altering the request slightly (e.g., by reading a sector with ECC enabled or reading the data slightly off the track), the fault can be bypassed or recovered.

Transient faults are often the first signs of media deterioration and they can be used for predicting permanent faults. However, all transient faults do not necessarily indicate media deterioration. Instead, they can be due to, for example, recalibration that is caused by temperature change [Räsänen 1994, Koolen 1992, McGregor 1992]. It is very unlikely that transient faults which are not related to the media deterioration will occur at the same location several times. Hence, by keeping a log of the location of transient faults, it is possible to sort out real deteriorated areas.

In contrast to transient faults, it is not possible to repair a permanent fault by retrying. The cause for a permanent fault can be either mechanical (e.g., bearings, motor or disk arms) or electrical (e.g., controlling logic or bus interface logic).

A permanent fault may also be caused by damage to the magnetic material. It can deteriorate either gradually or instantly. An instant degradation can be caused, for example, by a head crash or if the temperature of a disk rises over the so called *Curie point* when the entire disk loses its storage capacity [Räsänen 1994]. The disk temperature can rise, for example, due to an unrelated fault such as a faulty fan.

Disk unit faults and sector faults

The second disk fault category divides the faults based on the fault magnitude: total or partial. A fault can effect an entire disk (i.e., disk unit fault) or small part of it (e.g., sector fault). The size of the fault can be also intermediate (e.g., effecting one disk surface), but these faults are here considered to be disk unit faults as significant part of the data is effected.

6.3.2.1 Disk unit faults

Traditionally, the disk unit failure rates have been modeled using constant failure rates (leading to exponential fault distribution) [Schwarz 1994, Gibson 1991, Sierra 1990]. Constant failure rate is generally used for simplifying the reliability calculation of complex systems [Shooman 1968]. Beside the constant failure rate, also more complex failure rate models have been proposed (e.g., models leading to Weibull distribution) [Gibson 1991].

The actual disk unit failure rate is much more complicated. First of all, most electronic devices, such as hard disks, follow the *bathtub* curve (with high "infant" and "old age" failure rates) as shown in Figure 17 [Schwarz 1994, Gibson 1991, Sierra 1990, Shooman 1968]. However, during the useful lifetime of a disk, the failure rate is assumed to be more or less constant. Second, disk unit faults are not as independent of each other as assumed by typical fault models [Räsänen 1994, Gray 1993, Hillo 1992, Gibson 1991]. Third, the disk unit failure rate varies a lot between the disks of the different manufacturing batches [Voutilainen 1996, Schwarz 1994, Gray 1993, Hillo 1992].

Faults in disks are tightly related to each other. For example, the sector fault probability is significantly higher on those areas near by the area of known faults. Also, all sectors on the same disk surface suffer from the quality of a bad head. Furthermore, disks in the same storage cabinet are usually vulnerable to same cable, power supply, and ventilation faults.

The quality of disks depends also very much on the manufacturing batch. Typically, the disk

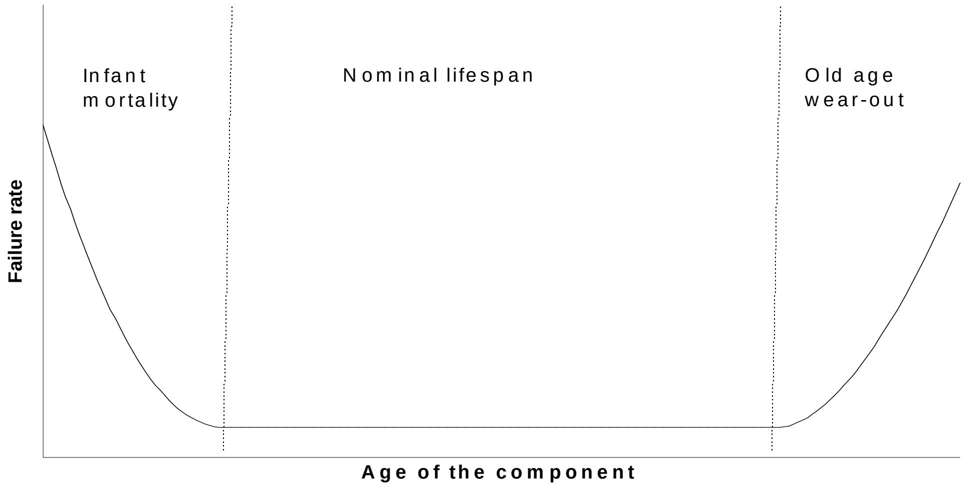

Figure 17. The bathtub curve of conventional lifetime distribution for an electrical device

properties (such as failure rate) in the same batch are similar, but among the different batches the quality can vary significantly as shown below. This effects the reliability in two ways. First, the actual reliability may vary dramatically and, for some units, MTBF can be only a fraction of the average MTBF. Second, if the disk array is built using disks from the same manufacturing batch, there is a significantly larger probability of having a second disk unit fault just after the first one than what the model of independent disk failure rates would predict [Kamunen 1996, Räsänen 1996, Hillo 1993, Hillo 1992]. This emphasizes the significance of the fast disk repair process and selecting the disks from different manufacturing batches.

As a practical example, there was a batch of disks of which significant portion (over 10%) failed in matter of hours after initial start up at the end user site [Hillo 1992]. The reason for this was a change in lubricating oil. It caused no problem in the factory, but transportation in cold environment changed completely the properties of the lubricant. An other example is that about 15% of 130 hard disk units installed in one customer got bad sectors within one year [Voutilainen 1996].

In practical systems, the reliability of a disk also depends heavily on how the disk is used. Several factors will shorten the life span of the disk. Some of them are listed in Table 5 [Hillo 1994, Räsänen 1994].

In this thesis, deterioration of the magnetic media is considered to be independent of the usage of the disk. This means that reading from or writing to a disk is assumed not to deteriorate the magnetic media. On the contrary, the media is assumed to deteriorate by itself also causing unaccessed areas to become faulty [Cioffi 1990, Sierra 1990, Schulze 1988, Williams 1988]. The read process is considered to have no effect on the media deterioration as the head is not touching

Table 5. Actions that effect on the disk life span

Action	Effects or potential problems	Comments
Start/Stop	• The motor wears out • Extensive stress on bearings	Typically about 10 000 - 100 000 starts/stops allowed
Temperature cycle up/down	• Extensive electrical or mechanical wear out	One cycle between 10°C and 40°C corresponds to one day operation time in constant temperature
Constant high temperature	• Expedites wear out • Risk of losing all data due to overheating	Temperature over Curie point causes permanent damage [Kuhn 1997]
High seek activity	• Higher power consumption • Increased temperature • Seek motor wear out	More heat and inadequate power may cause reduced lifetime of a power supply and disk malfunction due to lack of enough current
Uneven access pattern	• May cause extensive seeks • Higher latent fault possibility	Most of the applications are accessing unevenly the disk space; Uneven access pattern may cause more seek activity
Dust and humidity	• Electrical components wear out • Ventilation problem	Potential temperature increase
Vibration	• Mechanical wear out • Mechanical defects	Vibration may loosen components or connectors

the surface, thus no mechanical stress is focused on the magnetic storage media. However, other parts of the disk can suffer from extensive usage of the disk as listed in Table 5.

The actual media deterioration is caused by the impurity of the magnetic media and gradual degradation of the magnetic field [Sierra 1990]. If the magnetic media gradually loses its information, it can be refreshed (read and rewritten). Typically, the degradation and the impurity of the material are unevenly distributed.

The MTBF values of hard disks have improved in recent years rapidly. Ten years ago, the average MTBF was around 20 - 40 000 hours, but the official manufacturers' MTBF figures are currently around 500 000 - 1 000 000 hours [Quantum 1996a, Seagate 1996c]. However, the practical MTBF figures can be significantly less (even as low as 10 000 - 65 000 hours) especially when the disks are heavily loaded [Räsänen 1996, Voutilainen 1996, Hillo 1994].

In this thesis, constant failure rate for disk unit faults is used. MBTF figures from 10 000 hours to 1 million hours are used. This range covers the MTBF figures that are commonly presented in the technical literature in order to keep the results comparable with other studies such as [Schwarz 1994, Hillo 1993, Gibson 1991].

6.3.2.2 Sector faults

Media deterioration typically effects only a limited area of the disk surface. The minimum area that is normally effected is a sector [Haseltine 1996, Räsänen 1996, Scritsmier 1996, ANSI 1994, Seagate 1992, Seagate 1992a]. To survive such events, modern disks have a repair mechanism to reconstruct a faulty sector by replacing the faulty sector with a spare one while still maintaining the same logical representation as described in Chapter 4.

Generally, faults are assumed to be independent of each other, but, with the sector faults, the nearby sectors (previous or next sector on the same track or sectors on the neighboring tracks) of the faulty one have significantly higher fault probability. This has a major effect on the sector repair algorithms as after a sector fault the repair process should also check nearby sectors. Such an enhancement can be implemented as a part of the repair procedure in a scanning algorithm.

The repair process of a sector fault is significantly faster than the repair of a disk unit fault. Typically, a sector repair takes of the order of hundreds of milliseconds, while constructing an entire disk may take easily more than an hour [Räsänen 1994, Kamunen 1994, Hillo 1993].

Based on practical experience, typical hard disks have about the same amount of sector faults as entire disk unit faults (i.e., a sector fault on any sector of a disk is encountered as often as a faulty disk unit) [Kamunen 1996, Räsänen 1996]. Hence, the faults in a disk are split evenly between sector faults and disk unit faults in this thesis when both faults are considered. For example, if the conventional disk array model uses 100 000 hours MTBF for the disk faults (i.e., sector faults are ignored and all faults are considered to be disk unit faults), the disk failure rate is $1/_{100000h}$. Then, the disk failure rate is $1/_{200000h}$ in the enhanced model and the sector failure rate per sector is $1/_{S\times 200000h}$ where S is the number of sectors in the disk.

6.3.3 Disk unit fault detection efficiency

Detection of disk unit faults is typically fast. Both the transient and permanent disk unit faults are detected quickly by the disk control protocol [ANSI 1994, Räsänen 1994, Seagate 1992, Seagate 1992a]. The disk unit fault can be reported immediately to upper layers by the disk controller when an access to a disk fails. It is also possible that the disk or the disk controller can first retry to recover the problem a few times by itself before reporting the fault to upper layers (such as the disk driver or the operating system). The SCSI protocol specifies the maximum response time in which the disk must reply to the disk controller [ANSI 1994, Seagate 1992, Seagate 1992a, ANSI 1986]. This time is significantly shorter than the average access time of a disk. In practice, if the disk is not giving the initial acknowledgement to a controller request in a fraction of a second, it is considered that an error has occurred and a special error recovery procedure should be started.

The latent disk unit faults can be reduced by polling. This is mainly used for disks that are accessed only very seldom. An array controller polls with a constant interval the disks to see that they are still alive [Kamunen 1996, Räsänen 1996, Hillo 1994]. By having the poll interval in the order of seconds, the latent disk unit faults can be practically eliminated as MTBF of a normal disk is significantly higher (in order of tens or hundreds of thousands of hours).

In this thesis, the latent disk unit faults are ignored. As it has been shown above, the disk unit faults can be detected so quickly that it is possible to consider that there are no latent disk unit faults. Even in the case of a very large disk array with a hundred disks, a disk unit fault is detected in a few tens of seconds [Räsänen 1996].

6.3.4 Sector fault detection efficiency

The fault detection of an accessed sector is related to two factors: parameter settings in a disk and the success probability of the disk access.

Parameter setting

Depending on the parameter settings, the disk either tries to recover the media deterioration by itself or it reports the fault to higher levels immediately when the disk access fails the first time as described in Chapter 4 [Räsänen 1996, ANSI 1994, Antony 1992, Seagate 1992a]. The number of retries should be limited because a large number of retries would significantly delay other disk requests when a faulty sector is accessed. Because of the response time requirements, the retry count is typically low, one or two retries, but it could be up to ten times if the response time is not so critical [Räsänen 1996].[*]

Detection success probability

When a faulty sector is accessed, the fault is detected with a certain probability. Typically, the fault detection probability of a single access is quite close to one, but two special cases should be considered:

- False fault detection: In some cases, the disk may report an error even when the media has no actual problem [ANSI 1994, Seagate 1992a]. The reason for this is typically a transient fault in the disk.

[*] In practice, a disk access succeeds in either of the first two tries or it is not going to succeed at all [Räsänen 1996]. The first try may fail because of a transient fault, but it is very unlikely that an unrelated transient fault would occur again at the second try. If a disk is configured to retry by itself in case of an error, there is typically no need for a retry also in the disk controller level.

- Undetected faults: Some of the sector faults may not be detected during every access. The early warning signs of the potential sector faults are given by recoverable errors. If the disk is hiding the retries, the warning signs may be missed [Räsänen 1996].

For simplicity, it is assumed in this thesis that the sector faults are detected with 100% probability whenever the sector is accessed. In practice, actively used sectors are accessed so often that their fault detection probability would be close to one in any case. The only problem would be the seldom accessed sectors.

If the fault detection probability (per access) were less than 100%, the sector fault detection rate would be lower and several accesses to the same sector would be required to detect a fault. This would mainly decrease the latent fault detection rate. It should be noted that the sector fault detection rate will be zero in all cases if the sector is not accessed at all.

6.3.5 Other faults

Other faults (such as in the disk controller, main computer, etc.) are ignored in this phase. They will be discussed later in Chapter 10.

6.3.6 Sector fault detection rate

There are two ways to detect sector faults: by regular user disk requests and by scanning.

6.3.6.1 Detection with user disk requests

The regular user[*] disk accesses can also be used for detecting latent sector faults. While the user disk requests are accessing the disk space, they are simultaneously reporting the status of those sectors. If a problem is found, it can be fixed using the sector repair technique and redundant information on the other disks as described in Chapter 4 [Scritsmier 1996, ANSI 1994, Räsänen 1994, Platt 1992, ANSI 1986].

It should be noted that only read requests may detect a latent fault [Räsänen 1996, Scritsmier 1996]. When data is written to the disk, the old context of the sector has no value (except in the RAID-5 case). If the write requests were used for detecting sector faults, it would require read after every write operation (using WRITE AND VERIFY command), thus causing extra overhead and performance degradation [Räsänen 1996, Scritsmier 1996, ANSI 1994]. Thus, the sector fault

[*] Here, the “user” is understood broadly as an entity (such as an operating system, an application program, an end user or a disk array driver) that is accessing the disk array. The “user” is distinguished from the scanning algorithm that is actually accessing the disk because of its interest in the quality of the disk storage media and not because of data on the disk.

detection by the user request is a function of read activity and distribution of the read requests.

The sector fault detection rate of the user read requests can be expressed as a function

$$\mu_{s,user} = f(RA_u, RD_u, p_{fd}) \tag{38}$$

where RA_u is user read activity (measured as the number of operations in a unit of time), RD_u is user read distribution, and p_{fd} is the probability of sector fault detection. Parameter RA_u describes the total activity of the user disk read requests that are issued to a disk and are directly related to the user activity. Parameter RD_u describes the distribution of the user disk requests. When the requests are concentrated on certain disk locations, fewer distinct sectors are actually accessed and therefore fewer sector faults can be detected. The sector fault detection probability is assumed to be one ($p_{fd} = 1$).

Simplification of the fault detection model

For simplicity, it is assumed that the sector fault detection rate of the user read requests is a constant that depends only on the above mentioned parameters (i.e., user read activity and distribution). This leads to an exponential distribution of the fault detection time. In practice, the fault detection rate depends heavily on the location of the fault as, when the fault falls into a commonly (rarely) accessed sector, the fault detection rate is much higher (lower) than average. The constant fault detection rate is used for simplifying the analytical model that is presented in the following chapter.

6.3.6.2 Detection with scanning disk requests

The read requests of a scanning algorithm are specificly used for detecting latent sector faults. Actually, there is no difference between the user read requests and the scanning read requests from the point of view of a disk if a normal READ command is used [ANSI 1994, Seagate 1992a]. Only, the distribution of the read requests is different (i.e., the scanning read requests go through the entire disk periodically).

The sector fault detection rate by the read requests of the scanning algorithm can be expressed as a function

$$\mu_{s,scan} = f(RA_s, RD_s, p_{fd}) \tag{39}$$

where RA_s describes the activity of the scanning read requests that are issued to a disk (measured as the number of operations in a unit of time) and RD_s describes the distribution of the scanning disk requests. The scanning requests are distributed evenly over the disk space so that all sectors will be accessed in every scanning cycle.

The average sector fault detection time (by the scanning requests) is half of the time that it takes to scan the entire disk. For example, if the disk is scanned through every 24 hours, the sector fault is detected by the scanning algorithm on the average within 12 hours after occurring.

Simplification of the fault detection model

For simplicity, it is assumed that the sector fault detection rate of the scanning read requests is a constant that depends only on the scanning read activity. The constant sector fault detection rate leads to an exponential distribution of the detection time (i.e., there is non-zero probability that sector fault detection could take much longer time than the scanning cycle). This assumption underestimates the reliability. However, the constant fault detection rate is used for simplifying the analytical model that is presented in the following chapter.

6.3.6.3 Combined sector fault detection rate

The sector fault can be detected either by a user disk request or a scanning disk request. Thus, the combined fault detection rate can be expressed as follows

$$\mu_s = \mu_{s,user} + \mu_{s,scan} . \tag{40}$$

In practice, the combined sector fault detection rate is dominated by the scanning process as it is accessing all sectors in matter of hours while it may take several days or even weeks to access all sectors by the user disk requests.

6.3.7 User access patterns

A user access pattern affects the detection rate of the latent sector faults. Typically, it is assumed that the user access pattern is uniformly distributed over the entire disk space [Hou 1994, Seltzer 1992, Miller 1991, Reddy 1991a, Chen 1990, Reddy 1990a, Seltzer 1990a, Olson 1989, Bhide 1988, Ousterhout 1988]. The uniform access pattern is used for simplifying the performance analysis. From the point of view of the reliability analysis, the access pattern has traditionally been considered to be insignificant as the normal reliability analysis approach does not consider sector faults but only disk unit faults.

A practical disk access pattern is typically significantly different from any mathematical models. An example of such access patterns is illustrated in Figure 18 [Kari 1992]. The characteristics of these access patterns are the high peaks in certain areas and no accesses to others. Similar

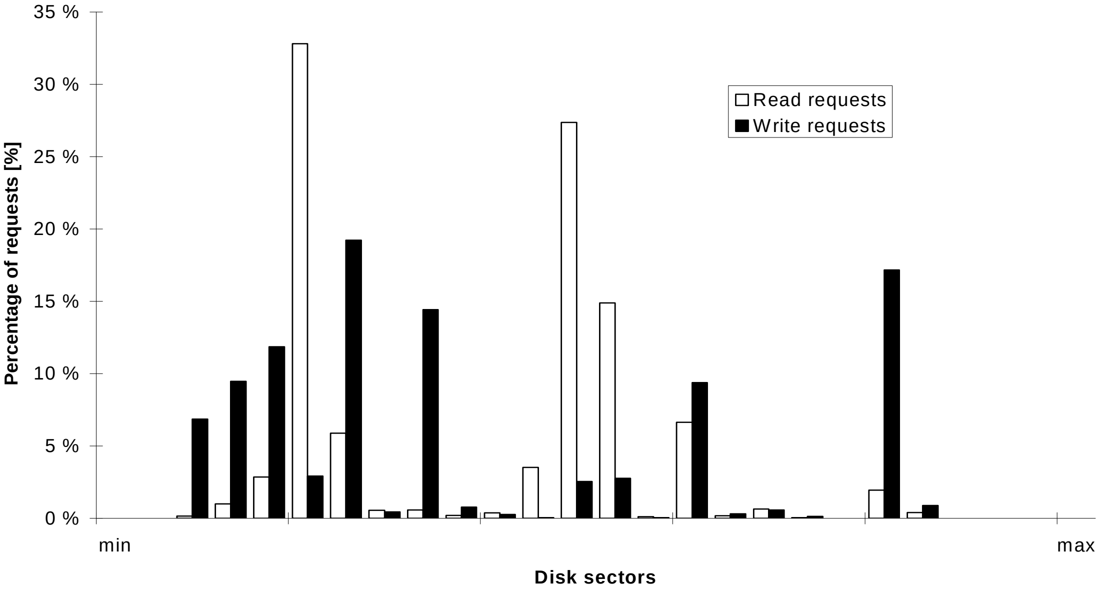

Figure 18: Example of an actual disk access pattern (the density function)

Table 6: Four distributions of the access patterns used in the analysis

Type of access pattern	Request distribution (b_i of requests falls into c_i of the area)
Uniform	100% of requests fall evenly over 100% of the area
Single-80/20	20 % of requests fall into 80% of the area 80 % of requests fall into 20% of the area
Double-80/20	4 % of requests fall into 64% of the area 32 % of requests fall into 32% of the area 64 % of requests fall into 4% of the area
Triple-80/20	0.8 % of requests fall into 51.2% of the area 9.6 % of requests fall into 38.4% of the area 38.4 % of requests fall into 9.6% of the area 51.2 % of requests fall into 0.8% of the area

observations have been made also in [Mourad 1993].

Actual user access patterns have also very high data locality where the next disk access is close to the previous one (e.g., read and then write same location) [Hou 1994]. This has a major effect on the performance, but it also effects the reliability as the mechanical parts are not wearing so much as the seeks are generally shorter. On the other hand, the latent sector fault detection rate is much lower as not so many different sectors are accessed.

This thesis uses four user access patterns as listed in Table 6. These patterns represent various concentrations of the user requests. The uniform access pattern provides a reference model where the user accesses are spread over the disk space evenly and therefore it has the highest detection rate for the latent sector faults while the other access patterns concentrate more and more into fewer and fewer sectors. Here, the common 80/20-rule is used as a basis (i.e., so called Zipf's law) [Bhide 1988]. Hence, the Single-80/20, Double-80/20, and Triple-80/20 access patterns try to represent more accurately the practical user access patterns than the conventional uniform access pattern [Kari 1992, Bhide 1988]. A similar access pattern division has been used earlier when 70/30-rule has been used instead of 80/20 [Gibson 1991, Kim 1987].

Practical access patterns fall probably in between Double-80/20 and Triple-80/20. For example in Triple-80/20, 0.8% of the modern 1 GB hard disk is 8MB of the disk space that easily covers disk index tables, directory entries, and database index tables.

The uniform distribution and the various 80/20 distributions are illustrated in Figure 19 as a function of the disk space. For example, 90% of all disk requests fall in the disk space area that represents about 90%, 60%, 30%, and 11% of all disk sectors in Uniform, Single-80/20, Double-80/20, and Triple-80/20 access patterns, respectively.

The estimated coverage (i.e., the number of accessed distinct sectors divided by the total number of sectors on the disk) for the different access patterns can be expressed with the following equations for Uniform, Single-80/20, Double-80/20, and Triple-80/20 access patterns, respectively [Laininen 1995]:[*]

$$C_{Uniform}(S_a, S) = c_1\left[1-(1-\frac{b_1/c_1}{S})^{S_a}\right], \tag{41}$$

$$C_{Single-80/20}(S_a, S) = \sum_{i=1}^{2} c_i\left[1-(1-\frac{b_i/c_i}{S})^{S_a}\right], \tag{42}$$

[*] Detailed proof of the equations (41) - (44) is omitted from this thesis. Instead, the accuracy of those equations is shown using a simulation approach of which results are presented and compared in Figure 20.

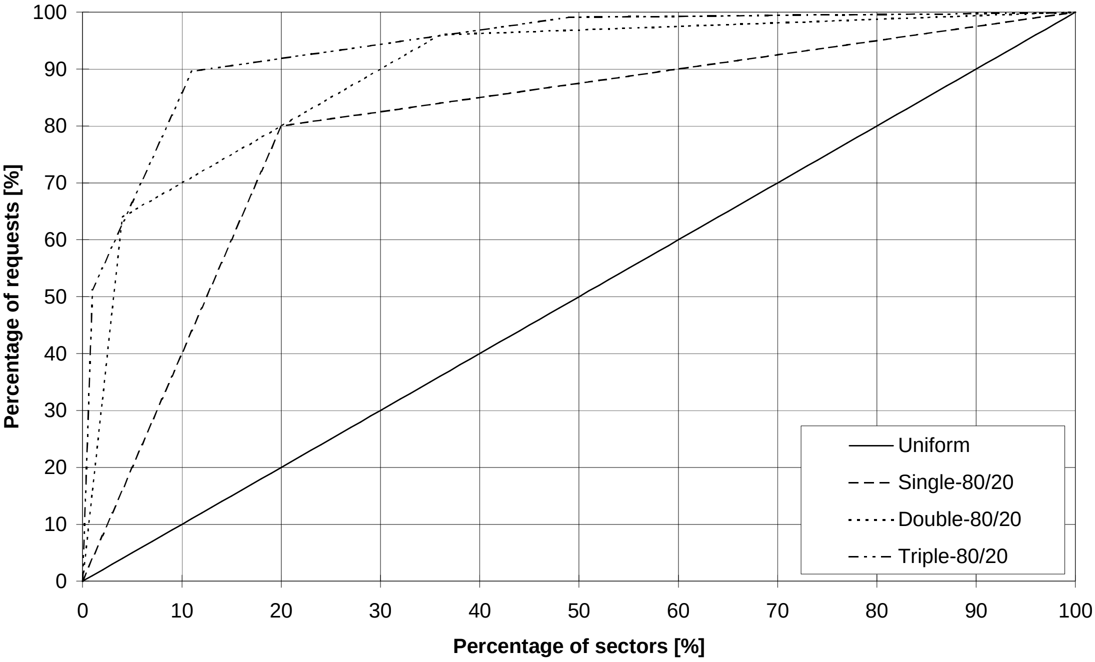

Figure 19. Distribution function of the different access patterns as a function of disk space

$$C_{Double-80/20}(S_a, S) = \sum_{i=1}^{3} c_i \left[1 - (1 - \frac{b_i / c_i}{S})^{S_a} \right], \tag{43}$$

and

$$C_{Triple-80/20}(S_a, S) = \sum_{i=1}^{4} c_i \left[1 - (1 - \frac{b_i / c_i}{S})^{S_a} \right]. \tag{44}$$

where S_a is the total number of requested sectors and S is the total number of sectors in the disk while b_i and c_i are specified in Table 6. By applying the values in Table 6, the following results are achieved for numerical values:

$$C_{Uniform}(S_a, S) = \left[1 - (1 - \frac{1}{S})^{S_a} \right], \tag{45}$$

$$C_{Single-80/20}(S_a, S) = \left\{ 0.2 \left[1 - (1 - \frac{4}{S})^{S_a} \right] + 0.8 \left[1 - (1 - \frac{0.25}{S})^{S_a} \right] \right\}, \tag{46}$$

$$C_{Double-80/20}(S_a,S)=\left\{0.04\left[1-(1-\frac{16}{S})^{S_a}\right]+0.32\left[1-(1-\frac{1}{S})^{S_a}\right]+0.64\left[1-(1-\frac{\frac{1}{16}}{S})^{S_a}\right]\right\}, \qquad (47)$$

and

$$C_{Triple-80/20}(S_a,S)=\left\{\begin{array}{l}0.008\left[1-(1-\frac{64}{S})^{S_a}\right]+0.096\left[1-(1-\frac{4}{S})^{S_a}\right]+\\ 0.384\left[1-(1-\frac{\frac{1}{4}}{S})^{S_a}\right]+0.512\left[1-(1-\frac{\frac{1}{64}}{S})^{S_a}\right]\end{array}\right\}. \qquad (48)$$

The estimated coverage of user access patterns is illustrated in Figure 20 as a function of the relative number of accessed sectors. The more uneven access pattern used, the larger number of accesses is needed to achieve the same coverage. For example, 90 % of the disk space is accessed using (on the average) 2.3, 8.3, 30, and 105 times the number of sectors in the disk when the access pattern is Uniform, Single-80/20, Double-80/20, and Triple-80/20, respectively. Hence, the sector fault detection rate (by the user disk accesses) depends heavily on the access pattern.

The estimated coverage is insensitive to the number of sectors in the disk as illustrated in Table 7. The coverage remains practically the same regardless of the number of sectors in the disk when the S_a / S -ratio is kept the same. Thus, analysis can be done without bothering with the actual size

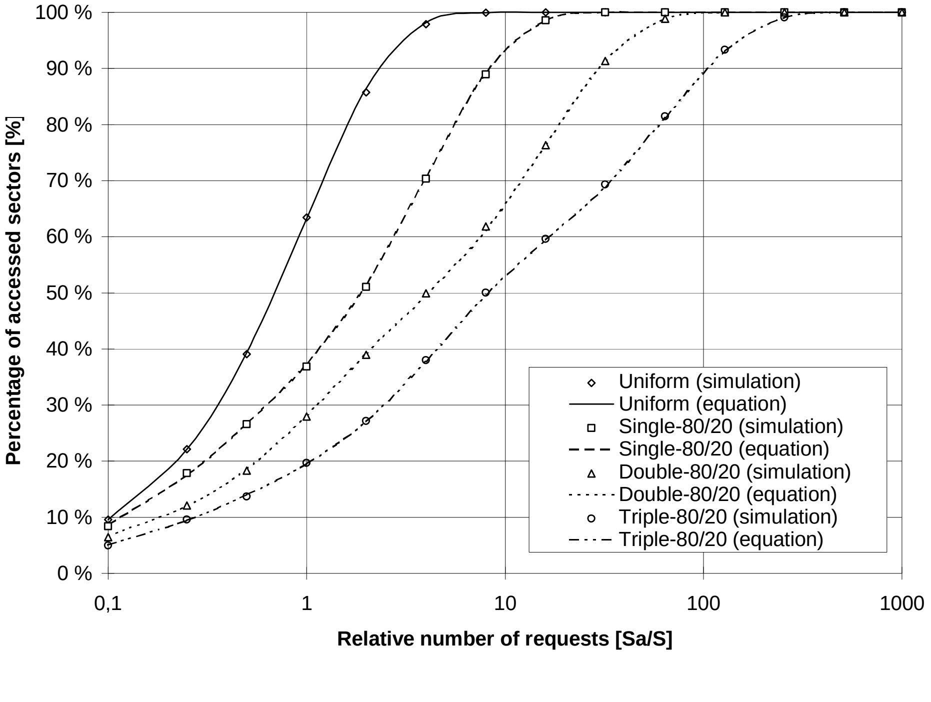

Figure 20. Percentage of all sectors accessed as a function of the total number of accessed sectors

of the disk, because the relative amount of disk requests to the disk capacity (S_a / S -ratio) provides the necessary information. This is very useful as the results of this analysis can be used with all sizes of disks as long as the access patterns are similar. However, the actual access patterns in practice tend to become more and more unevenly distributed when the size of the disk increases.

The mean number of accessed sectors to detect a sector fault can be estimated based on the equations (41)-(44). For example, the mean number of requests to detect a sector fault in Triple-80/20 access pattern can be achieved by the following equation[*]

Table 7: Accuracy of the coverage estimation as a function of the number of sectors in the disk

Access pattern	Coverage (S_a=100, S=100)/ [relative error %]	Coverage (S_a=10 000, S=10 000)/ [relative error %]	Coverage (S_a=1 000 000, S=1 000 000)/ [relative error %]
Uniform	0.633968 +0.2922%	0.632139 +0.0029%	0.632121 0%
Single-80/20	0.373780 +0.1297%	0.373301 +0.0013%	0.373296 0%
Double-80/20	0.281657 +0.2145%	0.281060 +0.0021%	0.281054 0%
Triple-80/20	0.195353 +0.1194%	0.195122 +0.0012%	0.195120 0%
Access pattern	Coverage (S_a=500, S=100)/ [relative error %]	Coverage (S_a=50 000, S=10 000)/ [relative error %]	Coverage (S_a=5 000 000, S=1 000 000)/ [relative error %]
Uniform	0.993430 +0.0169%	0.993264 +0.0017%	0.993262 0%
Single-80/20	0.771155 +0.0465%	0.770800 +0.0046%	0.770796 0%
Double-80/20	0.529709 +0.0188%	0.529611 +0.0019%	0.529610 0%
Triple-80/20	0.416635 +0.0420%	0.416461 +0.0042%	0.416460 0%

[*] Detailed proof of this equation is omitted from this thesis. Instead, the accuracy of the equation is shown using a simulation approach of which results are presented and compared in Table 8.

$$
\begin{aligned}
E_{Triple-80/20} &= \int_0^\infty \frac{d(C_{Triple-80/20}(S_a,S))}{dS_a} S_a d S_a \\
&= \int_0^\infty \sum_{i=1}^{4} \frac{d\left[c_i\left[1-(1-\frac{b_i/c_i}{S})^{S_a}\right]\right]}{d S_a} S_a d S_a \\
&= \sum_{i=1}^{4} \frac{c_i}{\ln\left[\frac{c_i S}{c_i S - b_i}\right]}
\end{aligned}
\tag{49}
$$

from which the normalized value is achieved by

$$
\bar{E}_{Triple-80/20} = \frac{E_{Triple-80/20}}{S} . \tag{50}
$$

Similarly, it is possible to achieve the normalized mean number of accessed sectors to detect a sector fault for the other access patterns. The numerical results are listed in Table 8. In this table, the equation is also compared with the simulation results.

This table shows in practice that it is possible to compare the user access activity and the scanning algorithm activity relative to each other independent of the actual number of sectors in the disk. There is no need to consider the actual size of the hard disks. Also, this allows us to compare the disk reliability (both disk unit fault rate and sector fault rate) with the user access patterns and the scanning algorithm without any actual knowledge of the size of the disks.

Table 8: Estimated relative number of sectors to be accessed to detect a sector fault

Access pattern	Relative number of accessed sectors to detect a sector fault (analytical results)	Relative number of accessed sectors to detect a sector fault (simulation results)	Error between the analytical and simulation results
Scanning algorithm	$\bar{E}_{Scan} = 0.5$		
Uniform	$\bar{E}_{Uniform} = 1.0$	$\bar{E}_{Uniform} = 1.00319$	+0.319%
Single-80/20	$\bar{E}_{Single-80/20} = 3.25$	$\bar{E}_{Single-80/20} = 3.26764$	+0.543%
Double-80/20	$\bar{E}_{Double-80/20} = 10.56$	$\bar{E}_{Double-80/20} = 10.49351$	-0.630%
Triple-80/20	$\bar{E}_{Triple-80/20} = 34.33$	$\bar{E}_{Triple-80/20} = 34.37536$	+0.132%

6.3.8 Conclusions for assumptions

In the following list, a summary of conclusions for the assumptions is collected.

- It is possible to approximate a non-steady state Markov models in two phases when repair rates are significantly higher than failure rates.
- Disks are assumed to have two fault types: disk unit faults and sector faults.
- Disk unit and sector failure rates are constant.
- Repair rates of disk unit and sector faults are constant.
- After a disk unit fault, next disk operation detects the fault in matter of seconds.
- After a sector fault, next read request to that sector detects the fault. This may take a long time.
- Sector faults are independent of the usage of the disk (i.e., reading from or writing to a disk does not deteriorate the disk).
- User disk requests are not accessing the disk evenly. Four different user access patterns are used: Uniform, Single-80/20, Double-80/20, and Triple-80/20.
- Reliability analysis is independent of the actual size of disks.

7. NOVEL RELIABILITY MODELING APPROACHES

In this chapter, two new enhanced reliability models are built in which also sector faults are included. The first reliability model (*Enhanced Markov Model 1, EMM1*) is based on the hot swap principle while the second reliability model (*Enhanced Markov Model 2, EMM2*) is based on the hot spare principle [Chandy 1993, Hillo 1993, RAB 1993, Shooman 1968]. The former model is analyzed both analytically (referred to EMM1) and approximately (referred to EMM1A) while the latter model is analyzed only with approximations (referred to EMM2A).

7.1 Simplifications of Markov models

The Markov models that are used for the enhanced analysis have the following simplifications:

- Exponential repair times;
- Exponential failure times;
- Only zero, one, or two simultaneous sector faults (sector faults that are occurring at other disk addresses than the first one are ignored); and
- Only zero, one, or two simultaneous disk faults.

These assumptions are made to simplify the analysis of the models so that analytical approach can be used. With these simplifications, the number of states in the Markov models can be reduced significantly. This is important because even a simple non-steady state Markov model complicates the analytical approach radically as will be seen later in this chapter.

7.2 Markov models

Three reliability models are used in this chapter: a traditional reliability model and two enhanced models.

7.2.1 Traditional reliability model

The traditional Markov model (TMM) used for analyzing conventional disk arrays was presented in the previous chapter in Figure 14. Equations (18), (19), (20), and (21) express MTTDL and mission success probabilities for 1, 3, and 10 year missions, respectively.

7.2.2 Enhanced reliability model with no on-line spare disks

The first enhanced Markov model (EMM1) of disk arrays is illustrated in Figure 21. Here, the model is derived from TMM that contains only three states as stated for example in [Geist 1993, Gibson 1991].

Failure models

In EMM1, there are two different fault types (sector and disk unit faults) both having their own state in the Markov model. Only, when there are at least two faults at the same time and in the same disk group, the disk array loses its consistency and then data is lost. There are four alternative scenarios how the consistency can be lost:

- After a disk unit fault, a second disk unit fault (in the same disk group) occurs before the repair process of the first disk is completed; or
- After a disk unit fault, a sector fault (in the same disk group) occurs on any sector before the disk repair process is completed;[*] or
- After a sector fault, any other disk (in the same disk group) fails before the sector fault has been detected and repaired; or
- After a sector fault, any other disk (in the same disk group) has also a sector fault at the corresponding sector before the first sector fault has been detected and repaired.[†]

Transition rules

The transition rules of the Markov model for EMM1 are illustrated in Figure 21 and can be expressed as follows:

- The system moves from the fault-free state (p_{00}) to the sector fault state (p_{01}) if any of the sectors in any of the disks becomes faulty (with total rate $(D+1)S\lambda_s$);
- The system moves back from the sector fault state (p_{01}) to the fault-free state (p_{00}) when

[*] In the accurate model, the system may survive the sector fault even when the disk repair process in not yet fully completed. This is because only the corresponding sectors in all other disks need to be accessed not the entire disks. The best (worst) case scenario of the system reliability can be analyzed by excluding (including) the disk reconstruction time from (to) the total disk unit repair time. If no on-line spare disks are used, it may take a significant amount of time before the disk recovery can be started [Gibson 1991]. In this thesis, the worst case scenario has been selected. Thus, the obtained results will provide the lower bound for the system reliability.

[†] In a practical reliability analysis, it is significantly more unlikely to have two sector faults at the corresponding sectors than having, for example, a disk unit fault at the same time as a sector fault. This is because the probability of having a disk unit fault is of the same order of magnitude as the probability of having a sector fault anywhere in the disk space [Räsänen 1996]. As the common disks may have even millions of sectors, the probability that two corresponding sectors become faulty at the same time is marginal when compared to, for example, the probability of having a disk unit fault after a sector fault. However, this fourth data loss scenario is included here just for the sake of symmetry and completeness.

the faulty sector is detected and repaired (with rate μ_s);

- The system moves from the sector fault state (p_{01}) to the disk fault state (p_{10}) if a disk fault occurs at the same disk as the sector fault (with rate λ_d);
- The system moves from the sector fault state (p_{01}) to the data loss state (p_f) if there is a sector fault at the corresponding sector or a disk unit fault in any disk other than the one that has the sector fault (with total rate $D(\lambda_s + \lambda_d)$);
- The system moves from the fault-free state (p_{00}) to the disk fault state (p_{10}) if any of the disks becomes faulty (with rate $(D+1)\lambda_d$);
- The system returns back from the disk fault state (p_{10}) to the fault-free state (p_{00}) when the faulty disk is replaced and repaired (with rate μ_d); and
- The system moves from the disk fault state (p_{10}) to the data loss state (p_f) if there is another disk unit fault or a sector fault on any of the remaining disks (with rate $D(S\lambda_s + \lambda_{df})$).*

In EMM1 as illustrated in Figure 21, $p_{xy}(t)$ indicates the probability of the system being at state *xy* (where *x* is the number of faulty disks and *y* is the number of faulty sectors in the system) at time *t* and $p_f(t)$ is the probability of data loss due to two (or more) simultaneous faults. The other

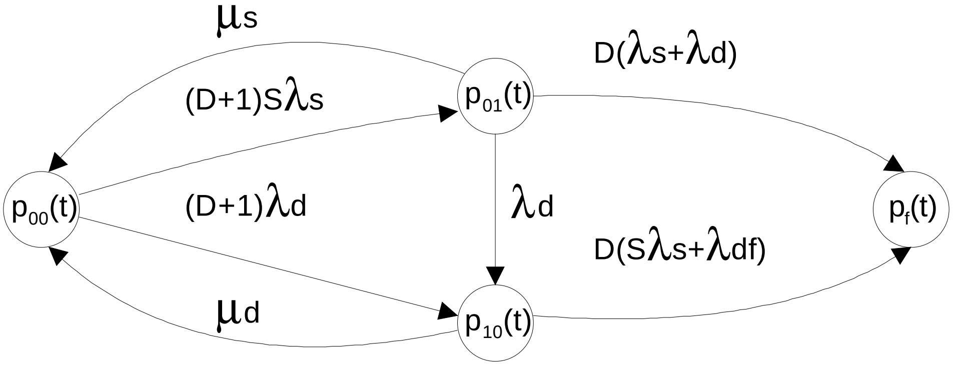

Figure 21. Markov model for EMM1 ($p_{xy}(t)$ indicates the probability of the system being at state xy at time t where x defines the number of faulty disks units and y defines the number of faulty sectors in the disk array. $p_f(t)$ indicates the probability of the data loss. Rest of the parameters are defined in Table 9.)

[*] The second disk failure rate is different from the first one as this indicates the possibility of having interrelated disk faults. This can happen, for example, when temperature is rising in the disk cabinet due to a faulty fan.

Table 9. Parameters for EMM1, EMM1A, and EMM2A

Parameter	Parameter description	Comments
D	number of (data) disks in an array	D disks are needed for data consistency
S	number of sectors in a disk	each sector is treated independently
λ_d	disk unit failure rate	
λ_{df}	disk unit failure rate after the first disk unit fault	this failure rate is greater than (or equal to) the failure rate of the first disk unit fault (λ_d)
λ_s	sector failure rate	
λ_{sd}	spare disk failure rate	failure rate for an online spare disk
μ_d	disk repair rate	includes both disk unit fault detection and repair time
μ_s	sector repair rate	includes both sector fault detection time and repair time
μ_{sd}	spare disk repair rate	includes delayed spare disk fault detection time, new disk ordering, and disk replacement time
μ_{dr}	disk repair rate when spare disk is missing	includes spare disk fault detection time, new disk ordering, and disk replacement time*

parameters are listed in Table 9.

The transition state equations of EMM1 can be expressed as:

$$p'_{00}(t) = -(D+1)(S\lambda_s + \lambda_d)p_{00}(t) + \mu_s p_{01}(t) + \mu_d p_{10}(t), \tag{51}$$

$$p'_{01}(t) = -(\mu_s + \lambda_d + D(\lambda_s + \lambda_d))p_{01}(t) + (D+1)S\lambda_s p_{00}(t), \tag{52}$$

$$p'_{10}(t) = -(\mu_d + D(\lambda_{df} + S\lambda_s))p_{10}(t) + (D+1)\lambda_d p_{00}(t) + \lambda_d p_{01}(t), \tag{53}$$

and

$$p'_f(t) = D(\lambda_s + \lambda_d)p_{01}(t) + D(\lambda_{df} + S\lambda_s)p_{10}(t) \tag{54}$$

where the initial conditions are

* μ_{dr} is basically the same as μ_{sd} with one exception: when the spare disk is not needed, its fault detection time can be significantly longer as the idle spare disk is tested only periodically. The test interval should not be too long because the disk may be faulty when it would be needed. On the other hand, extensive testing with start and stop cycles may reduce the lifetime of the spare disk. On the contrary, when the spare disk is needed after a disk unit fault, the fault detection is practically immediate. This means that μ_{sd} should be less than μ_{dr} .

$$p_{00}(t)+p_{01}(t)+p_{10}(t)+p_f(t)=1, \quad \forall t \tag{55}$$

and

$$p_{00}(0)=1, \quad p_{01}(0)=p_{10}(0)=p_f(0)=0\,. \tag{56}$$

Equations (51)-(54) can be then solved using Laplace transformation with the help of equations (55) and (56). First the equations are moved from t-state to s-state as follows

$$sP_{00}(s)-1=-(D+1)(S\lambda_s+\lambda_d)P_{00}(s)+\mu_s P_{01}(s)+\mu_d P_{10}(s)\,, \tag{57}$$

$$sP_{01}(s)=-(\mu_s+\lambda_d+D(\lambda_s+\lambda_d))P_{01}(s)+(D+1)S\lambda_s P_{00}(s)\,, \tag{58}$$

$$sP_{10}(s)=-(\mu_d+D(\lambda_{df}+S\lambda_s))P_{10}(s)+(D+1)\lambda_d P_{00}(s)+\lambda_d P_{01}(s)\,, \tag{59}$$

and

$$sP_f(s)=D(\lambda_s+\lambda_d)P_{01}(s)+D(\lambda_{df}+S\lambda_s)P_{10}(s)\,. \tag{60}$$

Then, the Laplace equations are solved and translated back to t-state achieving the following results:[*]

$$p_{00}(t)=\sum_{i=0}^{2}\frac{\left[r_i+\mu_s+\lambda_d+D(\lambda_s+\lambda_d)\right]\left[r_i+\mu_d+D(\lambda_{df}+S\lambda_s)\right]e^{r_i t}}{Q_I}\,, \tag{61}$$

$$p_{01}(t)=\sum_{i=0}^{2}\frac{(D+1)S\lambda_s[r_i+\mu_d+D(\lambda_{df}+S\lambda_s)]e^{r_i t}}{Q_I}\,, \tag{62}$$

$$p_{10}(t)=\sum_{i=0}^{2}\frac{\begin{bmatrix}(D+1)\lambda_d r_i+(D+1)\lambda_d\mu_s+(D+1)S\lambda_d\lambda_s+\\(D+1)\lambda_d^{\,2}+D(D+1)\lambda_d(\lambda_s+\lambda_d)\end{bmatrix}e^{r_i t}}{Q_I}\,, \tag{63}$$

and

[*] The equations are solved using Maple V -program. The printout of the program is listed in Appendix A.

$$
\begin{aligned}
p_f(t) &= 1 - p_{00}(t) - p_{01}(t) - p_{10}(t) \\
&= 1 - \sum_{i=0}^{2} \frac{\left[\begin{array}{l}
r_i^2 + \\
\left[\begin{array}{l}\mu_s + (D+1)S\lambda_s + \mu_d + (D+1)\lambda_d + \lambda_d + \\ D(\lambda_s + \lambda_d) + D(\lambda_{df} + S\lambda_s)\end{array}\right] r_i \\
+ \mu_d \mu_s + (D+1)\lambda_d \mu_s + D(\lambda_{df} + S\lambda_s)\mu_s + (D+1)S\lambda_s \mu_d + \\
(D+1)S\lambda_d \lambda_s + D(D+1)S\lambda_s(\lambda_{df} + S\lambda_s) + \lambda_d \mu_d + D(\lambda_s + \lambda_d)\mu_d + \\
(D+1)\lambda_d{}^2 + D(D+1)\lambda_d(\lambda_s + \lambda_d) + D\lambda_d(\lambda_{df} + S\lambda_s) + \\
D^2(\lambda_s + \lambda_d)(\lambda_{df} + S\lambda_s)
\end{array}\right] e^{r_i t}}{Q_I} \\
&= 1 - \sum_{i=0}^{2} \frac{S_i}{Q_I} e^{r_i t} \\
&= 1 - R_I(t)
\end{aligned}
\qquad (64)
$$

where

$$
\begin{aligned}
Q_I = {} & 3r_i^2 + 2(\mu_s + (D+1)S\lambda_s + \mu_d + (D+1)\lambda_d + \lambda_d + D(\lambda_s + \lambda_d) + D(\lambda_{df} + S\lambda_s))r_i + \\
& \mu_d \mu_s + (D+1)\lambda_d \mu_s + D(\lambda_{df} + S\lambda_s)\mu_s + (D+1)S\lambda_s \mu_d + (D+1)S\lambda_d \lambda_s + \\
& D(D+1)S\lambda_s(\lambda_s + \lambda_d) + D(D+1)S\lambda_s(\lambda_{df} + S\lambda_s) + \lambda_d \mu_d + D(\lambda_s + \lambda_d)\mu_d + \\
& (D+1)\lambda_d{}^2 + D(D+1)\lambda_d(\lambda_s + \lambda_d) + D(D+1)\lambda_d(\lambda_{df} + S\lambda_s) + D\lambda_d(\lambda_{df} + S\lambda_s) + \\
& D^2(\lambda_s + \lambda_d)(\lambda_{df} + S\lambda_s)
\end{aligned}
\qquad (65)
$$

and r_i (i=0, 1, or 2) are the three roots of the following equation

$$
\begin{aligned}
& r_i^3 + r_i^2(\mu_s + (D+1)S\lambda_s + \mu_d + (D+1)\lambda_d + \lambda_d + D(\lambda_s + \lambda_d) + D(\lambda_{df} + S\lambda_s)) + \\
& r_i[\mu_d \mu_s + (D+1)\lambda_d \mu_s + D(\lambda_{df} + S\lambda_s)\mu_s + (D+1)S\lambda_s \mu_d + (D+1)S\lambda_d \lambda_s + \\
& D(D+1)S\lambda_s(\lambda_{df} + S\lambda_s) + D(D+1)S\lambda_s(\lambda_s + \lambda_d) + \lambda_d \mu_d + D(\lambda_s + \lambda_d)\mu_d + \\
& (D+1)\lambda_d{}^2 + D(D+1)\lambda_d(\lambda_s + \lambda_d) + D(D+1)\lambda_d(\lambda_{df} + S\lambda_s) + D\lambda_d(\lambda_{df} + S\lambda_s) + \quad . \\
& D^2(\lambda_s + \lambda_d)(\lambda_{df} + S\lambda_s)] + D(D+1)\lambda_d(\lambda_{df} + S\lambda_s)\mu_s + D(D+1)S\lambda_s(\lambda_s + \lambda_d)\mu_d + \\
& D(D+1)S\lambda_d \lambda_s(\lambda_{df} + S\lambda_s) + D^2(D+1)S\lambda_s(\lambda_s + \lambda_d)(\lambda_{df} + S\lambda_s) + \\
& D(D+1)\lambda_d{}^2(\lambda_{df} + S\lambda_s) + D^2(D+1)\lambda_d(\lambda_s + \lambda_d)(\lambda_{df} + S\lambda_s) = 0
\end{aligned}
\qquad (66)
$$

Term $R_I(t)$ can be used for expressing the total reliability of EMM1.

MTTDL of EMM1

MTTDL of EMM1 can be expressed as

$$MTTDL_I = \int_0^\infty (p_{00}(t) + p_{01}(t) + p_{10}(t))dt = \int_0^\infty \sum_{i=0}^{2} \frac{S_i}{Q_I} e^{r_i t} dt = -\sum_{i=0}^{2} \frac{S_i}{Q_I r_i} \tag{67}$$

when $r_i < 0, i = 0,1,2$.

Mission success probabilities of EMM1

Similarly, the mission success probabilities for the one, three, and ten years missions of EMM1 can be expressed as

$$M1_I = R_I(1year), \tag{68}$$

$$M3_I = R_I(3years), \tag{69}$$

and

$$M10_I = R_I(10years). \tag{70}$$

7.2.2.1 Approximation of EMM1

The Markov model illustrated in Figure 21 can be approximated using the same simplification logic that was explained in the previous chapter. This approximation model is called EMM1A. The process is done in two phases as illustrated in Figure 22: steady state simplification (A) and transient state analysis (B).

Steady state simplification

The steady state equations for EMM1A are expressed as:

$$p'_{00,A} = -(D+1)(S\lambda_s + \lambda_d)p_{00,A} + \mu_s p_{01,A} + \mu_d p_{10,A} = 0, \tag{71}$$

$$p'_{01,A} = -(\mu_s + \lambda_d)p_{01,A} + (D+1)S\lambda_s p_{00,A} = 0, \tag{72}$$

and

$$p'_{10,A} = -\mu_d p_{10,A} + (D+1)\lambda_d p_{00,A} + \lambda_d p_{01,A} = 0 \tag{73}$$

while

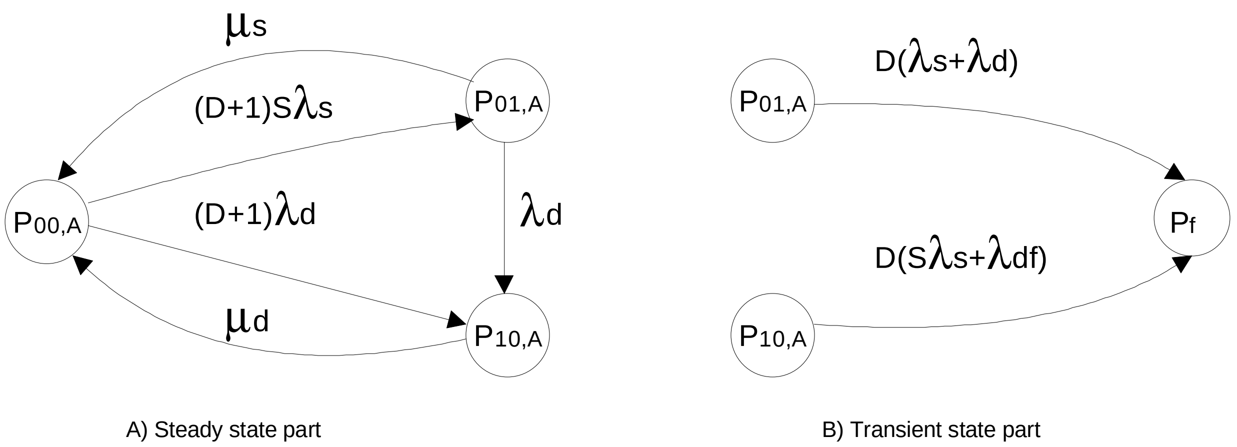

Figure 22. Two phase approximation of EMM1A ($p_{xy,A}$ indicates the approximation of the probability of the system being at state xy where x defines the number of faulty disks units and y defines the number of faulty sectors in the disk array. p_f indicates the probability of the data loss. Rest of the parameters are defined in Table 9.)

$$p_{00,A} + p_{01,A} + p_{10,A} = 1. \tag{74}$$

Solving the above equations (71)-(74) in the steady state leads to the following probabilities of the system being in different states:

$$p_{00,A} = \frac{\mu_d \mu_s + \lambda_d \mu_d}{Q_{I,A}}, \tag{75}$$

$$p_{01,A} = \frac{(D+1)S\lambda_s \mu_d}{Q_{I,A}}, \tag{76}$$

and

$$p_{10,A} = \frac{(D+1)S\lambda_d \lambda_s + (D+1)\lambda_d \mu_s + (D+1)\lambda_d{}^2}{Q_{I,A}} \tag{77}$$

where

$$Q_{I,A} = \mu_d \mu_s + \lambda_d \mu_d + (D+1)S\lambda_s \mu_d + (D+1)S\lambda_d \lambda_s + (D+1)\lambda_d \mu_s + (D+1)\lambda_d{}^2. \tag{78}$$

Transient state analysis

As $p_{00,A} >> p_{01,A}$ and $p_{00,A} >> p_{10,A}$, we will get an approximation for the failure rate of EMM1A model as follows[*]

$$\lambda_{fail,I,A} = D(\lambda_s + \lambda_d)p_{01,A} + D(\lambda_{df} + S\lambda_s)p_{10,A}. \tag{79}$$

From which we get for MTTDL

$$MTTDL_{I,A} = \int_0^\infty e^{-\lambda_{fail,I,A}t}dt = \frac{1}{\lambda_{fail,I,A}} = \frac{1}{D(\lambda_s + \lambda_d)p_{01,A} + D(\lambda_{df} + S\lambda_s)p_{10,A}}. \tag{80}$$

Similarly, the mission success probabilities are expressed as

$$M1_{I,A} = e^{-1year\lambda_{fail,I,A}}, \tag{81}$$

$$M3_{I,A} = e^{-3years\lambda_{fail,I,A}}, \tag{82}$$

and

$$M10_{I,A} = e^{-10years\lambda_{fail,I,A}}. \tag{83}$$

7.2.3 Enhanced reliability model with one on-line spare disk

The second enhanced Markov model (EMM2) of a disk array is illustrated in Figure 23. Here, the model has one spare disk that is used for quick repair of disk unit faults. The first disk repair can be started immediately after the fault detection. The second disk fault can be repaired after the spare disk is replaced. It has been shown that one spare disk is quite sufficient for a disk array [Gibson 1991].

This Markov model is analyzed only using an approximation due to the complexity of the model. The similar approach is used here as with EMM1A.

[*] When $1/\lambda_d$ and $1/S\lambda_s$ are of the order of one hundred thousand hours, mean disk and sector repair times of the order of tens of hours, and the array has tens of disks, then $p_{00,A}$ is about one thousand times larger than $p_{01,A}$ or $p_{10,A}$.

Failure models

In EMM2, there are three different faults (sector, active disk unit, and spare disk unit faults). The states are divided so that a spare disk unit fault can exist at the same time as a sector or an active disk unit fault.* Only, when there are at least two faults at the same time and in the same disk group of the active disks, the disk array loses its consistency and data is lost. There are four alternative scenarios how the consistency can be lost:

- After an active disk unit fault, a second active disk unit fault (in the same disk group) occurs before the repair process of the first disk is completed; or
- After an active disk unit fault, a sector fault (in the same disk group) occurs on any sector of active disks before the disk repair process is completed;† or
- After a sector fault of an active disk, any other active disk (in the same disk group) fails before the sector fault has been detected and repaired; or
- After a sector fault of an active disk, any other active disk (in the same disk group) has also a sector fault at the corresponding sector before the first sector fault has been detected and repaired.‡

Transition rules

The transition rules of EMM2 illustrated in Figure 23 are:

- The system moves from the fault-free state (p_{000}) to the sector fault state (p_{001}) when any of the sectors in any of the active disks becomes faulty (with total rate $(D+1)S\lambda_s$);
- The system moves back from the sector fault state (p_{001}) to the fault-free state (p_{000}) when the faulty sector is detected and repaired (with rate μ_s);
- The system moves from the sector fault state (p_{001}) to the active disk fault state (p_{010}) when a disk unit fault occurs at the same disk as the sector fault (with rate λ_d);
- The system moves from the sector fault state (p_{001}) to the spare disk and sector fault state (p_{101}) when the spare disk becomes faulty (with rate λ_{sd});
- The system moves back from the spare disk and sector fault state (p_{101}) to the sector fault

* A spare disk unit fault does not directly reduce the reliability as the disk array can still tolerate a disk unit or a sector fault after a spare disk fault. Indirectly, the spare disk unit fault effects on the reliability as the repair time is longer.

† The same comment as for the failure models of EMM1.

‡ The same comment as for the failure models of EMM1.

state (p_{001}) when the spare disk unit fault is detected and a new spare disk is installed (with rate μ_{sd});

- The system moves from the sector fault state (p_{001}) to the data loss state (p_f) when there is a sector fault at a corresponding sector or a disk unit fault in any other active disk than the one that has the sector fault (with total rate $D(\lambda_s + \lambda_d)$);
- The system moves from the fault-free state (p_{000}) to the active disk fault state (p_{010}) when any of the active disks becomes faulty (with total rate $(D+1)\lambda_d$);
- The system moves from the active disk fault state (p_{010}) to the spare disk fault state (p_{100}) when the faulty disk is logically replaced with the on-line spare disk and data is reconstructed to that disk (with rate μ_d);
- The system moves from the active disk fault state (p_{010}) to the data loss state (p_f) when there is another disk unit fault or any sector fault on the active disks (with total rate $D(S\lambda_s + \lambda_{df})$);*
- The system moves from the fault-free state (p_{000}) to the spare disk fault state (p_{100}) when the spare disk becomes faulty (with rate λ_{sd});
- The system moves back from the spare disk fault state (p_{100}) to the fault-free state (p_{000}) when the spare disk fault is detected and a new spare disk is installed (with rate μ_{sd});
- The system moves from the spare disk fault state (p_{100}) to the spare disk and sector fault state (p_{101}) when any of the sectors in any of the active disks get faulty (with total rate $(D+1)S\lambda_s$);
- The system moves back from the spare disk and sector fault state (p_{101}) to the spare disk fault state (p_{100}) when the faulty sector is detected and repaired (with rate μ_s);
- The system moves from the spare disk and sector fault state (p_{101}) to the spare disk and active disk fault state (p_{110}) when the disk fault occurs at the same disk as the sector fault (with rate λ_d);
- The system moves from the spare disk and sector fault state (p_{101}) to the data loss state (p_f) when there is a sector fault at a corresponding sector or a disk unit fault in any other

* Like in EMM1, the second disk unit failure rate is different from the first one as this indicates a possibility of having interrelated disk unit faults.

active disk than the one that has the sector fault (with total rate $D(\lambda_s + \lambda_d)$);

- The system moves from the spare disk fault state (p_{100}) to the spare disk and active disk fault state (p_{110}) when any of the active disks get faulty (with total rate $(D+1)\lambda_d$);
- The system moves from the active disk fault state (p_{010}) to the spare disk and active disk fault state (p_{110}) when the spare disk becomes faulty during the disk array repair process (with rate λ_d);
- The system moves back from the spare disk and active disk fault state (p_{110}) to the active disk fault state (p_{010}) when a new spare disk is installed in the array (with rate μ_{dr}); and
- The system moves from the spare disk and active disk fault state (p_{110}) to the data loss state (p_f) when there is another disk unit fault or any sector fault on the active disks (with total rate $D(S\lambda_s + \lambda_{df})$).*

In the model illustrated in Figure 23, p_{wxy} indicates the system being at state *w*,*x*,*y* (where *w* is the number of faulty spare disks, *x* is the number of faulty disks, and *y* is the number of faulty sectors in the system) and p_f is the probability of data loss due to two (or more) simultaneous faults in the active disks. The other parameters are listed in Table 9.

Steady state simplification

The approximation of EMM2 is done in two phases of which the steady state part is illustrated in Figure 24. The approximation of this model is called EMM2A. The steady state equations can be expressed as follows:

$$p'_{000,A} = -((D+1)(S\lambda_s + \lambda_d) + \lambda_{sd})p_{000,A} + \mu_s p_{001,A} + \mu_{sd} p_{100,A} = 0, \tag{84}$$

$$p'_{001,A} = -(\mu_s + \lambda_d + \lambda_{sd})p_{001,A} + (D+1)S\lambda_s p_{000,A} + \mu_{sd} p_{101,A} = 0, \tag{85}$$

$$p'_{010,A} = -(\mu_d + \lambda_d)p_{010,A} + (D+1)\lambda_d p_{000,A} + \lambda_d p_{001,A} + \mu_{dr} p_{110,A} = 0, \tag{86}$$

$$p'_{100,A} = -(\mu_{sd} + (D+1)(S\lambda_s + \lambda_d))p_{100,A} + \lambda_{sd} p_{000,A} + \mu_d p_{010,A} + \mu_s p_{101,A} = 0, \tag{87}$$

* Like in EMM1, the second disk unit failure rate is different from the first one as this indicates a possibility of having interrelated disk unit faults.

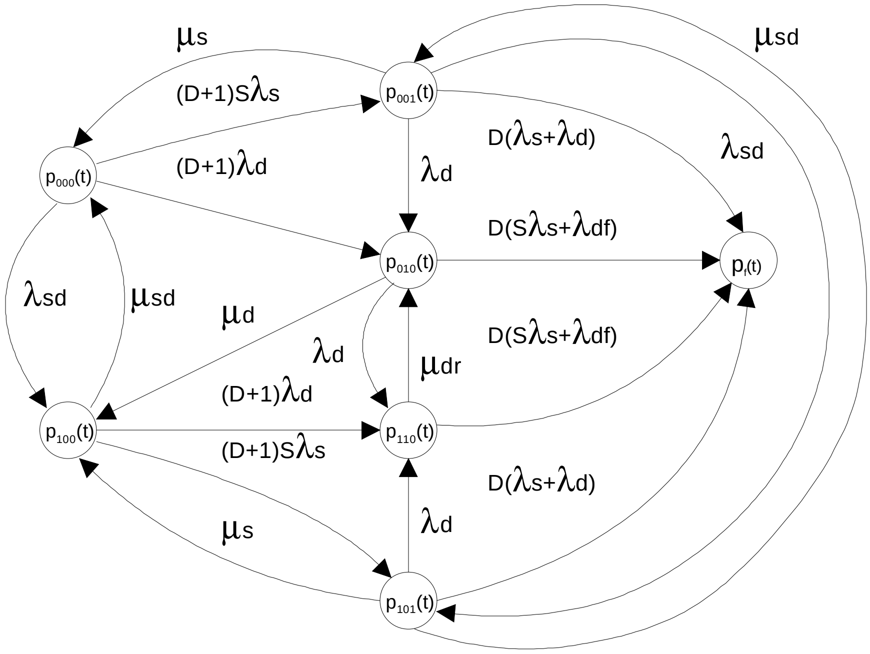

Figure 23. Markov model for EMM2 ($p_{wxy}(t)$ indicates the probability of the system being at state wxy at time t where w defines the number of faulty spare disks units, x defines the number of faulty disks units, and y defines the number of faulty sectors in the disk array. $p_f(t)$ indicates the probability of the data loss. Rest of the parameters are defined in Table 9.)

$$p'_{101,A} = -(\mu_s + \mu_{sd} + \lambda_d)p_{101,A} + \lambda_{sd}p_{001,A} + (D+1)S\lambda_s p_{100,A} = 0\,, \tag{88}$$

and

$$p'_{110,A} = -(\mu_{dr})p_{101,A} + \lambda_d p_{010,A} + (D+1)\lambda_d p_{100,A} + \lambda_d p_{101,A} = 0 \tag{89}$$

where the initial condition is

$$p_{000,A} + p_{001,A} + p_{010,A} + p_{100,A} + p_{101,A} + p_{110,A} = 1\,. \tag{90}$$

Equations (84)-(89) can be then solved with the help of equation (90). Thus, the probabilities are[*]

[*] The equations are solved using Maple V -program. The printout of the program is listed in Appendix A.

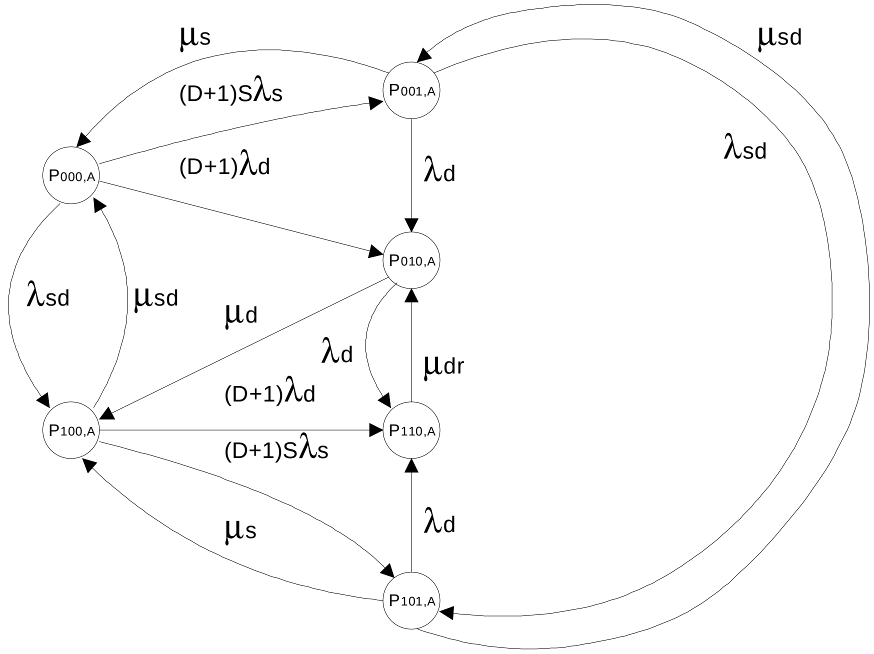

Figure 24. Steady state part of the Markov model of EMM2A ($p_{wxy,A}$ *indicates the approximation of the probability of the system being at state* wxy *where* w *defines the number of faulty spare disks units,* x *defines the number of faulty disks units, and* y *defines the number of faulty sectors in the disk array. Rest of the parameters are defined in Table 9.)*

$$p_{000,A} = \frac{\mu_d \mu_{dr} \mu_{sd} \begin{bmatrix} \lambda_d^{\,2} + \lambda_d \lambda_{sd} + 2\lambda_d \mu_s + \lambda_d \mu_{sd} + \lambda_{sd} \mu_s \\ + (D+1)S\lambda_s \mu_s + \mu_s \mu_{sd} + \mu_s^{\,2} \end{bmatrix}}{Q_{II,A}}, \tag{91}$$

$$p_{001,A} = \frac{(D+1)S\lambda_s \mu_d \mu_{dr} \mu_{sd} (\lambda_d + \lambda_{sd} + (D+1)(S\lambda_s + \lambda_d) + \mu_s + \mu_{sd})}{Q_{II,A}}, \tag{92}$$

$$p_{010,A} = \frac{\mu_{dr} \begin{pmatrix} (D+1)(S\lambda_s \lambda_d + \lambda_d^{\,2} + \lambda_d \mu_s) \\ \begin{bmatrix} (\lambda_d + \lambda_{sd} + \mu_{sd})((D+1)S\lambda_s + \lambda_{sd} + (D+1)\lambda_d + \mu_{sd}) + \\ \mu_s((D+1)\lambda_d + \lambda_{sd} + \mu_{sd}) \end{bmatrix} \end{pmatrix}}{Q_{II,A}}, \tag{93}$$

$$p_{100,A} = \frac{\mu_d \mu_{dr} \begin{pmatrix} ((\lambda_d + \mu_s) \\ (((D+1)\lambda_d + \lambda_{sd})(\lambda_d + \lambda_{sd} + \mu_s + \mu_{sd}) + \\ (\lambda_d + \lambda_{sd})(D+1)S\lambda_s) + (D+1)S\lambda_s \lambda_d \mu_{sd}) \end{pmatrix}}{Q_{II,A}}, \tag{94}$$

$$p_{101,A} = \frac{(D+1)S\lambda_s\mu_d\mu_{dr}\left[\begin{array}{l}(((D+1)\lambda_d+\lambda_{sd})(\lambda_d+\lambda_{sd}+\mu_s)+\\ \lambda_{sd}((D+1)S\lambda_s+\mu_{sd})+(D+1)S\lambda_s\lambda_d)\end{array}\right]}{Q_{II,A}}, \tag{95}$$

and

$$p_{110,A=}\frac{\left\{\begin{array}{l}
\left[\begin{array}{l}\lambda_{\mathrm{sd}}\mu_d\left[(\mathrm{D}+1)\lambda_{\mathrm{d}}\mu_{\mathrm{s}}+(\mathrm{D}+1)\lambda_{\mathrm{d}}{}^2+(\mathrm{D}+1)\mathrm{S}\lambda_{\mathrm{s}}\lambda_{\mathrm{d}}\right]\\ \left[(\mathrm{D}+1)\mathrm{S}\lambda_{\mathrm{s}}+\lambda_{\mathrm{sd}}+\mu_{\mathrm{s}}+\mu_{\mathrm{sd}}\right]\end{array}\right]+\\
\left[\lambda_{\mathrm{d}}{}^2(\mathrm{D}+1)(\lambda_{\mathrm{d}}+\mathrm{S}\lambda_{\mathrm{s}})(\lambda_{\mathrm{sd}}+\mu_{\mathrm{sd}})\left[(\mathrm{D}+2)\lambda_{\mathrm{d}}+(\mathrm{D}+1)\mathrm{S}\lambda_{\mathrm{s}}+\lambda_{\mathrm{sd}}+\mu_{\mathrm{s}}+\mu_{\mathrm{sd}}\right]\right]+\\
\left[(\mathrm{D}+1)\lambda_{\mathrm{d}}{}^2((\mathrm{D}+1)\lambda_{\mathrm{d}}+\lambda_{\mathrm{sd}}+\mu_{\mathrm{s}}+\mu_{\mathrm{sd}})(\lambda_{\mathrm{sd}}\mu_{\mathrm{s}}+\mu_{\mathrm{s}}\mu_{\mathrm{sd}})\right]+\\
\left[(\mathrm{D}+1)\lambda_{\mathrm{d}}{}^2(\lambda_{\mathrm{sd}}+\mu_{\mathrm{sd}})\left[(\mathrm{D}+1)\mathrm{S}\lambda_{\mathrm{s}}\mu_{\mathrm{s}}+\lambda_{\mathrm{d}}\mu_{\mathrm{s}}+(\mathrm{D}+1)\mathrm{S}\lambda_{\mathrm{s}}\mu_d\right]\right]+\\
\left[\lambda_{\mathrm{d}}{}^2(\mathrm{D}+1)\mathrm{S}\lambda_{\mathrm{s}}\left[\mu_d((\mathrm{D}+1)\mathrm{S}\lambda_{\mathrm{s}}+\lambda_{\mathrm{sd}})+\lambda_{\mathrm{d}}(\mathrm{D}+1)(\lambda_{\mathrm{d}}+\mathrm{S}\lambda_{\mathrm{s}})\right]\right]+\\
\left[(\mathrm{D}+1)\lambda_{\mathrm{d}}(\lambda_{\mathrm{d}}+\mu_{\mathrm{s}})\left[\begin{array}{l}(\mathrm{D}+1)\lambda_{\mathrm{d}}\mu_d(\lambda_{\mathrm{sd}}+\mu_{\mathrm{s}}+\mu_{\mathrm{sd}})+\\ \lambda_{\mathrm{d}}{}^2(\mathrm{D}+1)(\lambda_{\mathrm{d}}+\mathrm{S}\lambda_{\mathrm{s}})\end{array}\right]\right]+\\
\left[(\mathrm{D}+1)\lambda_{\mathrm{d}}{}^2\left[\begin{array}{l}(\lambda_{\mathrm{d}}+\mu_d)(\mathrm{D}+1)\mathrm{S}\lambda_{\mathrm{s}}\mu_{\mathrm{s}}+\\ (\mathrm{D}+1)\lambda_{\mathrm{d}}\mu_{\mathrm{s}}(\lambda_{\mathrm{d}}+\mu_d+\mu_{\mathrm{s}})+\\ \lambda_{\mathrm{d}}\mu_d(\mathrm{D}+1)(\lambda_{\mathrm{d}}+\mathrm{S}\lambda_{\mathrm{s}})+\\ \mu_d(\mathrm{D}+1)\mathrm{S}\lambda_{\mathrm{s}}(\lambda_{\mathrm{d}}+\mu_{\mathrm{s}})+\\ \lambda_{sd}\mu_{\mathrm{d}}(\lambda_{\mathrm{d}}+\mu_{\mathrm{s}})\end{array}\right]\right]
\end{array}\right\}}{Q_{II,A}} \tag{96}$$

where

$$
\begin{aligned}
Q_{II,A} = {} & \mu_d \mu_{dr} \mu_{sd} \left[\begin{matrix} \lambda_d{}^2 + \lambda_d \lambda_{sd} + 2\lambda_d \mu_s + \lambda_d \mu_{sd} + \lambda_{sd} \mu_s + \\ (D+1) S \lambda_s \mu_s + \mu_s \mu_{sd} + \mu_s{}^2 \end{matrix} \right] + \\
& (D+1) S \lambda_s \mu_d \mu_{dr} \mu_{sd} \left[\lambda_d + \lambda_{sd} + (D+1)(S\lambda_s + \lambda_d) + \mu_s + \mu_{sd} \right] + \\
& \mu_{dr} \left[\begin{matrix} (D+1)(S\lambda_s \lambda_d + \lambda_d{}^2 + \lambda_d \mu_s) \\ \left[\begin{matrix} (\lambda_d + \lambda_{sd} + \mu_{sd})((D+1)S\lambda_s + \lambda_{sd} + (D+1)\lambda_d + \mu_{sd}) + \\ \mu_s((D+1)\lambda_d + \lambda_{sd} + \mu_{sd}) \end{matrix} \right] \end{matrix} \right] + \\
& \mu_d \mu_{dr} \left[\begin{matrix} ((\lambda_d + \mu_s)(((D+1)\lambda_d + \lambda_{sd})(\lambda_d + \lambda_{sd} + \mu_s + \mu_{sd}) + \\ (\lambda_d + \lambda_{sd})(D+1)S\lambda_s) + (D+1)S\lambda_s \lambda_d \mu_{sd}) \end{matrix} \right] + \\
& (D+1) S \lambda_s \mu_d \mu_{dr} \left[\begin{matrix} (((D+1)\lambda_d + \lambda_{sd})(\lambda_d + \lambda_{sd} + \mu_s) + \\ \lambda_{sd}((D+1)S\lambda_s + \mu_{sd}) + (D+1)S\lambda_s \lambda_d) \end{matrix} \right] + \\
& \left[\begin{matrix} \lambda_{sd} \mu_d \left[(D+1)\lambda_d \mu_s + (D+1)\lambda_d{}^2 + (D+1)S\lambda_s \lambda_d \right] \\ \left[(D+1)S\lambda_s + \lambda_{sd} + \mu_s + \mu_{sd} \right] \end{matrix} \right] + \\
& \left[\lambda_d{}^2 (D+1)(\lambda_d + S\lambda_s)(\lambda_{sd} + \mu_{sd}) \left[(D+2)\lambda_d + (D+1)S\lambda_s + \lambda_{sd} + \mu_s + \mu_{sd} \right] \right] + \\
& \left[(D+1)\lambda_d{}^2((D+1)\lambda_d + \lambda_{sd} + \mu_s + \mu_{sd})(\lambda_{sd}\mu_s + \mu_s \mu_{sd}) \right] + \\
& \left[(D+1)\lambda_d{}^2(\lambda_{sd} + \mu_{sd}) \left[(D+1)S\lambda_s \mu_s + \lambda_d \mu_s + (D+1)S\lambda_s \mu_d \right] \right] + \\
& \left[\lambda_d{}^2 (D+1) S\lambda_s \left[\mu_d((D+1)S\lambda_s + \lambda_{sd}) + \lambda_d (D+1)(\lambda_d + S\lambda_s) \right] \right] + \\
& \left[(D+1)\lambda_d(\lambda_d + \mu_s) \left[\begin{matrix} (D+1)\lambda_d \mu_d (\lambda_{sd} + \mu_s + \mu_{sd}) + \\ \lambda_d{}^2 (D+1)(\lambda_d + S\lambda_s) \end{matrix} \right] \right] + \\
& \left[(D+1)\lambda_d{}^2 \left[\begin{matrix} (\lambda_d + \mu_d)(D+1)S\lambda_s \mu_s + \\ (D+1)\lambda_d \mu_s (\lambda_d + \mu_d + \mu_s) + \\ \lambda_d \mu_d (D+1)(\lambda_d + S\lambda_s) + \\ \mu_d (D+1) S\lambda_s (\lambda_d + \mu_s) + \\ \lambda_{sd} \mu_d (\lambda_d + \mu_s) \end{matrix} \right] \right]
\end{aligned}
\quad . \tag{97}
$$

Transient state analysis

As $p_{000,A} >> p_{001,A}$, $p_{000,A} >> p_{010,A}$, $p_{000,A} >> p_{101,A}$, and $p_{000,A} >> p_{110,A}$ we will get for the failure rate for the approximation based on the Figure 25 as follows

$$
\lambda_{fail,II,A} = D(\lambda_s + \lambda_d)(p_{001,A} + p_{101,A}) + D(\lambda_{df} + S\lambda_s)(p_{010,A} + p_{110,A}) \tag{98}
$$

from which we get for MTTDL

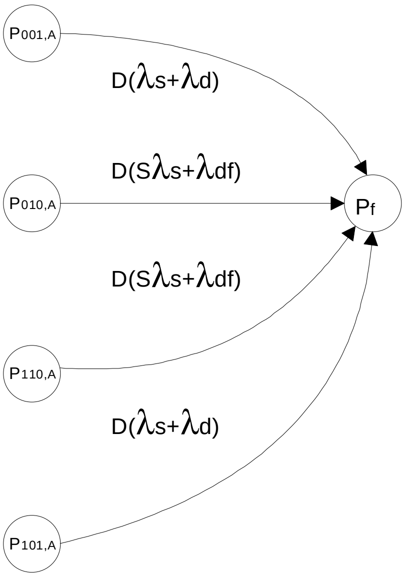

Figure 25. Transient state part of the Markov model of EMM2A ($p_{wxy,A}$ indicates the approximation of the probability of the system being at state wxy where w defines the number of faulty spare disks units, x defines the number of faulty disks units, and y defines the number of faulty sectors in the disk array. p_f indicates the probability of the data loss. Rest of the parameters are defined in Table 9.)

$$\begin{aligned} MTTDL_{II,A} &= \int_0^\infty e^{-\lambda_{fail,II,A}t}dt \\ &= \frac{1}{\lambda_{fail,II,A}} \\ &= \frac{1}{D(\lambda_s+\lambda_d)(p_{001,A}+p_{101,A})+D(\lambda_{df}+S\lambda_s)(p_{010,A}+p_{110,A})} \end{aligned} . \tag{99}$$

The mission success probabilities are then expressed as

$$M1_{II,A} = e^{-1year\lambda_{fail,II,A}}, \tag{100}$$

$$M3_{II,A} = e^{-3years\lambda_{fail,II,A}}, \tag{101}$$

and

$$M10_{II,A} = e^{-10years\lambda_{fail,II,A}}. \tag{102}$$

8. ANALYSIS OF NOVEL RELIABILITY MODELS

In this chapter, new reliability models of Chapter 7, the reliability effects of the proposed scanning algorithms, and a delayed disk array repair process are studied in comparison with traditional disk array reliability models and repair algorithms. The goal is to get knowledge concerning how much the scanning algorithm improves the disk array reliability by detecting the latent sector faults and how much the disk array reliability is decreased when the repair process is either delayed or obstructed. Other reliability scenarios are also studied.

This chapter is divided into four parts: validation of the reliability models, sensitivity analysis of the parameters, accuracy of the approximation models, and reliability scenarios. The first part verifies that the achieved equations provide the same results as the previous studies with the same input parameters. The second part studies how stable the equations are with respect to different input parameters. The third part estimates the accuracy of the approximation models. Finally, the various scenarios of the reliability aspects are evaluated in the fourth part.

8.1 Validation of novel reliability models

In Appendix B, the MTTDL figures are illustrated in various parameter combinations. In this part, the corresponding equations of the technical literature (here called Traditional Markov Model, TMM) [Schwarz 1994, Hillo 1993, Gibson 1991] are compared with the results of EMM1 and EMM2A.* The main objective for this validation is to verify that the new equations agree with the results of the previous studies.

The validation is divided into two parts. The first part compares the equations with exactly the same parameters while the second part compares with different values. In the first three comparisons, some of the parameters, such as sector fault rate, are ignored (i.e., those values are set so low/high that they have no effect). In the last three comparisons, it is checked that the new models give reasonable results also when the new features are included.

Comparison with identical values

Figures B-1, B-2, and B-3 of Appendix B illustrate the comparison of the MTTDL values of

* This comparison uses mainly the MTTDL figures as the mission success probabilities are calculated from the same origin and equations as the MTTDL figures. Here, only EMM1 (exact analytical approach) and EMM2A (approximation of EMM2) are used. EMM1A will be compared with EMM1 later in this chapter.

TMM and the MTTDL values of EMM1 and EMM2A models as presented in this thesis with comparable parameters.[*]

Figure B-1 compares TMM with EMM1 and EMM2A with no sector faults as a function of the reliability of the disk unit. The MTTDL values of TMM and EMM1 give the same results while EMM2A has a small error when the disk unit reliability is low. The approximation error is studied later in this chapter.

Figure B-2 compares TMM with EMM1 and EMM2A with no sector faults as a function of the mean time to repair a disk unit. The MTTDL values of TMM and EMM1 give the same results while EMM2A has a similar magnitude error when the repair time is long as in the previous comparison. Again, this is because of the approximation that is used in EMM2A.

Figure B-3 compares TMM with EMM1 and EMM2A with no sector faults as a function of the number of disks in the array. The MTTDL values of TMM, EMM1, and EMM2A give the same results in all three cases with the entire range of the number of disks.

Comparison with sector faults included

Figures B-4, B-5, and B-6 of Appendix B illustrate the comparison of the MTTDL values of TMM and the MTTDL values of EMM1 and EMM2A as presented in this thesis when the sector faults are not ignored.[†]

Figure B-4 compares TMM with EMM1 and EMM2 with sector faults as a function of the reliability of the disk unit. The MTTDL values of TMM provide somewhat poorer MTTDL than EMM1 and EMM2A (with sector fault detection). This is because they all have the same probability of having the first fault in the array (either a sector or a disk unit fault), but EMM1 and EMM2A have lower probability of the second fault because the failure rate in the sector fault states is lower than in the disk unit fault states. On the other hand, MTTDL of EMM1 and EMM2A drops dramatically from the values of TMM if the sector faults are included but not detected. This is well in the line what is expected because due latent faults the system is mainly on the sector fault state.

Figure B-5 compares TMM with EMM1 and EMM2A with sector faults as a function of the mean time to repair a disk unit. Here, both sector and disk unit repair times are varied simultaneously. The MTTDL values of TMM are somewhat worse than those of EMM1 and EMM2A with sector fault detection because of the same reason as above in Figure B-4. MTTDL of

[*] In these three comparisons, sector faults are totally ignored and all faults are disk unit faults.

[†] The disk unit faults of TMM are split in EMM1 and EMM2A into two parts: 50% of the faults are used for disk unit faults and while the another 50% are used for sector faults.

EMM1 and EMM2A without sector fault detection is significantly lower as the reliability is totally dominated by the undetected sector faults.

Figure B-6 compares TMM with EMM1 and EMM2A with sector faults as a function of the number of disks in the array. The MTTDL values of TMM are somewhat worse than those of EMM1 and EMM2A with sector fault detection because of the same reason as above in Figures B-4 and B-5. Respectively, the MTTDL values of EMM1 and EMM2A with no sector fault detection result significantly poorer as the reliability of the array is effected by the undetected sector faults and growing number of disks.

Mission success probabilities

Some of the mission success probabilities of above comparisons are listed in Table 10. These mission success probabilities are based on the default values of the parameters listed in Appendix B. The results in all four cases are almost the same when the same parameters are used. The approximation methods (EMM1A and EMM2A) have a marginal underestimation for the mission success probabilities.

Conclusions of validation of novel reliability models

When EMM1 and EMM2A are compared with TMM the following observations can be made:

- with the same input parameters (i.e., the sector faults ignored), EMM1 provides exactly the same results as TMM;

Table 10. Sample mission success probabilities for TMM, EMM1, EMM1A, and EMM2A (the values of the parameters are listed in third and fourth columns of Table B-1 in Appendix B)

Comparison	Mission	TMM	EMM1	EMM1A	EMM2A
B-3	M1	0.987	0.987	0.987	0.987
B-3	M3	0.961	0.961	0.960	0.960
B-3	M10	0.876	0.875	0.873	0.873
B-6	M1	0.987	0.351 *) 0.990 **)	0.117 *) 0.990 **)	0.117 *) 0.990 **)
B-6	M3	0.961	0.011 *) 0.971 **)	0.002 *) 0.970 **)	0.002 *) 0.970 **)
B-6	M10	0.876	0.000 *) 0.905 **)	0.000 *) 0.905 **)	0.000 *) 0.905 **)
			*) with no sector fault detection	**) with sector fault detection	

- with the same input parameters (i.e., the sector faults ignored), EMM2A provides very good approximation of TMM;
- when the sector faults are included but not detected, EMM1 and EMM2A indicate significantly worse reliability than TMM with no sector faults; and
- when the sector faults are included and detected, EMM1 and EMM2A indicate slightly better reliability than TMM with no sector faults.

This concludes that the novel reliability models comply with the old model since EMM1 provides the same results as TMM with the same input parameters and EMM2A provides a good approximation of TMM.

8.2 Sensitivity analysis

In the sensitivity analysis of the reliability models, the parameters of the EMM1 and EMM2A are studied. In Appendix C, MTTDL of disk arrays is illustrated in various parameter combinations. The sensitivity analysis is divided into three main parts: effect of the number of disks, effect of the failure rates, and effect of the repair rates.

The sensitivity analysis is done so that only two parameters are varied at the same time. The primary parameter is varied from one extreme to another while the second parameter has usually only values: minimum, typical, and maximum. Rest of the parameters are kept in their default values. With this configuration, it is possible to analyze effects of the parameters with a limited set of the combinations.

The following abbreviations are used:

• MTBDF:	Mean Time Between Disk unit Failures;
• MTBSF:	Mean Time Between Sector Faults (in a disk);
• MTTRDF:	Mean Time To Repair Disk unit Failure;
• MTTRSF:	Mean Time To Repair Sector Fault;
• MTBSDF:	Mean Time Between Second Disk unit Failures; and
• MTTOSD:	Mean Time To Order and replace Spare Disk.

8.2.1 Sensitivity to the number of disks

Figures C-1 to C-5 illustrate the reliability effect of the number of disks. Five scenarios are studied in combination with the number of disks in the array:

- disk unit failure rate;

- sector failure rate;
- disk unit repair rate;
- sector repair rate; and
- disk unit repair rate that is relative to the number of disks in the array.

Disk unit failure rate

Figure C-1 illustrates the reliability of the disk array as a function of the number of disks in the array and the disk unit reliability. Regardless of the number of disks in the array, the system MTTDL is improving slightly over one decade when MTBDF improves with one decade (from 200 000 hours to 2 million hours). Similarly, MTTDL drops over one decade when MTBDF drops one decade (from 200 000 hours to 20 000 hours) regardless of the number of disks in the array. The drop is higher as the probability of having second disk fault causes higher probability of data loss while the increase in reliability is limited by the other parameters in the system.

Sector failure rate

Figure C-2 illustrates the reliability of the disk array as a function of the number of disks in the array and the sector reliability. Regardless of the number of disks in the array, the system MTTDL is improving by a factor of three when MTBSF improves by one decade (from 200 000 hours to 2 million hours). Similarly, MTTDL drops almost one decade when MTBSF drops one decade (from 200 000 hours to 20 000 hours) regardless of the number of disks in the array. The reliability change is smaller when MTBSF is increased as the reliability is limited by the other component while the decrease of MTBSF causes the sector failure rate to be the reliability bottleneck.

These results are better than when varying the disk unit reliability because the probability of having a data loss after a sector fault is smaller than after a disk unit fault. After a sector fault, either a disk unit fault or a corresponding sector fault in another disk causes data loss. On the contrary, after a disk unit fault any sector fault or any disk unit fault causes data loss.

Disk unit repair rate

Figure C-3 illustrates the reliability of the disk array as a function of the number of disks in the array and the disk unit repair rate. Regardless of the number of disks in the array, the system MTTDL improves about 50% when the mean disk unit repair time drops from 8 hours to 2 hours. Respectively, the system MTTDL is reduced to one half (quarter) when the mean disk unit repair time is increased from 8 hours to 24 (72) hours regardless of the number of disks in the array.

Sector fault repair rate

Figure C-4 illustrates the reliability of the disk array as a function of the number of disks in the array and the sector fault detection and repair rate*. Regardless of the number of disks in the array, the system MTTDL is improving about 20% (40%) when the sector faults are detected and repaired in 12 hours instead of 24 hours for EMM1 (EMM2A). The system MTTDL drops significantly when the sector fault repair takes a longer time (78, 254, or 824 hours) leading up to one decade worse MTTDL for the whole range of disks.

Disk unit repair rate relative to the number of disks

Figure C-5 illustrates the reliability of the disk arrays as a function of the number of disks in the array when the repair time is related to the number of disks in the array. It is assumed that it takes 15 minutes to read/write an entire disk at normal repair rate (i.e., about 1 MB/s average transfer rate for a disk of 1 GB capacity). In addition, there is the minimum startup delay of 8 (2) hours in EMM1 (EMM2A). When the number of disks in the array is small, there is no significant difference in the system MTTDL with different repair activities as the difference in the repair times is only a few tens of minutes. On the other hand, the effect on the system MTTDL is almost one decade when there are many disks in the array. For example, when there are 100 disks in the array, the difference in the repair time with normal and 10% repair speeds is 225 hours. This explains the significant difference also in the MTTDL values.

Conclusions of the number of disks

Figures C-1 to C-5 illustrate that the relative reliability of the disk arrays (as expressed with MTTDL) is not significantly related to the number of disks in the array. Naturally, the absolute reliability improves when the number of disks decreases, but the relative difference between configurations remains quite the same regardless of the number of disks in the array.

Only, if the disk unit repair time depends on the number of disks (as illustrated in Figure C-5), the number of disks in the array plays a significant role in the reliability estimates.

As for the conclusion, this means that it is possible to use a fixed number of disks in the array (in the future analysis 50 disks is used as a default) and the relative improvement in MTTDL will apply reasonably well also for disk arrays with larger or smaller number of disks.

* The MTTRSF values of 12, 24, 78, 254, and 824 hours represent values that are relative to the scanning and the user access patterns.

8.2.2 Failure rates

Four different failure rates are analyzed: disk unit, sector, second disk unit, and spare disk unit failure rates.

Disk unit failure rate

Figure C-6 illustrates the reliability of the disk array as a function of the reliability of disks and the disk unit repair rate. When MTBDF is over 100 000 hours, MTTDL increases linearly* with MTBDF. Below that, MTTDL decreases faster than MTBDF as the second disk unit fault becomes the reliability bottleneck.

Sector failure rate

Figures C-7 and C-8 illustrate the reliability of the disk array as a function of the reliability of the disk sectors and the sector fault detection rate. When MTBSF is in its normal range (10 000h - 1 000 000h), MTTDL increases but not linearly with MTBSF. With higher MTBSF values, MTTDL is limited by the disk unit failure rate (two disk units failing one after another). Similarly, MTTDL has lower limit as the probability of having two sector faults at the corresponding sectors is so marginal that also the lower bound of the reliability is eventually limited with the disk unit reliability (a disk unit fault is needed after a sector fault).

If the disk unit reliability were not the limiting factor, MTTDL in Figures C-7 and C-8 would behave the same way as in Figure C-6.

Second disk failure rate

Figure C-9 illustrates the reliability of the disk array as a function of the possibility to have interrelated disk faults (second disk fault probability is higher than the first one) and the reliability of the disk units. The system MTTDL is almost constant when the second disk unit failure rate is about the same as or smaller than the first disk unit failure rate. When the second disk unit failure rate increases (probability of having related disk unit faults), MTTDL approaches the case where no redundancy is provided.†

* Here, the “linear increase” means that it is a straight line in the logarithmic scale.

† If the second disk fault happens immediately after the first one, then the reliability is same as in a RAID-0 type array where the first disk fault causes also the data loss. For example, if MTBDF is 200 000 hours and there are 50 disks, then MTTDL is about 4 000 hours.

Spare disk failure rate

Figure C-10 illustrates the reliability of the disk array as a function of spare disk reliability. The disk array reliability seems to be almost independent of the spare disk reliability. Only, a marginal drop is noticeable when the spare disk reliability decreases. This is in the line with the observation made by Gibson that need for second spare disk is marginal [Gibson 1991]

Conclusions of failure rates

Figures C-6 to C-10 illustrate that the relative reliability of the disk array (as expressed with MTTDL) is not significantly related to failure rates when the failure rates are in the conventional range of modern disks (100 000 - 1 million hours). Naturally, the absolute reliability improves when the failure rate decreases, but the relative difference between configurations remains quite the same regardless of the failure rate. Hence, 200 000 hours can be used for both MTBSF and MTBDF in the further analysis.

8.2.3 Repair rates

Three different repair rates are analyzed: disk unit, sector, and spare disk unit repair rates.

Disk unit fault repair rate

Figure C-11 illustrates the reliability of the disk array as a function of mean disk unit repair time and reliability of the disks. When MTTRDF is in its practical range (from few hours to few hundred hours), MTTDL depends linearly on the mean repair time. The upper bound of MTTDL is limited by the other components in the system (such as sector faults). When the mean disk unit repair time approaches infinity, the lower bound of MTTDL is also limited (i.e., the array acts like a system that has no spare disks, but can tolerate one fault).

There is significant difference in MTTDL of EMM1 and EMM2A when the repair time is short and MTBDF is 20 000 hours. This is because it takes longer time (at least 24 hours) in EMM2A to install a new spare disk in the array and the probability of having a second faulty disk soon after the first one is high.

Sector fault repair rate

Figure C-12 illustrates the reliability of the disk array as a function of mean sector fault repair time and reliability of the disks. When MTTRSF is in its practical range (from a few hours to a few hundred hours), MTTDL depends linearly on the mean sector repair time. The upper bound of MTTDL is limited by the other components in the system (such as disk unit reliability). When the mean sector repair time approaches infinity, the lower bound of MTTDL is also limited (i.e., when the sector faults are not detected, the latent sector fault and a disk unit fault cause the data loss).

Spare disk fault repair rate

Figure C-13 illustrates the reliability of the disk array as a function of the spare disk fault detection and repair rate. When this rate is in its practical range (from few hours to few hundred hours), MTTDL is quite independent of the spare disk fault detection and repair rate. Only, when the disk reliability is low or the repair takes hundreds of hours, the repair rate plays a significant role.

Figure C-14 illustrates the reliability of the disk array as a function of the spare disk replacement rate. When this rate is in its practical range (from few hours to hundred hours), MTTDL is quite independent of the spare disk repair rate. Only, when the disk reliability is low or the repair takes hundreds of hours, the repair rate plays a significant role.

Conclusions of repair rates

Figures C-11 to C-14 illustrate that the relative reliability of the disk arrays (as expressed with MTTDL) is not significantly related with repair rates when the repair rates are in the conventional range of modern disks (few hours to few hundred of hours). Naturally, the absolute reliability improves when the repair rate increases, but the relative difference between configurations remains quite the same regardless of the repair rate. Thus, repair times from few hours to hundred hours can be used in the future analysis with no risk of unstability of the equations.

8.3 Accuracy of approximations

In the accuracy analysis, the approximation methods of EMM1A and EMM2A are compared with the analytical approach EMM1. In Appendix D, MTTDL of disk arrays is illustrated for various parameter combinations. The accuracy analysis is done by varying the main parameters of the disk array: the number of disks, disk unit failure rate, sector failure rate, second disk failure rate, disk unit repair rate, and sector repair rate.

Number of disks in the array

Figure D-1 illustrates the accuracy of the approximation of EMM1A and EMM2A when compared with EMM1 as a function of the number of disks in the array. Both approximations provide very accurate results. When the number of disks is varied from one to 100 disks and the disk repair time is varied between 8 to 24 hours while the other parameters are in their default values, the maximum error (6%) in this figure is achieved when the number of disks is 100 and MTTRDF is 72 hours. In most cases, the error is less than 1%.

Disk unit failure rate

Figure D-2 illustrates the accuracy of the approximation of EMM1A and EMM2A when compared with EMM1 as a function of disk unit reliability. Both approximations provide very accurate results. The maximum error (22%) in this figure is achieved when MTBDF is 10 000 hours and MTTRDF is 72 hours. When MTBDF is better than 100 000 hours, the error is less than 5% and in most cases less than 1%.

Sector failure rate

Figure D-3 illustrates the accuracy of the approximation of EMM1A and EMM2A when compared with EMM1 as a function of sector reliability. Both approximations provide quite accurate results when the sector reliability is in its practical range (MTBSF is greater than 10 000 hours). Then, the maximum error is about 7% and in most cases less than 1%. When MTBSF is below 10 000 hours, over 30% error in MTTDL is experienced. This is because the failure rates in the array are no longer significantly smaller than repair rates.[*]

Second disk unit failure rate

Figure D-4 illustrates the accuracy of the approximation of EMM1A and EMM2A when compared with EMM1 as a function of second disk unit failure probability. Both approximations provide accurate results only when the MTBSDF is over 10 000 hours. Then, the maximum error is about 10% and in most cases less than 1%. When MTBSDF decreases below 10 000 hours, the error grows dramatically. The reason for this is exactly the same as above in Figure D-3. In some cases, the disk array failure rate is much larger than the repair rate which contradicts the assumptions of the approximation.

Disk unit fault repair rate

Figure D-5 illustrates the accuracy of the approximation of EMM1A and EMM2A when compared with EMM1 as a function of disk unit repair rate. When MTBDF is at least 100 000 hours and the mean repair time is less than few hundred hours, both approximations provide quite accurate results. Then, the maximum error is 10% and in most cases less than 1%. When the repair time is

[*] For example, when MTBSF is 10 000 hours, MTTRSF is 78 hours, and there are 50 disks in the array, the repair rate is only about 2.5 times larger than the failure rate.

long or disks unreliable, significant error results. Again, the approximation is erroneous when the repair rates are no longer significantly larger than the failure rates.

Sector fault repair rate

Figure D-6 illustrates the accuracy of the approximation of EMM1A and EMM2A when compared with EMM1 as a function of sector repair rate. Both approximations provide quite accurate results when the sector repair time is less than a few hundred hours. Then, the maximum error is 10% and in most cases less than 1%. When the repair time is long, significant error results. Again, the approximation is erroneous when the repair rates are no longer significantly larger than the failure rates.

Conclusions of approximation models

Figures D-1 to D-6 illustrate that the approximation models provide accurate results when the repair rates are significantly higher than the failure rates. In practical systems this is usually the case, as the mean time between failures is typically over 100 000 hours and the mean time to repair less than a hundred hours, while the number of disks is less than a hundred. Then, the failure rate is at most one tenth of the repair rate.

Both EMM1A and EMM2A provide the same results. It can be therefore assumed that both models approximate reasonably well the actual reliability model.

The only case, when it is not possible to use the approximation, is the study of interrelated disk unit faults as shown in Figure D-4. However, this is not such a serious problem because this can be studied analytically as shown later in this chapter in Scenario 5.

In all cases when there is an error in comparison with EMM1, both approximation methods, EMM1A and EMM2A, underestimate the reliability. This agrees with the prediction in Chapter 6.

8.4 Reliability scenarios

Table 11 lists nine different comparison scenarios. The columns in the table indicate whether the emphasis is more on the repair process, the sector faults, or the scanning algorithm. These scenarios illustrate various aspects of reliability related to the sector faults or the new reliability models. The main objective here is to analyze the scenarios to achieve deeper understanding of the behavior of the disk arrays.

The default parameters for the scenarios are listed in Table 12. These values are used in all scenarios unless otherwise stated.

Effect of the disk access patterns

The analysis uses those user access patterns that are listed in Chapter 6, namely Uniform, Single-80/20, Double-80/20, and Triple-80/20. Actually, the access patterns are relevant here only for specifying the sector fault detection rate. Beside the user access pattern, the relative activity of the user read requests compared with the scanning requests specifies the relative sector fault detection rate.

Disk array configurations

The analysis is mainly done for a RAID-5 disk array and in some cases also a RAID-1 disk array is used. From the point of view of the reliability analysis, the main differences of these two array configurations are the different repair times and the number of disks involved. In practice, the

Table 11. Different scenarios for the reliability analysis

Scenario	Repair process method	Sector faults	Scanning algorithm
SC1: *Effect of the sector faults*			
a)	normal	ignored	ignored
b)	normal	included	ignored
c)	normal	included	only by user requests
SC2: *Effect of scanning algorithm*			
a)	normal	included	ignored
b)	normal	included	included
SC3: *Delayed disk repair*			
a)	normal	included	ignored
b)	delayed	included	ignored
SC4: *Delayed disk repair with scanning algorithm*			
a)	normal	included	ignored
b)	delayed	included	included
SC5: Related faults	normal	included	included
SC6: *Hot swap vs. hot spare*	normal	included	included
SC7: *Effect of the spare disk*	normal	included	included
SC8: *Percentage of sector faults*	normal	included	included
SC9: *RAID-1 vs. RAID-5*	normal	included	included

Table 12. Default parameters for the scenarios

Parameter	Default values in models EMM1, EMM1A, and EMM2A
D	50
S	1 000 000
$1/\lambda_d$	200 000h
$1/\lambda_{df}$	200 000h
$1/\lambda_s$	200 000h * S
$1/\lambda_{sd}$	2 000 000h
$1/\mu_d$	8h
$1/\mu_s$	24h
$1/\mu_{sd}$	44h
$1/\mu_{dr}$	32h

RAID-1 array with one pair of mirrored disks has equal reliability as RAID-5 with two disks. Hence, the RAID-1 array can be treated as a special case of RAID-5. Hence, EMM1, EMM1A, and EMM2A can be used for both RAID-1 and RAID-5 disk array configurations. Actually, the reliability models are also applicable for RAID-3 and RAID-4 disk arrays since RAID-3, RAID-4, and RAID-5 have identical failure models.

In the case of a larger RAID-1 array (such as ten pairs of mirrored disks), the results can be achieved from the single pair of disks and treating the pairs as independent units. It is infeasible to have a hot spare disk for every pair. Instead, a common spare disk is used. This means that only one such pair could be repaired at any time. This is considered to cause only minor error as it has been stated that even with a disk array of 32 disks there is very little use for more than one on-line spare disk [Gibson 1991].

8.4.1 Scenario 1: Effect of sector faults

The first scenario compares the effect of sector faults in general (i.e., how much worse is the disk array reliability if the sector faults are included, but not detected efficiently when compared with the conventional disk array model where sector faults are totally ignored?). The results are illustrated in Figure 26. To make the reliability figures comparable, it is assumed that 50% of the faults in EMM1 are sector faults and the remaining 50% are disk unit faults. This corresponds to the view of the distribution of the faults between sector and disk faults in modern disks [Räsänen 1996]. Thus, the total failure rate of each disk is the same, but cause is different.

When the sector faults are included and detected slowly, the disk array reliability (as expressed with MTTDL) lags significantly behind the predicted MTTDL of TMM. This is because the sector faults can remain latent for a long time in the array and just one disk unit fault is needed to cause data loss. The lower limit of the reliability is achieved when the sector faults are not detected at all. In that case, MTTDL is only about 2% of MTTDL that is predicted by TMM when the average disk lifetime is about 200 000 hours.

On the other hand, when the sector faults are detected faster, the reliability of the disk array is at the same level as what TMM predicts. Actually, even better reliability can be achieved as it is shown in Figure 26 if the average sector fault detection time is 12 or 24 hours. The reasoning for this was given earlier in this chapter (8.1) when the new reliability models were validated.

Conclusions of Scenario 1

This scenario points out two issues. First, the reliability drop is dramatic if the sector faults are included but not detected. However, the same level of reliability can be achieved even when the sector faults are included if they are detected efficiently. Second, disk requests of a typical user are not very suitable for detecting latent sector faults because of their uneven access patterns. Instead, efficient scanning algorithms that can scan the entire disk in matter of hours (typically once or twice

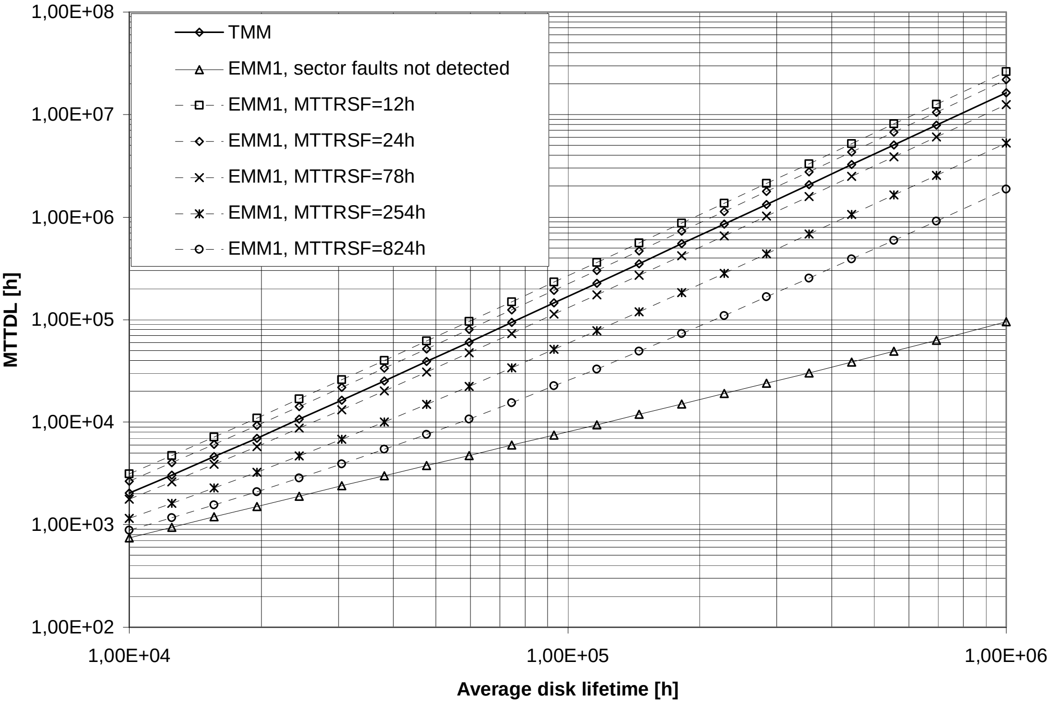

Figure 26. Effect of the sector faults

per day) can provide good latent fault detection and also good reliability.

8.4.2 Scenario 2: Effect of scanning algorithm

The second scenario studies the effect of the scanning algorithm and its activity. The efficiency of the scanning algorithm depends on the following aspects:

- access patterns of the user disk requests;
- relative activity of the scanning algorithm (compared with the user disk requests); and
- absolute activity of the scanning algorithm.

An example for a situation when the scanning algorithm would provide significant improvement on the latent fault detection is when the user access pattern is uneven (such as Triple-80/20 distribution) and the scanning algorithm is running at the same order of magnitude as the user disk requests. Then, sector fault detection rate is significantly higher with the scanning algorithm. The ratio of the sector fault detection of the scanning algorithm and the user disk access pattern is

$$\frac{\mu_{s,scan}}{\mu_{s,user}} = \frac{1/\bar{E}_{scan}}{1/\bar{E}_{Triple-80/20}} = \frac{1/0.5}{1/34.33} = 68.66\,. \tag{103}$$

The results of fast scanning can be seen in Figure 26 of Scenario 1. The reliability of the disk array can be improved significantly (MTTDL can improve ten fold) when an efficient scanning algorithm is used.

On the other hand, if the scanning algorithm has significantly lower activity than the user requests and the user’s disk access pattern is distributed evenly, then the effect of the scanning algorithm in quite minimal. For example, if Uniform access pattern is used and the scanning activity is only 5% of the user read requests, then the ratio of the fault detection of the scanning algorithm and the user disk access pattern is

$$\frac{\mu_{s,scan}}{\mu_{s,user}} = \frac{scan\_activity \times 1/\bar{E}_{scan}}{1/\bar{E}_{Uniform}} = \frac{0.05 \times 1/0.5}{1/1} = 0.1\,. \tag{104}$$

Then, the scanning algorithm would improve the sector fault detection rate by 10%. This would have only a marginal effect on the reliability.

The practical ratio of the fault detection of the scanning algorithm and the user access pattern is somewhere in between. For example, 5% scanning algorithm and Triple-80/20 access pattern leads to ratio

$$\frac{\mu_{s,scan}}{\mu_{s,user}}=\frac{scan\_activity\times {}^{1}\!/\!_{\bar{E}_{scan}}}{{}^{1}\!/\!_{\bar{E}_{Triple-80/20}}}=\frac{0.05\times {}^{1}\!/\!_{0.5}}{{}^{1}\!/\!_{34.33}}=3.433. \tag{105}$$

This would mean that the scanning algorithm can still improve MTTDL by 100% as shown by Figure 26.

<u>RAID-5 parity sectors</u>

An important point in the RAID-5 array architecture is that if the array is used only for reading data and never (or hardly ever) data is written into the array, there is no need to access the parity sectors.* If write operations are also done into the disk array, they can either detect the latent sector faults or mask them.

If there is a latent sector fault in a parity area of the RAID-5 array, the user disk read requests would never detect that. On the other hand, the scanning algorithm has no problem detecting sector faults in the parity areas as it can access information on the disk regardless of the array configuration.

Depending on the size of the array, a certain percentage of the sectors will be unaccessed by the user requests. For example, if the RAID-5 array consists of 10 (50) disks, 10% (2%) of the disk space is inaccessible by user read requests because that space is used for storing parity information. Those sectors can be checked non-destructively only by the scanning algorithm.

<u>Conclusions of Scenario 2</u>

This scenario shows clearly that the proposed scanning algorithm can improve the reliability of the disk array in two ways. First, the scanning algorithm can also access sectors (such as RAID-5 parity sectors) that are not accessible by the normal user read requests. Second, if the user access pattern is uneven, the scanning algorithm can significantly expedite the latent sector fault detection even when the activity of the scanning algorithm is low.

8.4.3 Scenario 3: Delayed disk unit repair process

The third scenario compares the normal disk array repair algorithm with a delayed repair

* The parity sectors are conventionally read for two reasons: when recovering after a disk failure or when writing a small block of data. A small write can be converted to two reads and two writes as described in Chapter 4.

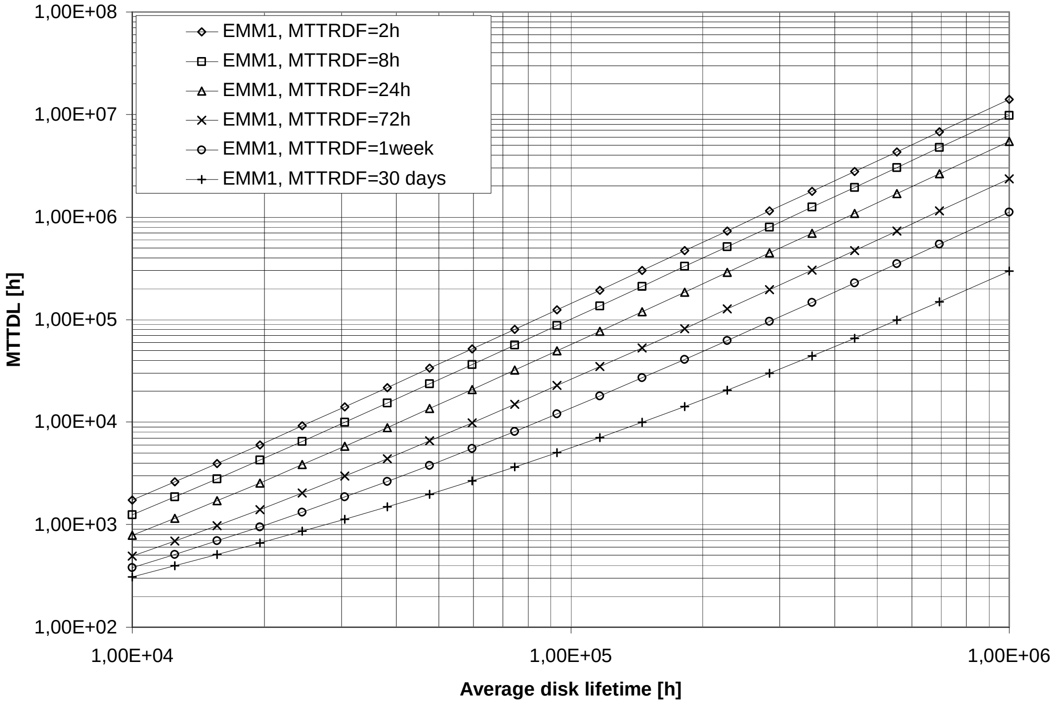

Figure 27. MTTDL in Scenario 3 as a function of average repair time and average disk lifetime

algorithm where either the repair process is slowed down or it is postponed until later for a more suitable time. Figures 27 and 28 illustrate the effect of the delayed disk repair process as a function of average disk lifetime and the number of disks in the array.

Both figures show that by delaying the repair process from 2 (8) hours to 8 (24) hours, the reliability (as expressed with MTTDL) drops about 30% (50%) when MTBDF is 200 000h. The mission success probabilities are illustrated in Table 13. There is no significant impact on reliability if the repair time is delayed from two hours to eight hours. If the repair time is further delayed to 24 hours, the ten years mission success probability starts to degrade significantly.

The main benefit of delaying the repair time is the possibility to reduce the performance degradation during the repair process. For example, if the repair time is extended from two to eight hours, the load caused by the repair process is reduced by 75% as the repair time is four times longer. This can be done if there is a hot spare disk in the array that can be used for starting the repair immediately but slowly.

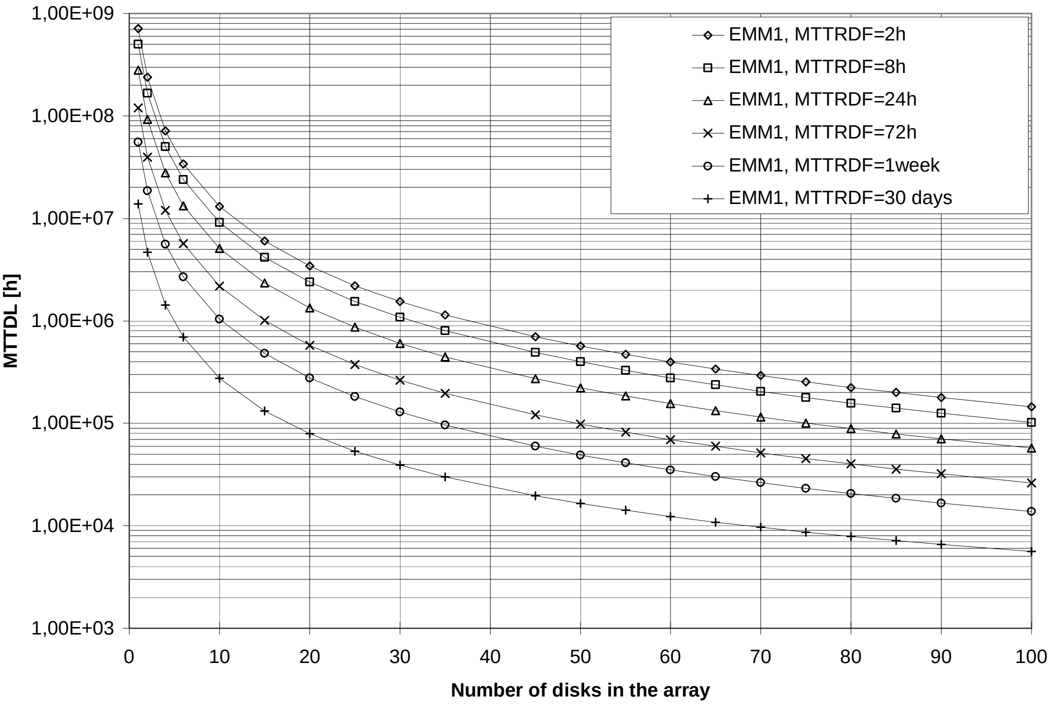

Figure 28. MTTDL in Scenario 3 as a function of average repair time and the number of disks in the array

Conclusions of Scenario 3

This scenario shows that it is possible to delay the repair process at the expense of reduced reliability. This is needed if, for example, the performance requirements must be met also during the repair time. This analysis assumes that the faults are not related and therefore the probability of having a second fault just after the first one is typically low.

Related faults are studied further in Scenario 5.

8.4.4 Scenario 4: Effect of combined scanning algorithm and delayed repair

The fourth scenario compares the combined scanning algorithm and delayed repair process. In

Table 13. Mission success probabilities of Scenario 3

Average disk repair time	M1 (1 year)	M3 (3 years)	M10 (10 years)
2h	98.8%	96.5%	88.7%
8h	98.3%	95.0%	84.3%
24h	97.0%	91.3%	73.7%

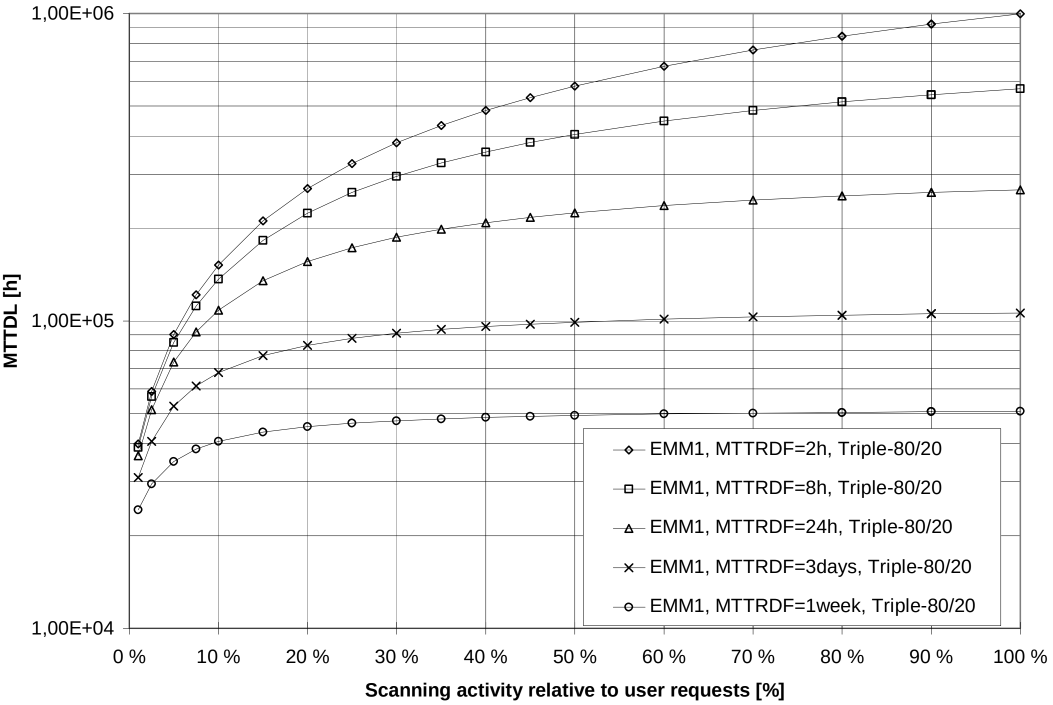

Figure 29. MTTDL as a function of relative activity of the scanning algorithm

Figure 29, reliability of a disk array is illustrated as a function of the activity of the scanning algorithm relative to the user requests. In this figure, it is assumed that user access pattern is Triple-80/20 and the scanning algorithm is adjusted relative to it.

The effect of the combined scanning algorithm and the delayed repair can be illustrated with the following two examples:

1. If the average disk repair time is two hours and the relative activity of the scanning algorithm is 5%, then MTTDL of the disk array is, according to Figure 29, 90 000 hours. However, the same reliability can be achieved:

- using the average disk repair time of eight hours and the scanning activity about 5.5%;
- using the average disk repair time of 24 hours and the scanning activity 7.5%; or
- using the average disk repair time of 72 hours and the scanning activity 30%.

2. If the average disk repair time is two hours and the relative activity of the scanning algorithm is 20%, then MTTDL of the disk array is, according to Figure 29, 270 000 hours. However, the same reliability can be achieved:

- using the average disk repair time of eight hours and the scanning activity about 25%; or
- using the average disk repair time of 24 hours and the scanning activity 100% leaving no capacity for user disk requests.

In the second example, the latter alternative is not feasible as the additional load caused by the scanning algorithm is very likely much higher than what the repair process would generate (at least when compared with the former alternative where the repair process is spread over eight hours, but the scanning algorithm is increased from 20% to 25%. Hence, this is a typical optimization problem.

Conclusions of Scenario 4

This scenario points out that it is possible to delay the disk repair algorithm and compensate for the decreased reliability by increasing the activity of the scanning algorithm to detect the latent faults. The result is that the reliability can be kept at the same level while the performance degradation due to the high speed disk recovery process can be eliminated.

8.4.5 Scenario 5: Effect of related faults

The fifth scenario studies the effect of related faults in the disk array. For example, if the disks are coming from the same manufacturing batch or if they are located in the same cabinet, they are more likely to get faulty at the same time. Thus, their faults are not independent anymore.

Figure 30 illustrates the effect of related faults. If the faults are independent, it is very unlikely that the second disk should fail soon after the first one. Then, the risk of having data loss is quite

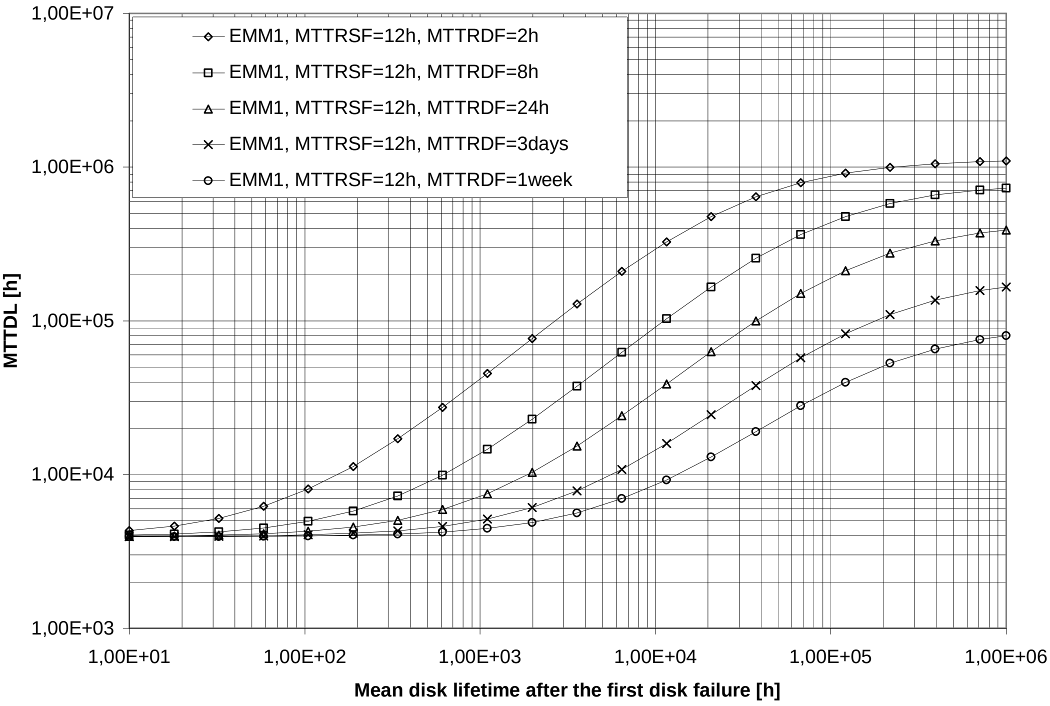

Figure 30. Effect of the related disk unit faults as a function of second disk unit failure rate

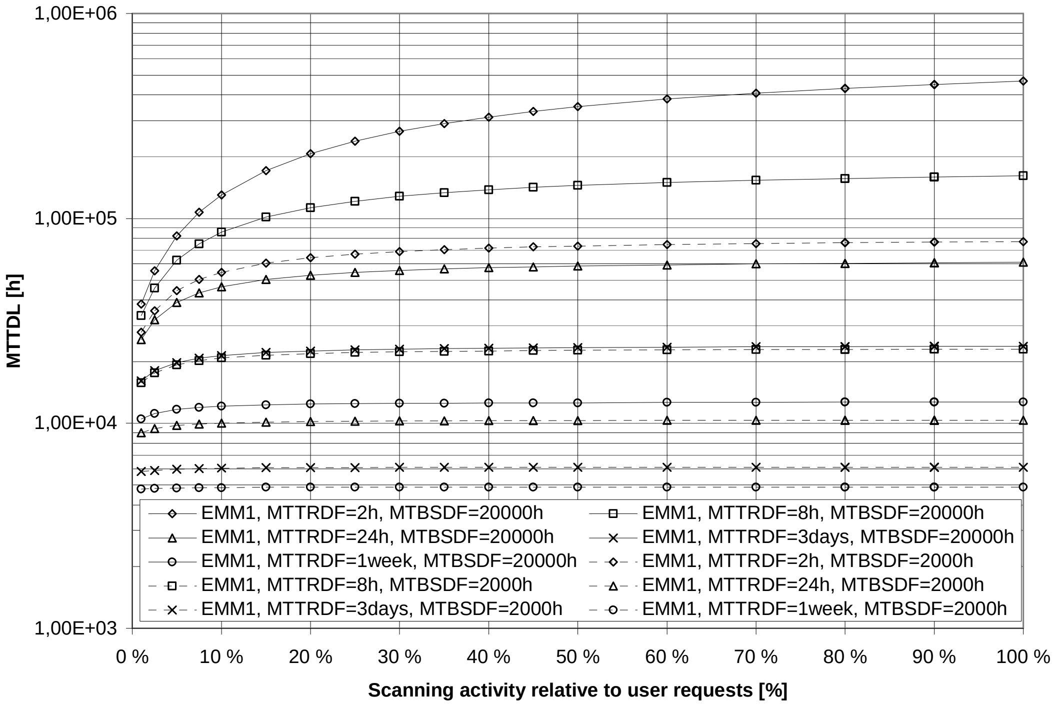

Figure 31. Effect of the related disk unit faults as a function of the scanning algorithm activity

small. On the other hand, if the faults are related, there is a significantly higher probability of having soon a second fault that causes data loss. As shown in Figure 30, drop in the reliability is much higher than in Figure 27 of Scenario 3. Here, the reliability drops to one third of MTTDL if the disk repair time is delayed from two hours to eight or from eight to 24 hours.

An even more dramatic effect on the reliability can be seen in Figure 31. In Scenario 4 it was shown that by combining delayed repair and expedited scanning algorithms it is possible to maintain the same level of reliability. If the average disk unit lifetime after the first disk has failed is 20 000 hours or 2 000 hours, it is more difficult to compensate for the delayed repair with expedited scanning algorithm. Especially in the latter case, the scanning algorithm is no longer capable of regaining the reliability drop that is caused by the delayed disk repair process.

The lower limit of the reliability can be achieved when the second disk is assumed to fail immediately after the first disk. Then, estimated MTTDL of the disk array can be expressed as the reliability of the single disk divided by the number of disks in the array. For example, in the array of 50 disks where the average disk unit lifetime is 200 000 hours, MTTDL of the array is only 4000 hours.*

* When the second disk fails immediately after the first one, the disk array then resembles an array that has no redundancy, just like a RAID-0 array.

The situation of the related faults can be even more serious than expressed above. The reliability can drop even further if even the first disk unit failure rate is much higher than expected due to, for example, increased temperature in the disk cabinet.

Conclusions of Scenario 5

Scenario 4 showed a possibility to compensate for the obstructed repair process by increasing the sector fault detection to maintain the reliability level. This is not the case when the possibility to have related faults is considered as shown in Scenario 5. Hence, the disk repair process should be completed as quickly as possible to minimize the risk of interrelated faults that would dramatically drop the reliability.

The worst case of related faults is when the first failing disk causes the second disk to fail immediately. Then, the D+1 reliable disk array turns into an array of D+1 disks with no redundancy (just like RAID-0). The reliability of the disk array can decrease even further, if the reason for the related faults causes extended degradation on the disk unit from the beginning.

8.4.6 Scenario 6: Effect of hot swap or hot spare disks

The sixth scenario studies the effect of the spare disk replacement time. In Figure 32, the

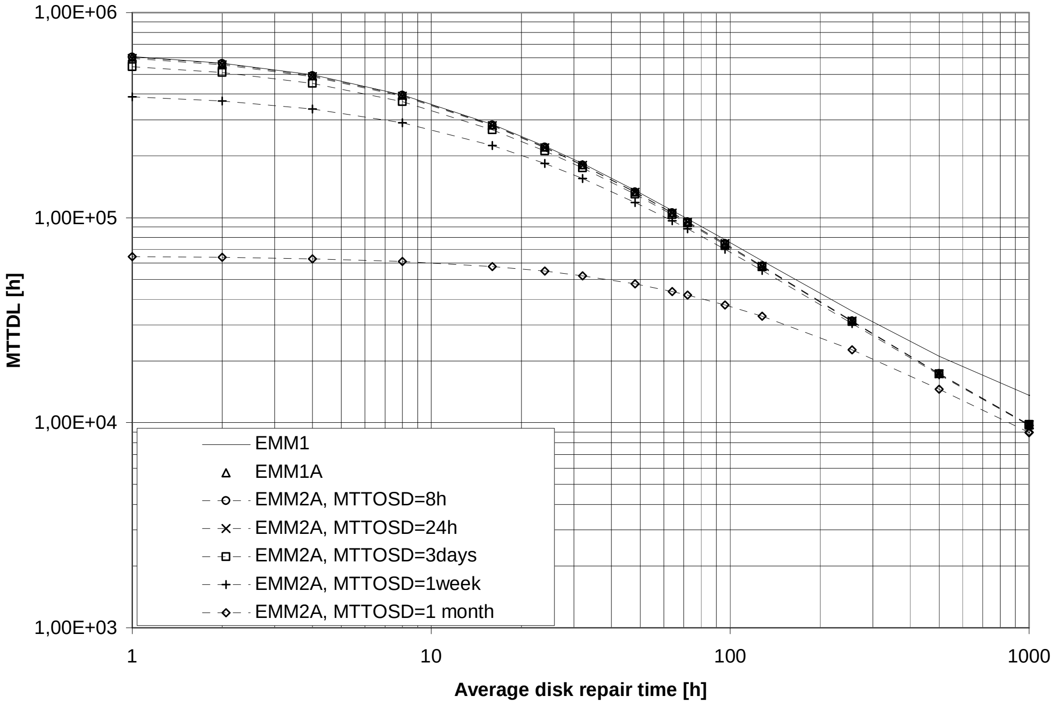

Figure 32. MTTDL as a function of spare disk replacement time

reliability of the disk array is illustrated as a function of spare disk replacement time. Here, EMM2A is compared with EMM1 and EMM1A. Figure 32 shows that the reliability is almost the same if the spare disk replacement time is 8, 24, or 72 hours. This means, that if the disk array has an online spare disk, the time to order another spare disk after a disk failure is not so important. Only, if the spare disk ordering takes one week or one month, then there is significant effect on the disk array reliability.

More significant effects can be found if the disk unit faults are related. For example, Figure 33 illustrates the case when the disk unit faults are related and the disk unit failure rate is ten times higher after the first disk failure. In this case, the reliability drops more when the spare disk order and replacement time increases.

Conclusions of Scenario 6

This scenario shows the benefit of hot spare disks. When the disk array is repaired with a hot spare disk, it is not so critical when a new spare disk is added to the array. If the new spare disk is added within 24 hours there is no significant effect on the reliability even if the disk unit faults are related.

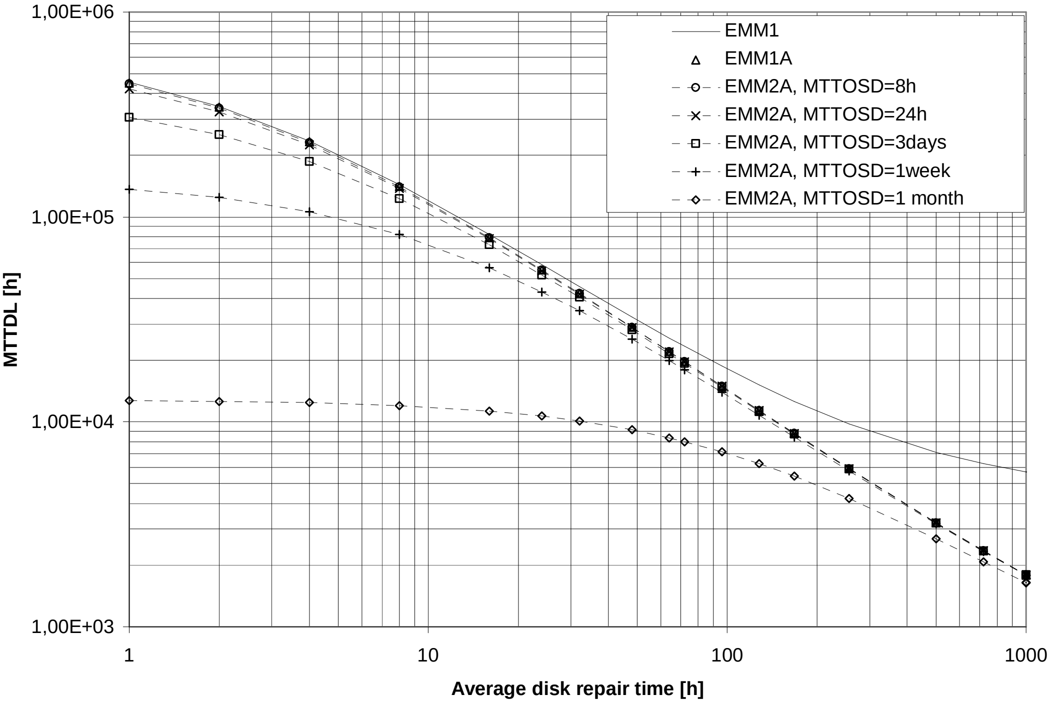

Figure 33. MTTDL as a function of spare disk replacement time when MTBSDF is 20 000 hours

8.4.7 Scenario 7: Effect of spare disk reliability

The seventh scenario illustrates the effect of the spare disk reliability. In Figure 34, the reliability of the disk array is represented as a function of spare disk reliability. As it can be seen, the reliability of the spare disk plays an insignificant role in the total reliability. Only a marginal drop is noticed when the spare disk reliability decreases.

Conclusions of Scenario 7

This scenario together with Scenario 6 show that it is possible to use the simpler model (EMM1) of the disk array with no significant error when the disk array repair time is the same and the new spare disk is assumed to be added soon enough (e.g., within 24 hours after the disk failure).

This is a significant observation as in the next chapter, where the performability is studied, the analysis can be done using only the analytical approach (EMM1) to analyze both EMM1 and EMM2 models.

8.4.8 Scenario 8: Effect of the percentage of sector faults

The eighth scenario studies the effect of the relative amount of sector and disk unit faults in the

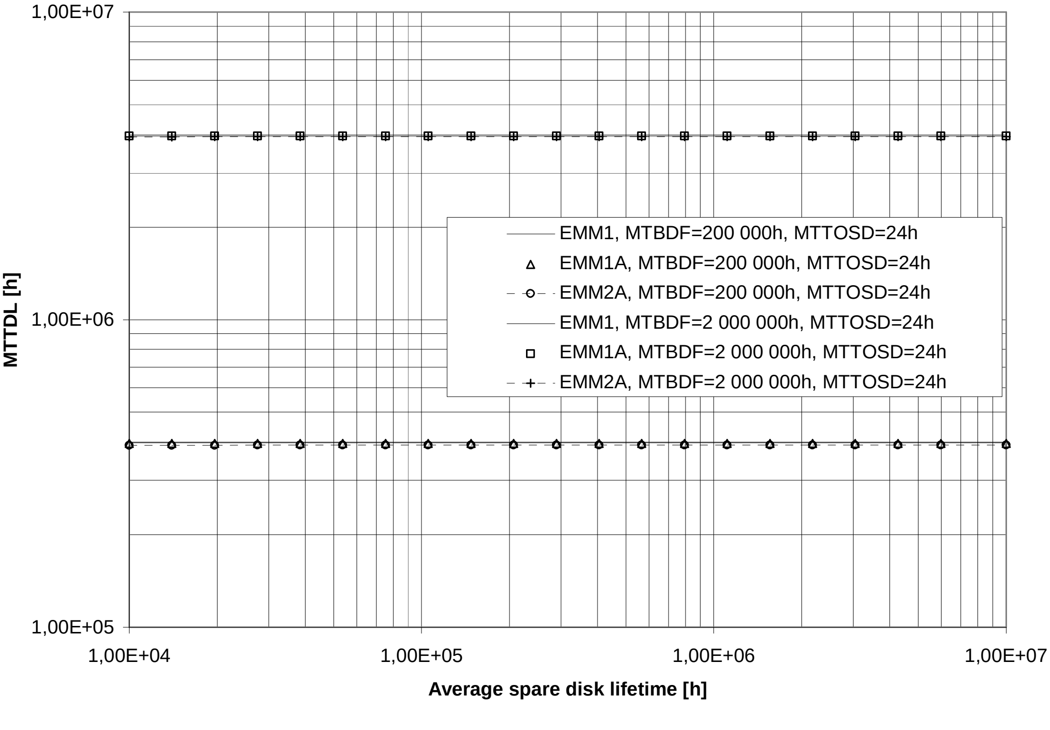

Figure 34, MTTDL as a function of spare disk reliability array. Figure 35 illustrates the disk array reliability as a function of the percentage of sector faults of all faults. The reliability of the disk array does not change very much if the percentage of sector faults is changed from the default value (50%) down to 0%. The effect is only 25% (15%) for an array that has the average disk unit fault repair time of 24h (8h) and the average sector fault repair time of 24h (12h).

On the other hand, if the percentage of sector faults increases significantly, the reliability of the disk array also increases radically. This is because having two sector faults at the corresponding sectors is not very probable as it has been described earlier in this chapter. Eventually, if all faults were sector faults, the disk array reliability would be extremely high as it would require having two sector faults in the corresponding sectors for a data loss.

Conclusions of Scenario 8

This scenario shows that it is quite safe to assume that 50% of all faults in a disk are sector faults, when the disk unit and sector repair times are comparable. If the percentage of sector faults is in the range of 0% to 50%, the effect on the disk array reliability is around 15-25%.

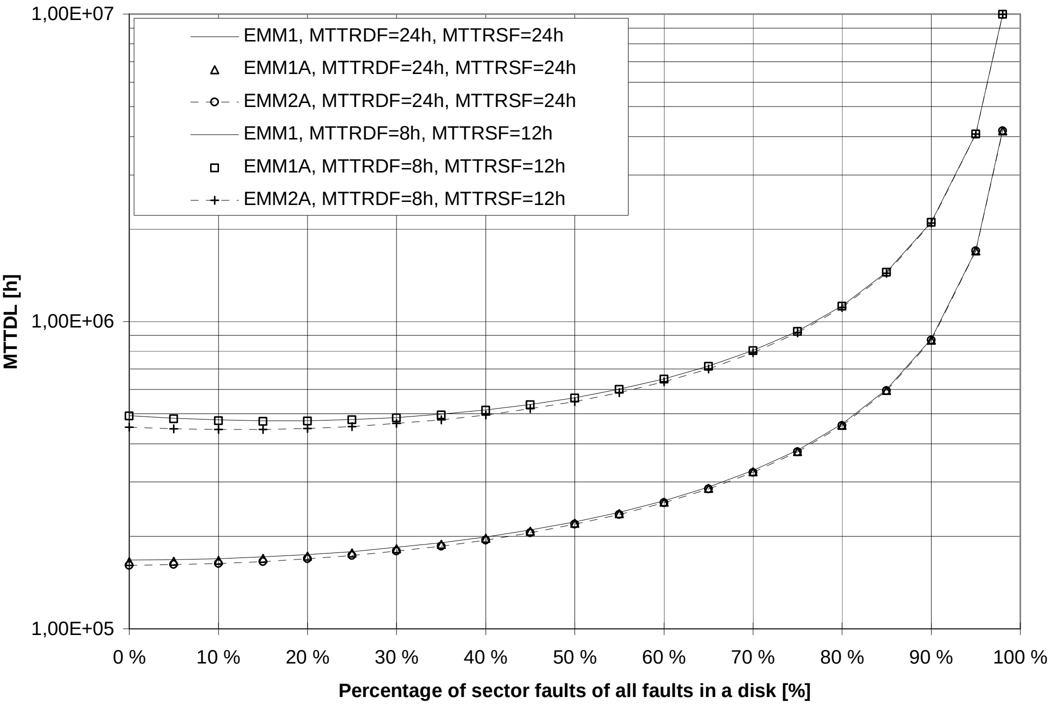

Figure 35. MTTDL as a function of the percentage of sector faults in the array

8.4.9 Scenario 9: RAID-1 vs. RAID-5

The ninth scenario compares two different array architectures: RAID-1 and RAID-5. The goal for this comparison is to compare the reliability of an array when the array is either built using D+1 redundancy (RAID-5) or D+D redundancy (RAID-1). Figure 36 illustrates the reliability of the disk array as a function of the number of data disks in the array. It is clearly seen that RAID-1 provides significantly higher reliability than RAID-5 with the same number of data disks. However, this is done at the expense of the increased cost of the array as almost double the number of disks is needed.

One additional point that increases the difference between RAID-1 and RAID-5 reliability estimations is the repair time. As in RAID-5, all disks are involved with the disk array repair, the repair time is much longer and also the crippled array performance is lower. If the RAID-5 with 50 data disks can be repaired in 8 (24) hours, then RAID-1 with 50 disks can be easily repaired respectively in 2 (8) hours because in the latter case basically only two disks are involved in the repair process, not the entire array. Hence, RAID-1 can provide 30 times higher MTTDL than RAID-5 with 50 data disks while the same repair time would have provided only 24 times higher MTTDL.

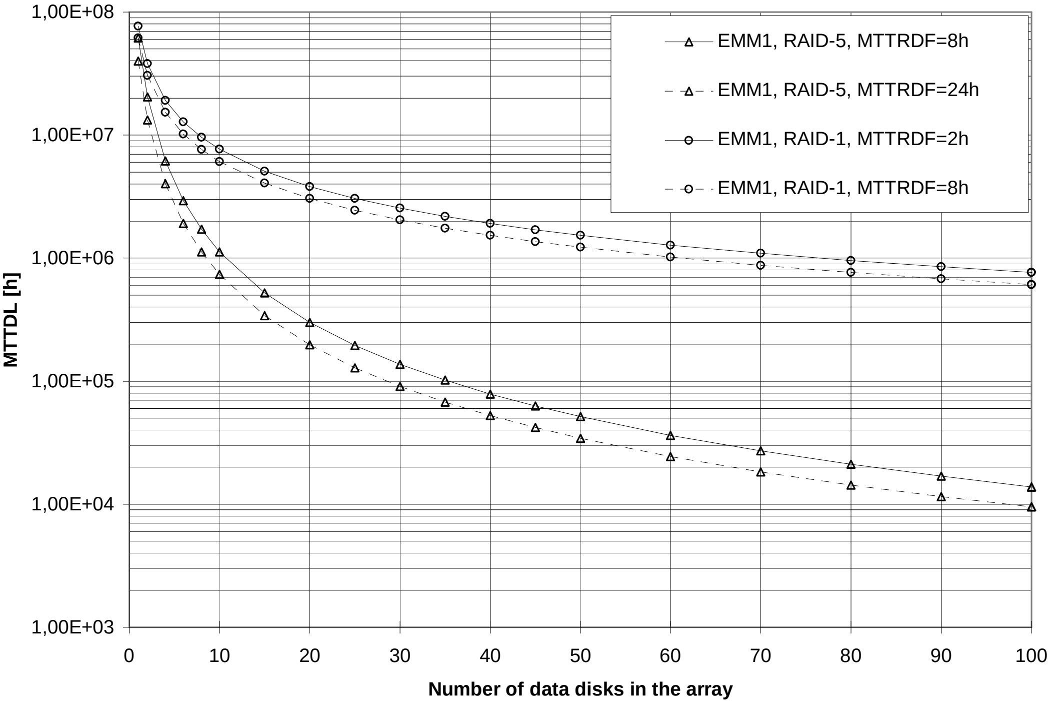

Figure 36. Comparison of RAID-1 and RAID-5 disk arrays

Conclusions of Scenario 9

This scenario shows that RAID-1 can provide significantly higher reliability than a RAID-5 array, but at the cost of an increased number of disks. Hence, there are almost double the number of failed disks in RAID-1, but because of smaller number of disks in the disk group and the faster repair process, the probability of a data loss is still smaller.

8.4.10 Summary of scenarios

In the following list, a summary of conclusions for the scenarios is collected.

- Reliability of the disk array depends heavily on the latent sector faults and their detection rate.
- Typical user disk accesses are not very good in detecting latent sector faults because of their uneven access patterns. User access patterns are not capable of detecting all latent faults (such as faults in RAID-5 parity blocks).
- A special scanning algorithm should be used for detecting latent sector faults. The proposed scanning algorithm can improve the reliability significantly.
- Performance degradation of a crippled disk array can be reduced by delaying or obstructing the repair process at the expense of reduced reliability. Within a certain limits, the reliability can be regained by expediting the latent (sector) fault detection.
- Related faults decrease dramatically the reliability of the disk array. In the worst case, the reliability of RAID-5 array with D+1 disks is as bad as the reliability of RAID-0 with D+1 disk (having no redundancy).
- A hot spare disk provides better reliability than a hot swap disk. This is mainly because the repair process can start earlier. It is not so important to add a new spare disk into the disk array after a disk failure. There is no big difference if the new spare disk is added immediately or on the average in 8 or 24 hours.
- It is possible to model a disk array with a hot spare disk with EMM1, if the repair time is as fast as with the hot spare disk case and the new spare disk is added within 24 hours after the disk failure. Then, the same analytical approach (EMM1) can be used for hot swap and hot spare arrays in Chapter 9.
- When the percentage of sector faults (of all faults in the disk) is in the range 0-50%, the analysis provides similar results for the reliability. Thus, the analysis made in Scenarios 1-7 and in Scenario 9 should be quite independent of the percentage of sector faults. Only, when majority of the faults is sector faults, then there is a significant effect on the

reliability.

- RAID-1 provides significantly better reliability for the same capacity disk array than RAID-5 at the expense of a larger number of disks and therefore also a higher number of faulty disks.

8.5 Mission success probabilities

In this final reliability study, mission success probabilities of RAID-1 and RAID-5 arrays are compared. Table 14 illustrates reliability of disk arrays with 1, 5, and 50 data disks. With one and five data disks, all three mission success probabilities are very close to 100% as reliability of the disk arrays is very high as shown by MTTDL. There is no significant difference between RAID-1 and RAID-5 arrays. On the contrary, there is a significant difference in the mission success probabilities when there are 50 data disks in the disk array. The RAID-1 configuration can still provide good reliability while RAID-5 has a significantly degraded reliability.

If the faults are related, reliability will drop dramatically as it was shown earlier in this chapter in Scenario 5. Then, the reliability would be only a fraction of the values listed in Table 14. However, RAID-1 would still provide a significantly better reliability for three main reasons. First, the mirroring approach is more suitable for high reliability. Second, the repair process is much faster as only two disks are involved and the repair process is simpler (as described in the next chapter). Finally, the RAID-1 architecture is less sensitive to related faults because it is constructed of pairs of disks and not like in the RAID-5 architecture where a disk group has several disks. As a conclusion, the RAID-1 architecture can repair disk faults much faster and therefore it tolerates better related faults.

Table 14. Reliability of RAID-1 and RAID-5 arrays with 1, 5, and 50 disks

Disks	Array configuration	MTTDL [hours]	M1 [%]	M3 [%]	M10 [%]
1	EMM1: RAID-1	3,34E+08	99,997 %	99,992 %	99,974 %
	EMM1: RAID-5	3,34E+08	99,997 %	99,992 %	99,974 %
5	EMM1: RAID-1	6,67E+07	99,987 %	99,961 %	99,869 %
	EMM1: RAID-5	2,23E+07	99,961 %	99,882 %	99,607 %
50	EMM1: RAID-1	6,67E+06	99,869 %	99,607 %	98,695 %
	EMM1: RAID-5	2,66E+05	96,772 %	90,610 %	71,973 %

9. PERFORMABILITY

In this chapter, performability of a disk array subsystem is studied. The focus is concentrated on the performability as the definition of "cost" in the cost-performability is somewhat ambiguous. A simple performance model of the disk array is used in this chapter, because more accurate models are considered to be out of the scope of this thesis.

9.1 Performability models

Performability (i.e., combined performance and reliability) of a system can be expressed using Markov reward models [Trivedi 1994, Catania 1993, Pattipati 1993, Smith 1988, Furchgott 1984, Meyer 1980, Beaudry 1978]. Figure 37 illustrates a performability model for TMM presented in Chapter 6. This is a typical RAID-5 array with a *D+1* redundancy scheme. For each state *i* (i=0,1, and 2), two parameters are defined: probability and reward.

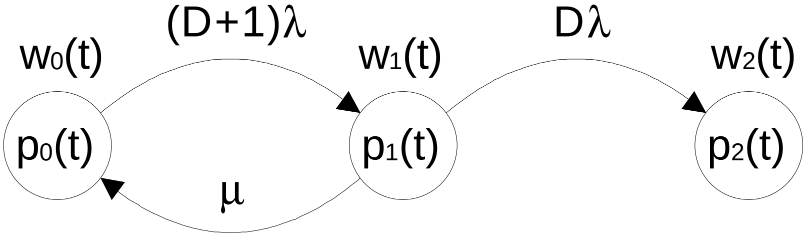

Figure 37. Simple Markov model for performability of TMM (D is the number of disks, λ is the disk failure rate, μ is the repair rate, $p_i(t)$ defines the probability of the system being at state i at time t where i defines the number of faulty disks in the disk array, and $w_i(t)$ defines the reward function at state i at time t)

The first parameter, probability ($p_i(t)$), defines the probability of the system being in state *i* at a given time *t*. This is the same probability as used for the reliability analysis in Chapters 6, 7, and 8.

The second parameter, reward ($w_i(t)$), defines the reward what system gets while being in state *i* at a given time *t*. The reward function can be, for example, the performance of the system. In a disk array, the performance can be expressed using the number of I/O operations per second. For example, in state *0* of Figure 37, the reward function specifies the number of user I/O operations that the disk array can perform in the fault-free state. In state *1*, the reward function specifies the number of user I/O operations that the crippled disk array can perform while it is either waiting for the repair process to start or while the repair process in ongoing. In state *2*, the reward function is zero because the data is lost and the disk array has failed.

For simplicity, it is assumed in this thesis that the reward function is constant (i.e., $w_i(t) = W_i, \forall t$) and only depends on state *i* but not the time.

The performability (or computational availability) in state *i* at a given time *t* can be then expressed as a product of the two above mentioned parameters (i.e., reward and probability of state *i*) as follows

$$CA_i(t) = p_i(t)w_i(t) = p_i(t)W_i \, . \tag{106}$$

And, the total performability at a given time *t* can be expressed as the sum of the performabilities of all states *i* as follows

$$CA(t) = \sum_i CA_i(t) = \sum_i p_i(t)w_i(t) = \sum_i p_i(t)W_i \, . \tag{107}$$

Steady state performability

If the Markov model describes a steady state system, the performability can be expressed as follows

$$CA = \sum_i CA_i = \sum_i p_i W_i \, . \tag{108}$$

Non-steady state performability

In a non-steady state system, the probability of state *i* is changing. Eventually, the system will fail (in Figure 37, $\lim_{t\to\infty} p_2(t) = 1$). The cumulative performability of a system with non-repairable faults can be expressed as

$$CCA = \int_0^\infty CA(t)dt = \sum_i W_i CP_i \tag{109}$$

where CP_i is the cumulative reliability of state *i*.

9.1.1 Performability of TMM

The performability of a RAID-5 disk array that is modeled with TMM can be expressed using the above equation (109) and the probabilities of states *0*, *1*, and *2* as expressed in Chapter 6 in equations (12) - (14). The cumulative reliabilities of TMM are:

$$\begin{aligned} CP_0 &= \int_0^\infty p_0(t)dt \\ &= \int_0^\infty \frac{(D\lambda+\mu+\xi)e^{\xi t}-(D\lambda+\mu+\zeta)e^{\zeta t}}{\xi-\zeta}dt \\ &= -\frac{\zeta(D\lambda+\mu+\xi)-\xi(D\lambda+\mu+\zeta)}{\xi\zeta(\xi-\zeta)} \\ &= \frac{D\lambda+\mu}{D(D+1)\lambda^2} \end{aligned}, \tag{110}$$

$$\begin{aligned} CP_1 &= \int_0^\infty p_1(t)dt \\ &= \int_0^\infty \frac{(D+1)\lambda}{\xi-\zeta}(e^{\xi t}-e^{\zeta t})dt \\ &= -\frac{(D+1)\lambda}{(\xi-\zeta)}(\frac{1}{\xi}-\frac{1}{\zeta}) \\ &= \frac{(D+1)\lambda}{D(D+1)\lambda^2} \end{aligned}, \tag{111}$$

and the value of CP_2 has no effect since $W_2 = 0$. Then, the performability of TMM is

$$CCA_{TMM} = \sum_i W_i CP_i = W_0 CP_0 + W_1 CP_1 \tag{112}$$

where W_0 and W_1 are the reward functions of state *0* and *1*, respectively. The reward functions depend on the type of the operation (read or write).

It should be noticed that CCA_{TMM} equals MTTDL of TMM if W_0 and W_1 equal one. As W_0 is typically greater than or equal to W_1, it is possible to obtain an upper limit estimation for performability by multiplying MTTDL of the array with the reward function of the fault-free state. Hence, the approximation of the performability can be expressed as

$$CCA_{TMM,A} = W_0 MTTDL_{TMM} \tag{113}$$

where $MTTDL_{TMM}$ is MTTDL of the array.

9.1.2 Performability of EMM1

The performability of a RAID-5 disk array that is modeled with EMM1 can be expressed using the above equation (109) and the probabilities of states *00*, *01*, *10*, and *f* as expressed in Chapter 7 in equations (61) - (64). The cumulative reliabilities of EMM1 are:

$$\begin{aligned}
CP_{00} &= \int_0^{\infty} p_{00}(t)dt \\
&= \int_0^{\infty} \sum_{i=0}^{2} \frac{\left[r_i + \mu_s + \lambda_d + D(\lambda_s + \lambda_d)\right]\left[r_i + \mu_d + D(\lambda_{df} + S\lambda_s)\right]e^{r_i t}}{Q_I} dt\,, \\
&= \sum_{i=0}^{2} \frac{-\left[r_i + \mu_s + \lambda_d + D(\lambda_s + \lambda_d)\right]\left[r_i + \mu_d + D(\lambda_{df} + S\lambda_s)\right]}{Q_I r_i}
\end{aligned} \tag{114}$$

$$\begin{aligned}
CP_{01} &= \int_0^{\infty} p_{01}(t)dt \\
&= \int_0^{\infty} \sum_{i=0}^{2} \frac{(D+1)S\lambda_s[r_i + \mu_d + D(\lambda_{df} + S\lambda_s)]e^{r_i t}}{Q_I} dt\,, \\
&= \sum_{i=0}^{2} \frac{-(D+1)S\lambda_s[r_i + \mu_d + D(\lambda_{df} + S\lambda_s)]}{Q_I r_i}
\end{aligned} \tag{115}$$

$$\begin{aligned}
CP_{10} &= \int_0^{\infty} p_{10}(t)dt \\
&= \int_0^{\infty} \sum_{i=0}^{2} \frac{\begin{bmatrix} (D+1)\lambda_d r_i + (D+1)\lambda_d \mu_s + (D+1)S\lambda_d \lambda_s + \\ (D+1)\lambda_d{}^2 + D(D+1)\lambda_d(\lambda_s + \lambda_d) \end{bmatrix} e^{r_i t}}{Q_I} dt\,, \\
&= \sum_{i=0}^{2} \frac{-\begin{bmatrix} (D+1)\lambda_d r_i + (D+1)\lambda_d \mu_s + (D+1)S\lambda_d \lambda_s + \\ (D+1)\lambda_d{}^2 + D(D+1)\lambda_d(\lambda_s + \lambda_d) \end{bmatrix}}{Q_I r_i}
\end{aligned} \tag{116}$$

and the value of CP_f has no effect since $W_f = 0$. Then, the performability of EMM1 is

$$CCA_{EMM1} = \sum_i W_i CP_i = W_{00}CP_{00} + W_{01}CP_{01} + W_{10}CP_{10} \tag{117}$$

where W_{00}, W_{01}, and W_{10} are the reward functions of state *00, 01,* and *10,* respectively. The reward functions depend on the type of the operation (read or write).

9.1.3 Reward functions of disk array subsystems

The performance of disk arrays can be modeled using a simple performance model for the arrays like in [Hillo 1993, Gibson 1991, Kemppainen 1991]. Here, the reward functions are modeled using the performance of the disk array that is estimated for either read or write operations but not for mixed read and write operations. More accurate performance model of the disk arrays is considered to be out of the scope of this thesis.

RAID-5 performance

In a RAID-5 array, a total of $D+1$ disks is used for building an array of D data disks. There are D disks in the crippled array. If I/O requests are not assumed to span over several disks, each request would require the following number of disk accesses:

- 1 disk operation to read from a fault-free disk array;
- D disk operations to read from a crippled disk array (in the worst case)[*];
- 4 disk operations to write to a fault-free disk array;
- $D+1$ disk operations to write to a crippled disk array (in the worst case)[†]; and
- $D+1$ disk operations to reconstruct a faulty disk block.

RAID-1 performance

The above equations (112) and (117) for performability are dedicated to RAID-5 arrays analysis. However, later in this chapter, it is shown that good estimation for the performability can be made using MTTDL of the array and the reward function of the fault-free state. Hence, reward functions for the RAID-1 array are also included here.

In a RAID-1 array, a total of $2D$ disks is used for building an array of D data disks. There are $2D-1$ disks in the crippled array. If I/O requests are not assumed to span over several disks, each request would require the following number of disk accesses:

- 1 disk operation to read from a fault-free disk array;
- 1 disk operation to read from a crippled disk array;
- 2 disk operations to write to a fault-free disk array;
- 2 disk operations to write to a crippled disk array (in the worst case)[‡]; and

[*] To read from a crippled RAID-5 array requires either 1 or *D* disk operations depending on the completion of the repair process and whether the request refers to the failed disk or other disks. For simplicity, it is assumed that performance of the crippled array is the worst possible.

[†] To write to a crippled RAID-5 array requires either 4 or *D+1* disk operations depending on the completion of the repair process and whether the request refers to the failed disk or other disks. For simplicity, it is assumed that performance of the crippled array is the worst possible.

[‡] To write to a crippled RAID-1 array requires either one or two disk operations depending on the completion of the repair process and whether the request refers to the failed disk or other disks. For simplicity, it is assumed that performance of the crippled array is

- 2 disk operations to reconstruct a faulty disk block.

Relative performance

In a disk array, the maximum number of I/O operations depends on the array configuration, the type of the operation and the properties of the disks. The array configuration specifies how many parallel read and write operations can be performed as illustrated in the introduction in Chapter 1. In this thesis, the performance is expressed as relative comparison with a single disk. Relative performance value one corresponds to one fully working disk serving user requests. For example, a fault-free RAID-1 with two disks has relative performance two for read operations and one for write operations.

Effect of the scanning algorithm

The effect of the scanning algorithm is studied by reserving a certain capacity for the scanning algorithm. For every disk, a certain capacity (as expressed with a_{scan}) is reserved for scanning and remaining capacity ($1-a_{scan}$) is available for other requests (user requests or repair).

Effect of the repair process

The repair process decreases the maximum number of user operations in the crippled array. The degree of degradation depends on the activity of the repair process. When, for example, a disk array of a total of ten disks is being repaired using 20% of the capacity for repair (as expressed with the repair activity, a_{repair}), the theoretical remaining capacity is 8 units. This is further reduced if the read or write request needs to have several disk operations. For example, to write to a crippled RAID-5 array needs 10 disk operations. Hence, the relative performance is only $8/10 = 0.8$. As for comparison, the relative write performance in the same size fault-free array would be 2.5.

Reward functions of RAID-5 and RAID-1

The relative reward functions of RAID-5 and RAID-1 arrays are illustrated in Table 15. It is assumed that three different states from the point of view of performance are:

- all disks working (state *0* in TMM and states *00* and *01* in EMM1);
- one disk unit failed (state *1* in TMM and state *10* in EMM1); and
- data lost (state *2* in TMM and state *f* in EMM1).

Sector faults are considered not to degrade the performance.

the worst possible.

Table 15. Relative reward functions of RAID-5 and RAID-1

RAID-5	Reward function for a read operation	Reward function for a write operation
W_0, W_{00}, W_{01}	$(D+1)(1-a_{scan})$	$\frac{(D+1)(1-a_{scan})}{4}$
W_1, W_{10}	$(1-a_{repair})(1-a_{scan})$	$\frac{D(1-a_{repair})(1-a_{scan})}{(D+1)}$
W_2, W_f	0	0
RAID-1	Reward function for a read operation	Reward function for a write operation
W_0, W_{00}, W_{01}	$2D(1-a_{scan})$	$D(1-a_{scan})$
W_1, W_{10}	$(2D-1-a_{repair})(1-a_{scan})$	$(D-a_{repair})(1-a_{scan})$
W_2, W_f	0	0

In a RAID-5 array, all disks are involved with the repair process. As the worst case scenario is used here, the read operation to a crippled array would require to access all remaining disks. Hence, the relative performance is one from which the repair activity is deducted. Similarly in the worst case, the write operation requires to read all remaining disks once and to write to one disk. From this relative performance, the repair activity is deducted.

In a RAID-1 array, only two disks are involved with the repair process. When the array has D data disks ($2D$ disks totally), $(D-1)$ data disks are not effected by a disk unit fault. For a read operation, there are $(2D-1)$ disks available and the performance is further reduced by the repair process in one disk. For a write operation, there are $(D-1)$ data disks that are not effected by the disk unit fault and one data disk that is effected by the repair process.

9.1.4 Performability comparisons

Performability of a RAID-5 array modeled using TMM and EMM1 models is illustrated in Figure 38. Here, the same default parameters are used as in Chapter 8. This figure shows that the approximation (performability equals MTTDL multiplied with the reward function of the fault-free state) provides accurate results. Hence, the same approximation principle is used with the RAID-1 array. It is also concluded that both performability models provide similar results that correspond to the reliability results.

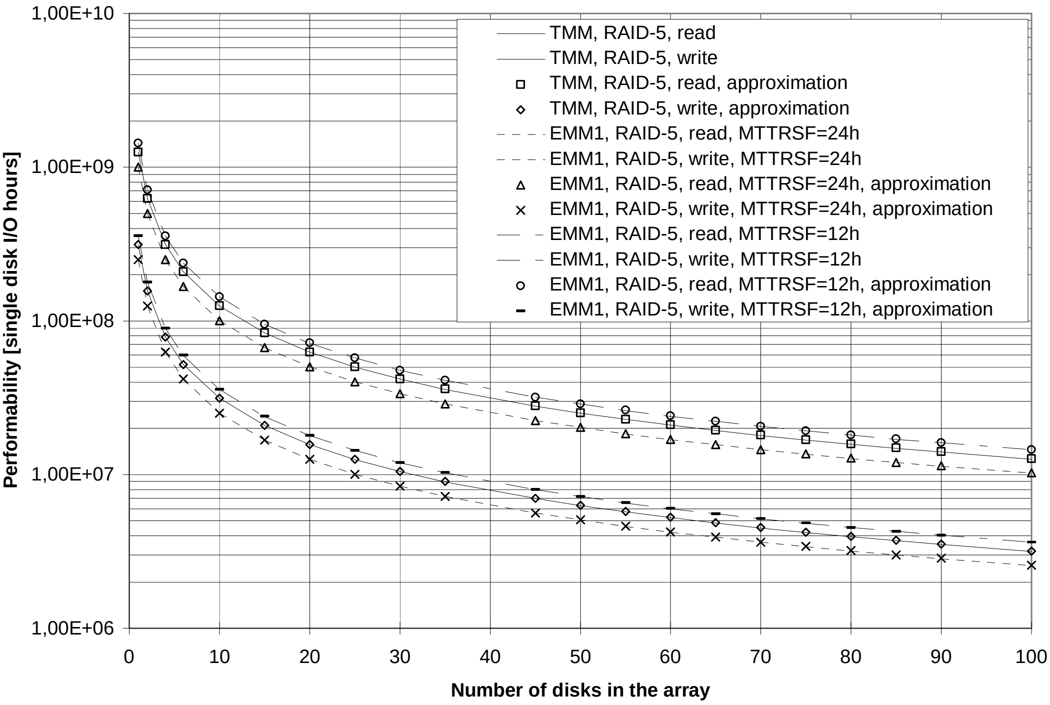

Figure 38. Performability of RAID-5 array as a function of the number of disks in the array

Effect of the repair activity

The effect of the repair activity is studied in a configuration where the repair time depends on the number of disks and the repair activity. In Figure 39, performability of a RAID-5 array is illustrated as a function of the repair activity. Here, a RAID-5 array of 50 data disk is studied in two configurations: hot swap (repair starts 8 hours after the disk failure) and hot spare (repair starts immediately after the disk failure). The read operation provides four times better performability than the write operation as its reward function in state *00* is four times better. The repair time is assumed to be two hours with 100% repair activity and relatively longer, if the repair activity is less than 100%. The hot spare configuration provides significantly better performability than the hot swap configuration as the repair time in the latter case is shorter. The performability of state *10* of EMM1 has only a marginal effect (less than 1%) on the total performability of the RAID-5 array. This is because the failure rates in EMM1 are much smaller than the repair rates and therefore the system is mainly in state *00*.

The only factor that may limit the disk repair activity is the performance requirement during the repair time. If no minimum requirement for performance during the repair time is set, then the repair can and should be done at full speed, otherwise the repair activity should be obstructed to guarantee the minimum performance. The reliability increase due to faster repair is much more significant

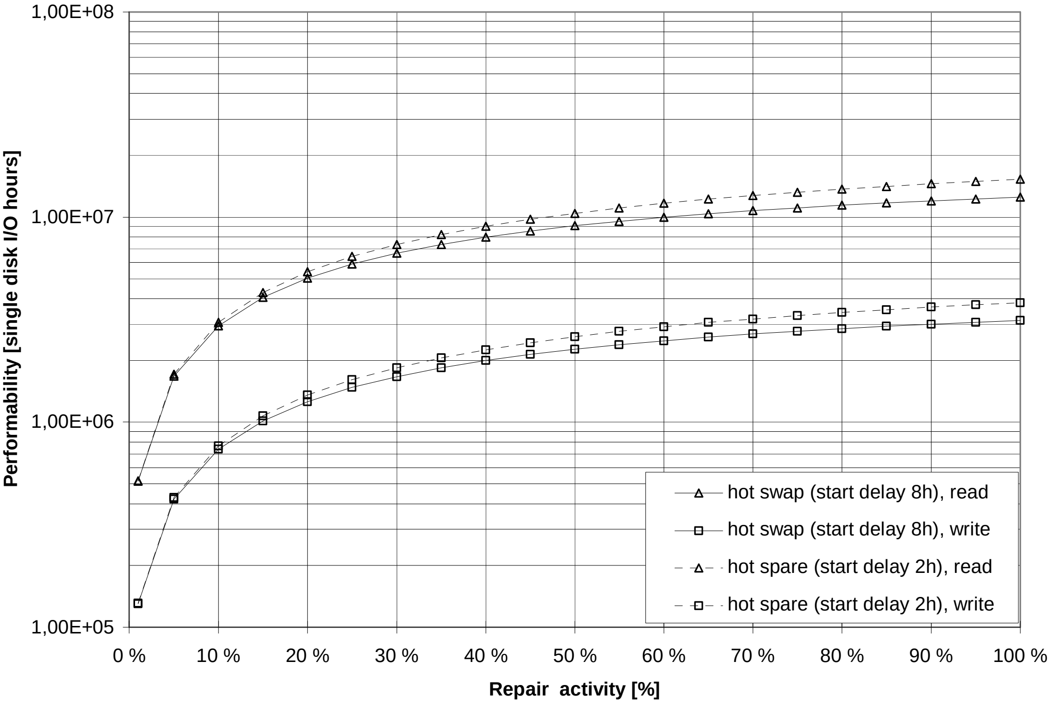

Figure 39. Performability of RAID5 array modeled with EMM1 as a function of repair activity

than the minor performance degradation during the repair time when the total performability is considered.

Effect of the scanning algorithm

The effect of the scanning algorithm on the performability is studied by varying the scanning activity. It is assumed that it takes 24 hours to scan all disks in the array with 5% scanning activity. The performability of a RAID-5 array is presented in Figure 40 as a function of the scanning activity. When the hot swap (hot spare) configuration is used, the optimum performability is achieved in this sample configuration when the scanning algorithm uses 20% (30%) of the capacity for scanning the disk surface. The hot swap configuration reaches its peak performability earlier than the hot spare configuration as its reliability is dominated more by the longer disk repair time than the hot spare where the reliability can be increased longer with the increased scanning activity and its sector faults detection. Eventually in both cases, the performability starts decreasing when the scanning activity approaches 100%. This is obvious since less and less capacity of the array remains for user disk requests and the reliability does not increase because it is limited by the repair time of the disk unit failure.

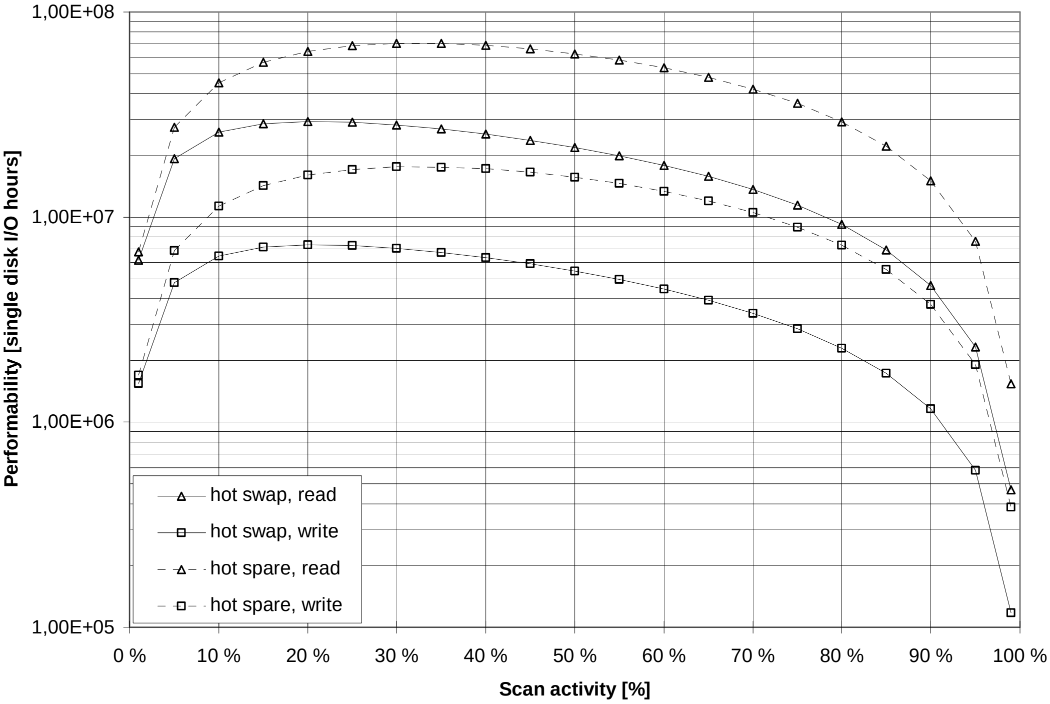

Figure 40. Performability of RAID5 array modeled with EMM1 as a function of scanning activity

<u>RAID-5 vs. RAID-1</u>

The performability of RAID-1 and RAID-5 arrays is compared in Figure 41. The performability of the RAID-5 array is achieved using the above equations (112) and (117) while the performability of the RAID-1 array is approximated using MTTDL of RAID-1 multiplied with the appropriate reward function. As MTTDL of a RAID-1 array is approximated by dividing MTTDL of RAID-1 with two disks with the number of data disks in the array while the reward function is relative to the number of disks in the array, the performability of the RAID-1 array is constant (i.e., while the performance of the disk array increases with the number of disks in the array, at the same time the reliability decreases thus keeping the performability constant). Actually, the same effect can be found also with RAID-0 arrays where the performability remains constant but at a much lower level because the RAID-0 array has no redundancy. On the other hand, the performance of the RAID-5 array increases almost linearly with the number of disks in the array, but the reliability decreases more rapidly as more and more disks are protected just with a single disk.

9.1.5 Conclusions of performability analysis

The conclusions of the performability analysis are gathered in the following list:

- Performability of a disk array can be well approximated by multiplying MTTDL of the

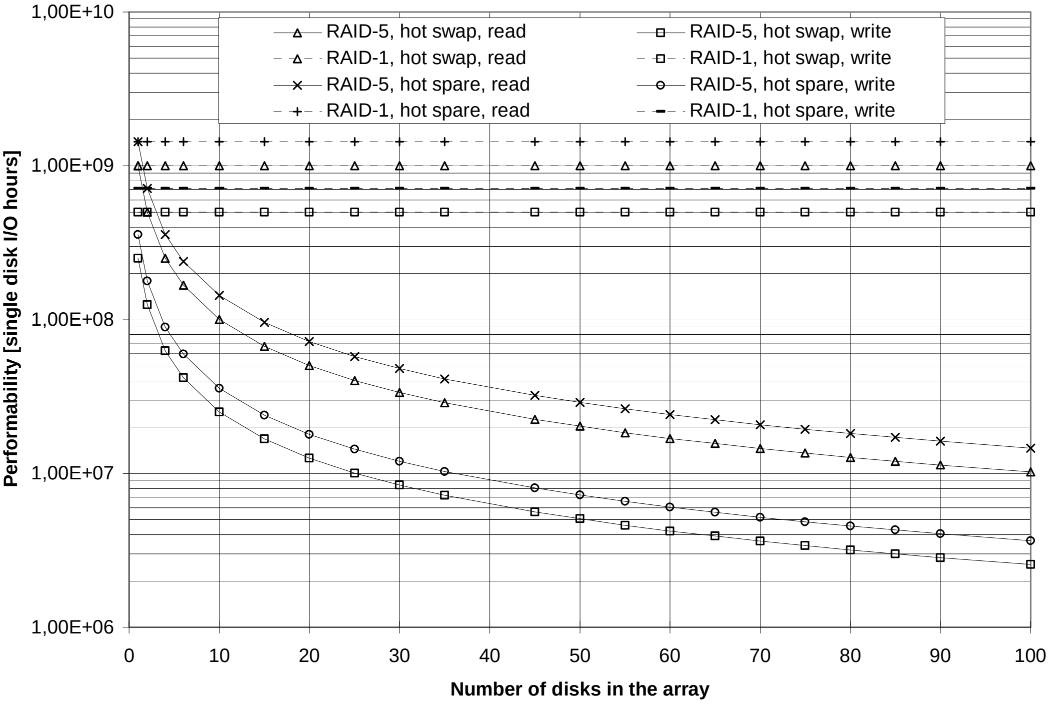

Figure 41. Performability of RAID-1 and RAID-5 arrays

array with the reward function of the fault-free state when the repair rates are much higher than the failure rates.

- Performability of RAID-0 and RAID-1 arrays is constant regardless of the number of disks in the array. Higher performance is achieved with larger number of disks but at the expense of reduced reliability.
- Performability of a RAID-5 array decreases as the number of disks increases. This is because reliability drops more than what performance increases.
- A RAID-1 array provides better performability than a RAID-5 array with the same number of data disks. The penalty for higher performability of the RAID-1 array is the larger number of disks in the array and higher number of failed disks.
- A scanning algorithm can improve performability. The scanning algorithm increases first the performability as the disk array reliability increases while the performance degradation remains still moderate. When the scanning activity increases further, the reliability no longer increases because the reliability bottleneck will be the disk unit faults, but at the same time the performance of the array drops. Thus, the performability also sinks.
- The increased speed of the repair process effects the performability by improving the reliability while the effect on the average performance is marginal. The only reason to limit

the speed of the repair process is to guarantee a certain performance even with a crippled array.

10. DISK ARRAY AS PART OF A COMPUTER SYSTEM

When a highly reliable disk array system is designed, it should be remembered that the disk array is just a part of a larger system and the reliability of the system is dominated by its weakest link. The average reliability of various components of the computer system is far less than the reliability of the disk arrays discussed in this thesis [PCMagazine 1996, Hillo 1993, Gibson 1991].

Disk subsystem

Beside hard disks, a disk subsystem has components such as fans, power supplies, power cables, data cables, and a disk controller [Hillo 1993, Gibson 1991]. The importance of the fans, for example, was stated already earlier as the temperature of disks rises rapidly if the fans do not operate or the ventilation is inadequate. Similarly, the importance of a reliable power supply is obvious. Beside the normal reliability requirements, the power supply should provide stable voltage for disks despite their activity as a disk can shut itself down if the voltage is not stable enough [Räsänen 1994, Seagate 1992]. The power and data cables are typically very reliable (at least when compared with other components) [Gibson 1991]. As a fault in cabling can disable several disks at the same time, a special care must be taken to arrange the disk array with minimized risk of related faults.

One of the most unreliable parts of the disk subsystem is the disk controller [Hillo 1993, Gibson 1991]. Especially, the large amount of RAM (e.g., used for cache buffers) reduces significantly the reliability of the controller unless non-volatile ECC based memory is used [Hillo 1993].

The major difference of the faults in the surrounding components of a disk subsystem compared with the faults in the disk units themselves is data unavailability instead of permanent data loss. The surrounding components can fail causing temporary data unavailability while the data is not actually lost (i.e., data can be made available again by repairing the faulty unit). However, some of the faults in the surrounding components may also cause data loss. For example, data stored temporarily in a disk controller (but not yet written into a disk) is lost during a power failure if the memory has no battery backup.

Computer system

The other parts of the computer system (such as host CPU, main memory, network interface, other I/O devices, and operating system) have also a significant impact on the total reliability. Typically, the reliability of the system is reduced further by these components. Only in highly reliable/available computer systems, the reliability of these other parts of the computer system is high enough (e.g., due to redundant components) that the impact of the disk subsystem reliability becomes significant.

Here, only hardware related components have been discussed, but, in practical systems, significant portion of faults is caused by software errors for example in the operating system, the device drivers, or the disk array firmware.

Human errors

One of the main causes for data loss in a modern computer system is neither the physical failures of the equipment nor the software errors but human errors. A disk array or any other reliable hardware configuration does not prevent a user from deleting accidentally the wrong files from the system.

Some of the human errors can be prevented by advanced hardware design. For example, if the disk array supports the hot swap concept, those disks that are currently in use should be protected against accidental pull out. A typical example that can cause data loss in such a system is when a serviceman pulls accidentally a wrong disk out of a crippled array. By pulling out the wrong disk, the consistency of the array is lost since no redundancy was left after the disk failure. This can be prevented by software controlled physical locks that allow the serviceman to pull out only the failed disk.

Importance of backups

Reliability improvement of a computer system does not make the backups obsolete. On the contrary, the backups are still needed and they are a way to protect against human errors and major accidents that could destroy an entire computer system. A good example of such an approach is a distributed computing and backup system where distant computers are mirrored to ensure a survival even after a major catastrophe [Varhol 1991].

11. CONCLUSIONS

In this thesis, performance and reliability effects of disk array subsystems have been studied. The main objective of this thesis has been to emphasize the importance of latent fault detection and its effect on the reliability and data availability of disk arrays. Significant improvements in both reliability and data availability can be achieved when latent faults are detected using the algorithms proposed in this thesis in comparison to normal disk arrays where latent faults are discovered only when a user request happens to access the faulty areas.

This thesis categorizes faults in a disk with two properties based on the fault severity (a sector fault or an entire disk unit fault) and its detection time (immediately detected or latent). A sector fault effects on a limited area of the disk causing one or a few sectors to have problems in maintaining data. On the contrary, a disk unit fault causes a significant part or an entire disk to be inaccessible. Detection of a disk unit fault is by its nature fast while sector faults can be either detected immediately or they may remain latent. In a disk array, disks are polled typically in a few seconds interval and a disk unit fault can be detected at the latest by the polling process what means in a matter of seconds. Hence, a disk unit fault seldom remains undetected for a longer time. Similarly, a sector fault is detected only when that area is accessed. Unfortunately, this can mean several weeks if the disk access pattern is unevenly distributed and the fault occurs in a rarely accessed area.

Modern disk arrays are designed to handle and recover disk unit and sector faults on the fly. While the array is serving normal user disk requests, information of the faulty disk can be reconstructed using the redundant information on the other disks and stored into a spare disk. The spare disk can be either a hot spare or the faulty disk can be hot swapped. Similarly, a sector fault can be recovered using appropriate recovery methods within a disk. Current commercially available disk arrays are not yet, however, equipped with a mechanism that would actively detect latent faults.

Typically, sector faults have been ignored in the technical literature. They are considered to be of lesser importance than disk unit faults as only one sector out of millions loses its data. However, the importance of even a single sector can be seen, for example, in a large database system where every sector counts. In such a database, even one lost sector may imply that the entire data must be considered to be inconsistent.

Modern disks, especially those that comply with the SCSI-2 standard, are capable of handling sector repairs when a sector fault is detected. A disk typically has a logical representation of the disk space (represented as a sequential list of logical sectors) that is separated from its physical structure (heads, tracks and physical sectors). In the case of a sector fault, a faulty physical sector can be replaced with a spare sector without changing the logical representation of the disk. If the disk detects a sector fault during a write operation, the sector remapping can be done automatically. However, the disk is unable to do the data recovery by itself with a read operation. In that case, the array configuration and its redundant information are needed as the missing data is recovered using data on the other disks of the disk array.

Latent faults in a disk array can be detected using scanning algorithms like those proposed in this thesis. The basic scanning algorithm is an adaptation of the memory scrubbing algorithm that is commonly used for detecting faults in primary memory. However, the scanning algorithms for latent fault detection in secondary memory are for the first time presented and analyzed in this thesis and in publications by the author.

The proposed disk scanning algorithms utilize the idle time of the system to scan the disk surface in order to detect latent faults. A scanning read request to the disk is issued only when the disk is detected to be idle. Hence, the additional delay that is experienced by a normal user disk request will not be significant even when the disk is heavily loaded. Any user disk request may need to wait additionally at most one scanning disk request to complete. As the size of a scanning disk request is typically approximately the same as that of normal user requests, the additional delay is nominal. However, the scanning algorithm may increase seek delays. If longer scanning requests are used, the request can be aborted in the case a user disk request is received.

The two benefits of using the disk scanning algorithm are: faster detection of latent faults and improved data availability. As user requests to a disk subsystem are typically accessing the disk space unevenly, the disk requests caused by normal user activity leave a significant part of the disk subsystem unaccessed for a long time. A problem arises due to the fundamental error recovery principle of the disk array. A typical disk array is capable of recovering only one fault in a group of disks. In the case of a latent fault (just a faulty sector) and a disk unit fault at the same time, the disk array loses its consistency as there are two simultaneous faults and the repair mechanism is unable to restore all data.

The main assumption of the proposed scanning algorithms is that the extra disk accesses cause no additional wear on the disk. This is generally true when the disk is spinning continuously without spindowns due to inactivity. Typically, the scanning requests represent only a minor portion of the disk load. Hence, the additional activity will not cause extensive wear in the form of seeks around the disk. As the scanning process is only reading the disk (not writing) there is no danger of losing data due to a power failure.

11.1 Results of this thesis

This thesis increases understanding of the reliability of a disk array. Especially, the importance of the latent fault detection is shown in the analysis and the proposed scanning algorithms indicate significant improvement on reliability and data availability. The impact on performance due to the scanning algorithms is shown to be usually marginal since scanning is typically done while the system is otherwise idle.

The analysis of disk array reliability with dual fault types is also new in this thesis. With this analysis, an analytical representation of a disk array reliability and data availability have been presented. Simple formulae have been derived for array reliability (mean time to data loss, MTTDL) and data availability (mission success probability).

The analysis is done for a generic array configuration. Hence, the produced formulae are in a general format and they can be used with arbitrary number of disks in the array. Also, the equations are independent of the disk array architecture and repair methods (except different repair time and the number of disks involved in the repair).

The analysis is divided into two categories based on the repair processes: hot swap or hot spare. The RAID-1 and RAID-5 arrays have been used as examples due to their popularity among disk arrays. Hot swap and hot spare methods are analyzed separately as the former assumes that the spare units are fault-free, but the repair process needs human intervention (the repair process may start a long time after a fault is detected) while the latter can start the repair process immediately after the fault detection, but it has a risk of having faulty spare unit. Due to complexity of the equations, the hot spare method is analyzed only using an approximation while the hot swap method is also analyzed analytically.

In the reliability analysis of the hot spare system, it has been noticed that the spare disk fault possibility does not have a significant effect on the reliability (neither decrease nor increase) when compared with the hot swap system if the active disk unit repair time is the same. This is in line with the results in the technical literature. The hot spare provides better reliability just because the repair process can be started immediately after the fault detection, and unlike in the hot swap case where user intervention is needed.

The results also have pointed out that it is possible to use the first analytical model (EMM1) in analyzing the hot spare disk arrays instead of the more complex model (EMM2) as both provide very similar results when the same repair times and failure rates are used. This is due to the fact that the spare disk reliability has no significant effect on the disk array reliability.

Interrelated faults

Interesting results were found when the interrelated faults were analyzed. When the second fault is assumed to occur with higher probability than the first fault (e.g., if the disks are from the same manufacturing batch or they are located in the same cabinet where temperature is increased due to a faulty fan), the reliability of the disk array drops dramatically. Eventually, a disk array system that was originally built as *D+1* redundancy is acting like a system with *D+1* parallel units with no redundancy (i.e., a RAID-5 array would actually be as reliable as a RAID-0 array). In practice, the situation may be even worse because the probability of having the first fault is even higher if the disks are coming from the same (inferior) manufacturing batch or the disks are otherwise prone to faults.

RAID-1 or RAID-5?

When the RAID-1 and RAID-5 disk arrays are compared, it has been noticed that RAID-1 provides better reliability and better performability than RAID-5 in all cases where the number of data disks is the same. An additional benefit of the RAID-1 array compared with the RAID-5 array is the speed of the repair process. In the RAID-1 array, only two disks are involved with the repair process while, in the RAID-5 array, all disks are involved. This means that, in large disk arrays, the RAID-1 architecture can repair a disk fault significantly faster than the RAID-5 architecture. The main disadvantages of the RAID-1 architecture are the high number of disks, larger number of faulty disks, and higher initial cost. As RAID-1 uses *D+D* redundancy instead of *D+1* redundancy like in RAID-5, the number of disks is almost doubled. This causes also almost double the number of faulty disks in the RAID-1 array, but still the reliability is higher. As the prices of hard disks are falling, the initial cost of the RAID-1 array should not be a significant problem for those who want to have a disk array that has both good performance and reliability.

Limitations

The main limitations of this analysis are that the array is assumed to tolerate only one fault in the disk group at any time and that only one array group is studied. The former limitation is a typical restriction of a conventional disk array as systems that tolerate multiple faults in the same disk group are generally considered to be too expensive with respect to money and performance. The latter limitation restricts the usage of these results in arrays with a single group of disks and in arrays where the number of spare disks is sufficient to allow multiple repair processes to be started simultaneously.

11.2 Usage of the results of this thesis

The results of this thesis can be used for obtaining more reliable disk array systems that will fulfill given performability requirements even during the recovery phase. This can be done by minimizing the reliability bottlenecks caused by latent faults in the disk arrays and by implementing a delayed repair method that reduces the performance degradation during the repair phase. This thesis will also increase the awareness of the effect of latent faults to the reliability of disk arrays and hopefully lead into better and more reliable disk arrays in the future.

With the new equations, it is possible to optimize disk arrays with respect to cost, performance, and reliability. Especially, it is possible to analyze the worst case scenarios when disk faults are related or disks are from the same manufacturing batch. In this case, it is very likely that second disk unit fault occurs soon after the first one.

This thesis has also a significant impact on the disk array development. The proposed scanning algorithms can be implemented already today. Actually, some of the basic scanning ideas are already in use [Scritsmier 1996]. Also, the ideas and the results of the reliability analysis of this thesis can be utilized when developing and optimizing new disk arrays.

The proposed scanning algorithms can also be used with non-redundant arrays and single disks. The scanning algorithms can detect early signs of media deterioration that are indicated as increased number of retries. This provides a mechanism to replace deteriorated sectors before the data is lost. Quite similar implementation is already in use in Microsoft’s Windows 95. Hence, the reliability can be improved also in a non-redundant disk subsystem.

11.3 Further studies in this area

The analysis of this thesis can be expanded in various areas. For example, hard disk diagnostics, next generation disk arrays, more sophisticated repair methods, and higher level fault resilient disk arrays can benefit from the ideas introduced here. Also, cost-performability of disk arrays should be studied.

One especially interesting area, where it is possible to utilize the scanning algorithms proposed in this thesis, is in the hard disks and their internal diagnostics. As a disk itself knows the best its own activity, it is obvious that the scanning process should be performed entirely inside the disk. There would be several benefits of doing this. First, the array controller would be released to do other duties. Also, the disk itself has better indication of the media deterioration as even the smallest problems are recognized. The main impact on the disk design would be in the standardization of the disk interfaces. The disks would then be able to predict data deterioration early enough that data loss could be prevented even with a single disk.

New generations of disk arrays have been introduced to improve array performance and reliability. Their performance effects and repair processes need more investigation. The analysis that is done in this thesis should be expanded into these new array architectures as well as systems with multiple arrays.

The computer systems are more and more used in continuously operating environments where no interrupts or down-times are tolerated and therefore faulty disk units should be repaired online. At the same time, the response time requirements tolerate no performance degradation even during the recovery or the degraded states. Hence, it should be possible to adjust the recovery process according to performance (and reliability) requirements. For example, the recovery process could adapt its activity based on the user activity or the degree of the completeness of the disk recovery. For example, the repair process of a disk unit fault in a RAID-5 array may delay its operation at the beginning as the user requests are already suffering from access to a crippled array. When the repair process is getting more complete, it can increase its activity as more and more user requests fall already at the repaired area where the performance is the same as in a fault-free array.

Some of the disk arrays can tolerate more than one fault at the same time in the same disk group. In such arrays, latent sector faults are not as catastrophic as in arrays that tolerate only one fault per time. However, latent faults will also decrease dramatically the reliability of those arrays. Hence, the scanning algorithm is vital even in those arrays as they typically have extremely high expectations on reliability. Thus, the effect of the proposed scanning algorithms in such environments should be analyzed.

In the future, the importance of the high performance data storage subsystem will increase with the new applications when large amounts of data are processed. As it has been shown, the performance gap between the secondary memory and the processing capacity is ever growing and therefore the bottleneck in the system lies in the I/O subsystem. Hence, the development efforts should be concentrated more on the data storage side to balance the performance of all components.

At the same time, reliability and cost of the system should not be forgotten. The total reliability should be at least as good as with the earlier systems (despite the larger number of components) but preferably even much higher. Total cost of the system can also be taken into account if cost-performability is used instead of performability in the disk arrays analysis. In principle, all costs should be minimized and all profits should be maximized. However, this is not so simple when also performance and reliability must be considered. Thus, a special interest should be focused on the definition of cost-performability equations to get similar generic metrics as with performability.

One of the main factors in cost-performability is the cost of lost data. Thus the reliability of a disk array should be very high. This can be achieved mainly by introducing redundancy on the computer system in all levels, and by using on-line self diagnostics for early fault detection. Here, the proposed scanning algorithms are good examples for the future direction.

REFERENCES

[ANSI 1986] American National Standard for Information Systems, "Small Computer System Interface (SCSI)", ANSI X3.131 - 1986, New York, NY, December 1986.

[ANSI 1994] American National Standard for Information Systems, "Small Computer System Interface (SCSI) -2", ANSI X3.131 - 1994, New York, NY, 1994. Also, “SCSI-2 Specification (Draft X3T9.2 Rev 10L)”, <http://scitexdv.com/SCSI2/Frames>, 1997.

[ANSI 1995] “Information Technology - SCSI-3 Architecture Model”, X3T10 Project 994D, <http://www.symbios.com/x3t10>, Revision 18, 27 November, 1995.

[ANSI 1996] “Information Technology - SCSI-3 Architecture Model-2”, X3T10 Project 1157D, <http://www.symbios.com/x3t10>, Revision 0.1, 2 September, 1996.

[ANSI 1997] “Information Technology - SCSI-3 Block Commands (SBC)”, X3T10 Project 996D, <http://www.symbios.com/x3t10>, Revision 8, 13 February, 1997.

[Antony 1992] Personal communication with Paul Antony, <email:antony@grok74.columbiasc.ncr.com>, April 1992.

[Beaudry 1978] M. D. Beaudry, “Performance-related Reliability Measures for Computing Systems”, IEEE Transaction on Computers, vol. C-27, June 1978, pp. 540-547.

[Bhide 1988] A. K. Bhide, "An Analysis of Architectures for High-Performance Transaction Processing", University of California, Berkeley CA, 1988, Doctoral Dissertation.

[Burkhard 1993] W. A. Burkhard, J. Menon, “Disk Array Storage System Reliability”, 23rd International Symposium on Fault Tolerant Computing, FTCS, Toulouse, France, June 1993, pp. 432-441.

[Catania 1993] V. Catania, A. Puliafito, L. Vita, "A Modeling Framework To Evaluate Performability Parameters In Gracefully Degrading Systems", IEEE Transactions on Industrial Electronics, vol. 40, no. 5, October 1993, pp. 461-472.

[Chandy 1993] J. A. Chandy, P. Banerjee, “Reliability Evaluation of Disk Array Architectures”, Coordinated Science Laboratory, University of Illinois at Urbana Champaign, 1993, p. 30.

[Chen 1988] P. M. Chen, G. Gibson, R. H. Katz, D. A. Patterson, M. Schulze, "Two Papers on RAIDs", (includes “Introduction to Redundant Arrays of Inexpensive Disks (RAID)” and “How Reliable is a RAID”, University of California, Technical Report UCB/CSD 88/479, Berkeley CA, December 1988.

[Chen 1989] P. M. Chen, "An Evaluation of Redundant Array of Disks Using an Amdahl 5890", University of California Technical Report UCB/CSD 89/506, Berkeley CA, May 1989, Master's Thesis.

[Chen 1990] P. M. Chen, "Maximizing Performance in a Striped Disk Array", Proceedings of the 17th Annual International Symposium of Computer Architecture (SIGARCH), Seattle WA, May 1990, pp. 322-331.

[Chen 1990a] P. M. Chen, G. A. Gibson, R. H. Katz, D. A. Patterson, "An Evaluation of Redundant Array of Disks Using an Amdahl 5890", Proceedings of the 1990 ACM Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), Boulder CO, May 1990.

[Chen 1992] P. M. Chen, "Input/Output Performance Evaluation: Self-Scaling Benchmarks, Predicting Performance", University of California, Technical Report UCB/CSD 92/714, Berkeley CA, November 1992, p. 124, Doctoral Thesis.

[Chervenak 1990] A. L. Chervenak, "Performance Measurements of the First RAID Prototype", University of California, Technical Report UCB/CSD 90/574, Berkeley CA, May 1990, Master's Thesis.

[Cioffi 1990] J. M. Cioffi, W. L. Abbot, H. K. Thapar, C. M. Melas, K. D. Fisher, “Adaptive Equalization in Magnetic-Disk Storage Channels”, IEEE Communications Magazine, February 1990, pp. 14-29.

[Coffman 1973] E. G. Coffman, P. J. Denning, “Operating Systems Theory”, Prentice-Hall, New York, 1973, p. 331.

[Comdisco 1989] Comdisco, “Block Oriented Network Simulator (BONes)”, Comdisco Systems Inc., product description, Foster City, California, 1989.

[Conner 1992] Conner Peripherals, “CP30084/CP30174E Intelligent Disk Drive, Product Manual”, revision B, Conner Peripherals, Inc., September 1992.

[Cox 1986] W. T. Cox, "The Performance of Disk Servers", University of Wisconsin, Madison, 1986, Doctoral Dissertation.

[Dibble 1989] P. C. Dibble, M. L. Scott, "Beyond Striping: The Bridge Multiprocessor File System", Computer Architecture News, vol. 17, no. 5, September 1989, pp. 32-39.

[DPT 1993] DPT, “Technology focus: Understanding RAID”, Distributed Processing Technology, Maitland, Florida, 1993.

[DPT 1993a] DPT, “Technology focus: RAID-1 versus RAID-5 Picking the right array for our application”, Distributed Processing Technology, Maitland, Florida, 1993.

[Faulkner 1991] Faulkner, “Hard Disk, Comparison Charts”, MIC1.0830.003, Faulkner Technical Reports Inc, 1991, p. 49.

[Fujitsu 1996] Fujitsu: “ErgoPro x653/200s”, <http://www.fujitsu.fi/Products/ErgoPro/ErgoPro-FS/dsep6013.htm>, 1996.

[Furchgott 1984] D. G. Furchgott, J. F. Meyer, “A Performability Solution Method for Nonrepairable Systems”, IEEE Transaction on Computers, vol. C-33, June 1984.

[Garcia-Molina 1988] H. Garcia-Molina, K. Salem, "The Impact of Disk Striping on Reliability", IEEE Data Engineering Bulletin, vol. 1, no. 2, 1988.

[Geist 1993] R. M. Geist, K. S. Trivedi, "An Analytic Treatment of the Reliability and Performance of Mirrored Disk Subsystems", FTCS, The 23rd Annual International Symposium on Fault-Tolerant Computing, Toulouse, France, June, 1993, pp. 442-450.

[Gibson 1989] G. A. Gibson, "Performance and Reliability in Redundant Arrays of Inexpensive Disks", University of California, Berkeley CA, Technical Report EECS, September 1989.

[Gibson 1989a] G. A. Gibson, L. Hellerstein, R. M. Karp, R. H. Katz, D. A. Patterson, "Coding Techniques for Handling Failures in Large Disk Arrays", Third International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS III), Boston MA, April 1989, pp. 123-132.

[Gibson 1991] G. A. Gibson, "Redundant Disk Arrays: Reliable, Parallel Secondary Storage", University of California, Berkeley CA, April 1991, Doctoral Dissertation.

[Gray 1993] J. Gray, "Disc Trends and Economics", Presentation held in Helsinki University of Technology, Finland, March 1993.

[Grossman 1985] C. P. Grossman, "Cache-DASD Storage Design for Improving System Performance", IBM Systems Journal, vol. 24 (3/4), 1985, pp. 316-334.

[Haseltine 1996] Phil Haseltine, <news:comp.periphs.scsi>, November 6th, 1996.

[Hillo 1992] Personal communications, Jarmo Hillo, ICL Personal Systems, Helsinki, Finland, 1992.

[Hillo 1993] J. Hillo, "The Design and Implementation of a Disk Array Host Adapter", Helsinki University of Technology, Faculty of Electrical Engineering, Espoo Finland, 1993, p. 77, Master's Thesis.

[Hillo 1994] Personal communications, Jarmo Hillo, ICL Personal Systems, Helsinki, Finland, 1994.

[Hillo 1996] Personal communications, Jarmo Hillo, ICL Personal Systems, Helsinki, Finland, 1996.

[Holland 1993] M. Holland, G. A. Gibson, “Fast, On-Line Failure Recovery in Redundant Disk Arrays”, 23rd International Symposium on Fault Tolerant Computing, FTCS, Toulouse, France, June 1993, pp. 422-431.

[Hou 1994] R. Y.-K. Hou, "Improving Reliability and Performance of Redundant Disk Arrays by Improving Rebuild Time and Response Time", University of Michigan, 1994, Doctoral Dissertation.

[IBM 1996a] IBM: “IBM leadership in disk storage technology”, <http://eagle.almaden.ibm.com/storage/technolo/grochows/grocho01.htm>, 1996.

[IBM 1996b] IBM: “IBM leadership in disk storage technology”, <http://eagle.almaden.ibm.com/storage/technolo/grochows/grocho16.htm>, 1996.

[IBM 1996c] IBM: “IBM leadership in disk storage technology”, <http://eagle.almaden.ibm.com/storage/technolo/grochows/grocho14.htm>, 1996.

[IBM 1996d] IBM: “Predictive Failure Analysis Advanced Condition Monitoring”, <http://www.storage.ibm.com/storage/oem/tech/predfail.htm>, 1996.

[Intel 1996] Intel: “iCOMP Index 2.0”, <http://www.intel.com/procs/perf/icomp/icomp_paper/ICOMP.HTM>, 1996.

[Intel 1996a] Intel: “Intel Microprocessor Quick Reference Guide”, <http://www.intel.com/pressroom/no_frame/quickref.htm>, 1996.

[Jhingran 1989] A. Jhingran, "A Performance Study of Optimization Algorithms on a Database System Supporting Procedures", University of California, Berkeley, UCB/ERL M89/15, January 1989, p. 21.

[Kamunen 1994] Personal communications, Kari Kamunen, ICL Personal Systems, Helsinki, Finland, 1992-94.

[Kamunen 1996] Personal communications, Kari Kamunen, ICL Personal Systems, Helsinki, Finland, 1996.

[Kari 1992] H. H. Kari, “Performance Measurements for SQLBase Database and ISHA Disk Array Controller”, ICL Personal Systems, Helsinki, Finland, 1992, p. 50.

[Kari 1993] H. H. Kari, H. Saikkonen, F. Lombardi, “On the Methods to Detect Latent Sector Faults of a Disk Subsystem”, International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, MASCOTS’93, Simulation Series, vol. 25, no. 1, San Diego, California, January 17-20 1993, 317-322.

[Kari 1993a] H. H. Kari, H. Saikkonen, F. Lombardi, “Detection of Defective Media in Disks”, ICYCS’93, Beijing, China, July, 1993.

[Kari 1993b] H. H. Kari, H. Saikkonen, F. Lombardi, “Detection of Defective Media in Disks”, IEEE International Workshop on Defect and Fault Tolerance in VLSI Systems, Venice, Italy, October 27-29, 1993, 49-55.

[Kari 1994] H. H. Kari, H. Saikkonen, F. Lombardi, “Detecting Latent Faults in Modern SCSI Disks”, Second International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, MASCOTS’94, Durham, North Carolina, January 31-February 2, 1994, 403-404.

[Katz 1989] R. H. Katz, J. K. Ousterhout, D. A. Patterson, P. Chen, A. Chervenak, R. Drewes, G. Gibson, E. Lee, K. Lutz, E. Miller, M. Rosenblum, "A Project on High Performance I/O Subsystem", Computer Architecture News, vol. 17, no. 5, September 1989, pp. 24-31.

[Katz 1993] R. H. Katz, P. M. Chen, A. L. Drapeau, E. K. Lee, K. Lutz, E. L. Miller, S. Seshan, D. A. Patterson, "RAID-II: Design and Implementation of a Large Scale Disk Array Controller", 1993 Symposium on Integrated Systems, 1993, University of California, Berkeley, UCB/CSD 92/705.

[Kemppainen 1991] J. Kemppainen: “Benchmarking Personal Computers”, Helsinki University of Technology, Espoo, Finland, 1991, Master’s Thesis.

[Kim 1985] M. Y. Kim, A. M. Patel, "Error-Correcting Codes for Interleaved Disks with Minimal Redundancy", IBM Computer Science Research Report, RC11185 (50403), May 1985.

[Kim 1986] M. Y. Kim, "Synchronized Disk Interleaving", IEEE Transactions on Computers, vol. C-35, no. 11, November 1986, pp. 978-988.

[Kim 1987] M. Y. Kim, A. N. Tantawi, "Asynchronous Disk Interleaving”, IBM T. J. Watson Research Center TR RC 12497 (#56190), Yorktown Heights, NY, February 1987.

[Kim 1991] M. Y. Kim, A. N. Tantawi, "Asynchronous Disk Interleaving: Approximating Access Delays", IEEE Transactions on Computers, vol. 40, no. 7, July 1991, pp. 801-810.

[King 1987] R. P. King, "Disk Arm Movement in Anticipation of Future Requests", IBM Computer Science Research Report, December 1987.

[Koch 1987] P. D. L. Koch, "Disk File Allocation Based on the Buddy System", ACM Transactions on Computer Systems, vol. 5, no. 4, November 1987, pp. 352-370.

[Koolen 1992] Personal communication with Adrie Koolen, <email:adrie@ica.philips.nl”, April 1992.

[Kuhn 1997] K. J. Kuhn, “Magnetic Recording - an introduction”, <http://www.ee.washington.edu/conselec/CE/kuhn/magtape/95x1.htm>, March, 1997.

[Laininen 1995] Personal communication, Pertti Laininen, Helsinki University of Technology, Espoo, Finland, 1995.

[Lee 1990] E. K. Lee, "Software and Performance Issues in the Implementation of a RAID Prototype", University of California, Technical Report UCB/CSD 90/573, Berkeley CA, May 1990, Master's Thesis.

[Lee 1991] E. K. Lee, R. H. Katz, "Performance Consequences of Parity Placement in Disk Arrays", Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS IV), Palo Alto CA, April 1991.

[Lee 1992] E. K. Lee, P. M. Chen, J. H. Hartman, A. L. Drapeau, E. L. Miller, R. H. Katz, G. A. Gibson, D. A. Patterson, "RAID-II: A Scalable Storage Architecture for High Bandwidth Network File Service”, Technical Report UCB/CSD 92/672, University of California, Berkeley, February 1992.

[Livny 1987] M. Livny, S. Khoshafian, H. Boral, "Multi-Disk Management Algorithms", Proceedings of the 1987 ACM Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), May 1987.

[MacWilliams 1977] F. J. MacWilliams, N. J. A. Sloane, "The Theory of Error-Correcting Codes", North-Holland Mathematical Library, vol. 16, Elsevier Science Publishing Company, New York NY, 1977.

[McGregor 1992] Personal communication with Cecil Harry McGregor, <email:cecil@sunfse.ese.lmsc.lockheed.com”, April 1992.

[Meyer 1980] J. F. Meyer, "On evaluating the performability of degradable computing systems", IEEE

Transactions on Computers, vol. C-29, August 1980, pp. 720-731.

[Miller 1991] E. L. Miller, "Input/Output Behavior of Supercomputing Applications", University of California, Technical Report UCB/CSD 91/616, January 1991, Master's Thesis.

[Milligan 1994] G. E. Milligan: “Magnetic Disc Technology Trends”, Seagate, February 1994.

[Mourad 1993] A. N. Mourad, W. K. Fuchs, D. G. Saab, “Recovery Issues in Databases Using Redundant Disk Arrays”, Journal of Parallel and Distributed Computing, January 1993, p. 25.

[Mourad 1993a] A. N. Mourad, W. K. Fuchs, D. G. Saab, “Assigning Sites to Redundant Clusters in a Distributed Storage System”, University of Illinois, Urbana, 1993, p. 23.

[Muntz 1990] R. R. Muntz, J. C. S. Lui, "Performance Analysis of Disk Arrays under Failure", Proceedings of the 16th International Conference of Very Large Data Bases (VLDB), D. McLeod, R. Sacks-Davis, H. Schek (Eds.), Morgan Kaufmann Publishers, August 1990, pp. 162-173.

[Nelson 1988] M. N. Nelson, B. B. Welch, J. K. Ousterhout, "Caching in the Sprite Network File System", ACM Transactions on Computer Systems, vol. 6, no. 1, February 1988.

[Ng 1991] S. W. Ng, "Improving Disk Performance Via Latency Reduction", IEEE Transactions on Computers, vol. 40, no. 1, January 1991, pp. 22-30.

[Nilsson 1993] Personal fax communication with Centh Nilsson, Micropolis, May 1993.

[Nokia 1986] “MikroMikko 3TT, technical description”, Nokia Informaatiojärjestelmät, Helsinki, Finland, 1986, in Finnish.

[Novell 1997] “SFT III for IntranetWare: Detailed Information”, Novell Corporation, <http://www.novell.com/catalog/bg/bge24110.html>1997.

[Olson 1989] T. M. Olson, "Disk Array Performance in a Random IO Environment", Computer Architecture News, vol. 17, no. 5, September 1989, pp. 71-77.

[Orji 1991] C. U. Orji, "Issues in High Performance Input/Output Systems", University of Illinois at Chicago, Illinois, 1991, Doctoral Dissertation.

[Ottem 1996] E. Ottem, J. Plummer, “Playing it S.M.A.R.T.: Emergence of Reliability Prediction Technology”, <http://www.seagate.com/corp/techsupp/smart.shtml>, 1996

[Ousterhout 1985] J. K. Ousterhout, H. De Costa, D. Harrison, J. A. Kunze, M. Kupfer, J. G. Thompson, "A Trace-Driven Analysis of the UNIX 4.2 BSD File System", Proceedings of the Tenth ACM Symposium on Operating System Principles (SOSP), ACM Operating Systems Review, vol. 19, no. 5, December 1985, pp. 15-24.

[Ousterhout 1988] J. K. Ousterhout, F. Douglis, "Beating the I/O Bottleneck: A Case for Log-Structured File Systems", University of California, Technical Report UCB/CSD 88/467, October 1988, p. 17.

[Pages 1986] A. Pages, M. Gondran, “System Reliability: Evaluation & Prediction in Engineering”, North Oxford Academic, 1986, p. 351.

[Patterson 1987] D. A. Patterson, G. A. Gibson, R. H. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)", University of California, Technical Report UCB/CSD 87/391, Berkeley CA, December 1987.

[Patterson 1988] D. A. Patterson, G. A. Gibson, R. H. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)", Proceedings of the 1988 ACM Conference on Management of Data (SIGMOD), Chicago IL, June 1988, pp. 109-116.

[Pattipati 1993] K. R. Pattipati, Y. Li, H. A. P. Blom, “A Unified Framework for the Performability Evaluation of Fault-Tolerant Computer Systems”, IEEE Transactions on Computers, vol. 42, no. 3, March 1993, pp. 312-326.

[Pawlikowski 1990] K. Pawlikowski, "Steady-State Simulation of Queueing Processes: A Survey of Problems and Solutions", ACM Computing Surveys, vol. 22, no. 2, June, 1990.

[PCMagazine 1996] PC Magazine, “Service and Reliability Survey 1996”,
<URL:http://www.zdnet.com/pcmaguk/sandr/1996.html”

[Peterson 1972] W. W. Peterson, E. J. Weldon, Jr, "Error-Correcting Codes", MIT Press, 1972.

[Platt 1992] Personal communication with Dave Platt, <email:dplatt@ntg.com”, April 1992.

[Quantum 1996a] Quantum: “Development of 800,000 hour MTBF High Capacity Disk Drives”,
http://www.quantum.com/products/whitepapers/MTBF/index.html, 1996

[RAB 1993] The RAID Advisory Board, “The RAIDBook: A Source Book for RAID Technology”, Edition 1-1, The RAID Advisory Board, St. Peter, Minnesota, November 1993.

[Reddy 1989] A. L. Reddy, P. Banerjee, "Evaluation of Multiple-Disk I/O Systems", IEEE Transactions on Computers, vol. 38, no. 12, December 1989, pp. 1680-1690.

[Reddy 1990] A. L. Reddy, P. Banerjee, "A Study of I/O Behavior or Perfect Benchmarks on a Multiprocessors", Proceedings IEEE 17th International Symposium Computer Architecture Conference, Seattle, Washington, May 1990, pp. 312-317.

[Reddy 1990a] A. L. Reddy, "Parallel Input/Output Architectures for Multiprocessors", University of Illinois at Urbana-Champaign, Urbana, 1990, Doctoral Dissertation.

[Reddy 1991] A. L. Reddy, P. Banerjee, "Gracefully Degradable Disk Arrays ", FTCS-91, 1991.

[Reddy 1991a] A. L. Reddy, P. Banerjee, "A Study of Parallel Disk Organizations", Computer Architecture News, vol. 17, no. 5, September 1989, pp. 40-47.

[Rosenblum 1992] M. Rosenblum, "The Design and Implementation of a Log-structured File System", University of California, Berkeley, UCB/CSD 92/696, June 1992, p. 82.

[Räsänen 1994] Personal communications, Olli-Pekka Räsänen, ICL Personal Systems, Helsinki, Finland, 1992-94.

[Räsänen 1996] Personal communications, Olli-Pekka Räsänen, ICL Personal Systems, Helsinki, Finland, 1996.

[Sahner 1986] R. A. Sahner, K. S. Trivedi, "Sharpe: Symbolic Hierarchical Automated Reliability and Performance Evaluator, Introduction and Guide for Users", Department of Computer Science, Duke University, September 1986.

[Sahner 1987] R. A. Sahner, K. S. Trivedi, "Reliability Modeling using SHARPE", IEEE Transactions on Reliability, vol. R-36, no. 2, June 1987, pp. 186-193.

[Saleh 1990] A. M. Saleh, J. J. Serrano, J. H. Patel, “Reliability of Scrubbing Recovery-Techniques for Memory Systems”, IEEE Transactions on Reliability, vol. 39, no. 1, April 1990, pp. 114-122.

[Salem 1986] K. Salem, H. Garcia-Molina, "Disk Striping", Proceedings of the 2nd IEEE International Conference on Data Engineering, 1986, pp. 336-342.

[Schulze 1988] M. E. Schulze, "Considerations in the Design of a RAID Prototype", University of California, Technical Report UCB/CSD 88/448, August 1988, p. 35.

[Schwarz 1994] T. J. E. Schwarz, "Reliability and Performance of Disk Arrays", University of California, San Diego, 1994, Doctoral Dissertation.

[Scritsmier 1996] Personal communication with Milton Scritsmier, <email:milton@arraytech.com>, November 1996.

[Seagate 1992] Seagate, “ST3600N/NDFamily, ST3500N/ND, ST3600N/ND, SCSI-2: Product Manual, Volume 1”, Seagate, Publication number 77738477-A, October 1992.

[Seagate 1992a] Seagate, “Disc Drive SCSI-2 Interface Family Models: ST11200N/ND, 12400N/ND, 3600N, ST31200N/ND, ST11750N/ND, ST12550N/ND: Product Manual, Volume 2; Version 2”, Seagate, Publication number 77738479-A, December 1992.

[Seagate 1996a] Seagate: “Fibre Channel Arbitrated Loop (FC-AL)”,
<http://www.seagate.com/corp/techsupp/fibfs.shtml>, 1996.

[Seagate 1996b] Seagate, Tim Sutton: “Barracuda 2 2HP: Parallel processing for storage devices”,
<http://www.seagate.com/corp/techsupp/cuda2.shtml>, 1996.

[Seagate 1996c] Seagate: “Barracuda 4LP Family”,
<http://www.seagate.com/new/cuda41p.shtml>, 1996.

[Seltzer 1990] M. I. Seltzer, P. M. Chen, J. K. Ousterhout, "Disk Scheduling Revisited", Proceedings of the Winter 1990 USENIX Technical Conference, Washington DC, January 1990.

[Seltzer 1990a] M. I. Seltzer, M. Stonebraker, "Transaction Support in Read Optimized and Write Optimized File Systems", Proceedings of the 16th International Conference on Very large Data Bases, VLDB, August 1990, pp. 174-185.

[Seltzer 1992] M. Seltzer, M. Stonebraker, "Read Optimized File System Designs: A Performance Evaluation", University of California, Technical Report UCB/CSD 92/64, June 1992, p. 24.

[Seltzer 1993] M. Seltzer, "File System Performance and Transaction Support", University of California, Technical Report UCB/CSD 93/1, January 1993, p. 118.

[Shooman 1968] M. L. Shooman, “Probabilistic Reliability: An Engineering Approach”, New York, McGraw-Hill, 1968.

[Sierra 1990] H. M. Sierra, "An Introduction to Direct Data Storage Devices", Academic Press, 1990.

[Siewiorek 1982] D. P. Siewiorek, R. S. Swarz, "The Theory and Practice of Reliable System Design", Digital Press, 1982.

[Smith 1988] R. M. Smith, K. S. Trivedi, A. V. Ramesh "Performability analysis: Measures, an algorithm and a case study", IEEE Transaction on Computers, vol. C_37, no. 4, April 1988, pp. 406-417.

[Stevens 1995] L. Stevens, "Hierarchical Storage Management”, Open Computing, May 1995, pp. 71-73.

[Stonebraker 1988] M. R. Stonebraker, R. Katz, D. Patterson, J. Ousterhout, "The Design of XPRS”, University of California, UCB/ERL M88/19 March 1988, p. 20.

[Stonebraker 1989] M. R. Stonebraker, G. A. Schloss, "Distributed RAID - A New Multiple Copy Algorithm", Proceedings of the 6th IEEE International Conference on Data Engineering, April 1990 also

University of California, UCB/ERL M89/56 May 1989, p. 22.

[T10 1997] “T10 Working Drafts Online”, <http://www.symbios.com/x3t10/drafts.htm>, February, 1997.

[Thiebaut 1992] D. Thiebaut, H. S. Stone, J. L. Wolf, "Improving Disk cache hit-ratios Through Cache Partitioning", IEEE Transactions on Computers, vol. 41, no. 6, June 1992, pp. 665-676.

[TPC 1992] Transaction Processing Performance Council (TPC): “TPC Benchmark B”, Standard specification, revision 1.1, March 1992, San Jose, California, p. 36.

[TPC 1992a] Transaction Processing Performance Council (TPC): “TPC Benchmark A”, Standard specification, revision 1.1, March 1992, San Jose, California, p. 37.

[Trivedi 1994] K. S. Trivedi, M. Malhotra, R. M. Fricks, “Markov Reward Approach to Performability and Reliability”, Second International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, MASCOTS’94, Durham, North Carolina, January 31- February 2, 1994, pp. 7-11.

[Varhol 1991] P. D. Varhol, "Gigabytes online", Personal workstation, June 1991, pp. 44-49.

[Voutilainen 1996] Petri Voutilainen, <news:sfnet.atk.laitteet>, December 17, 1996.

[Williams 1988] T. Williams, "Winchester drive reliability tracks capacity and performance gains", Computer Design, February, 1988, pp. 49-55.

[Ylinen 1994] Personal communications, Ville Ylinen, ICL Personal Systems, Helsinki, Finland, 1992-94.

Appendix A: Solving EMM1 and EMM2 equations

This appendix presents the listings of two Maple V programs (versions 2 and 4) in solving the Markov models illustrated in Chapter 7. Two Maple V versions were used as version 2 had better equation solving features, but version 4 could handle also Greek alphabets.

The principle to solve EMM1 analytically is as follows:

1. Maple V version 2 is used for solving the transient state equations of EMM1 using a set of differential equations and Maple V's equation solving tools.
2. From the results, the relevant parameters are extracted (such as dividends, divisor, and parameters for the cubic root equation).
3. Those parameters are then written to the file F_EMM0_T.
4. This file is read into Maple V version 4. Here, the actual parameters are inserted and the equations are ready to be inserted into Chapter 7.
5. The file F_EMM0_T is also used for converting the equations into Excel files that are used for drawing the charts in Chapter 8.

The principle to solve EMM1 using approximation is as follows:

1. Maple V version 2 is used for solving the steady state equations of EMM1 using a set of equations and Maple V's equation solving tools.
2. From the results, the relevant parameters are extracted (such as dividends and a divisor).
3. Those parameters are then written to the file F_EMM1_T.
4. This file is read into Maple V version 4. Here, the actual parameters are inserted and the equations are ready to be inserted into Chapter 7.
5. The file F_EMM1_T is also used for converting the equations into Excel files that are used for drawing the charts in Chapter 8.

The principle to solve EMM2 using approximation is as follows:

1. Maple V version 2 is used for solving the steady state equations of EMM2 using a set of equations and Maple V's equation solving tools.
2. From the results, the relevant parameters are extracted (such as dividends and a divisor).
3. Those parameters are then written to the file F_EMM2_T.

4. This file is read into Maple V version 4. Here, the dividends of all states are processed, and simpler representations are found for those dividends. After that, the actual parameters are inserted and the equations are ready to be inserted into Chapter 7.
5. The file F_EMM2_T is also used for converting the equations into Excel files that are used for drawing the charts in Chapter 8.

Analysis of EMM1 with Maple V version 2

```
> #
> # Solving EMM1 with exact analysis.
> #
> # This is Maple V version 2 part of the analysis. Here, EMM1 is solved in
> # the transient state using equation solving functions of Maple V.
> #
> # As Maple V version 2 is not capable to handle Greek alphabets nor
> # subscripts, they are replaced with normal letters as follows:
> #
> #      G:=(D+1)*S*lambda[s]
> #      H:=mu[s]
> #      K:=(D+1)*lambda[d]
> #      J:=mu[d]
> #      L:=lambda[d]
> #      M =D(lambda[s]+lambda[d])
> #      N =D(S*lambda[s]+lambda[df])
> #
> # where square brackets indicate the subscript.
> #
> # Transient state equations of EMM1.
> #
> SolutionSet := dsolve({
              diff(p00(t),t)=-(G+K)*p00(t)+H*p01(t)+J*p10(t),
              diff(p01(t),t)=-(H+L+M)*p01(t)+G*p00(t),
              diff(p10(t),t)=-(J+N)*p10(t)+K*p00(t)+L*p01(t),
              diff(pf(t),t)=p01(t)*M+p10(t)*N,
              p00(0)=1,p01(0)=0,p10(0)=0,pf(0)=0 },
  {p00(t),p01(t),p10(t),pf(t)},
  laplace);
```

$$\begin{aligned}
&\mathit{SolutionSet} := \{\mathrm{pf}(t) = 1 + \Bigg(\sum_{\_r = \%1} (-N\,H - G\,J - G\,N - J\,L - K\,L - N\,M - J\,M - K\,M - K\,H - J\,H - G\,L \\
&\quad - N\,L - K\,\_r - J\,\_r - \_r\,H - \_r\,L - N\,\_r - \_r\,M - G\,\_r - \_r^2)\,\mathbf{e}^{(\_r\,t)} \Big/ (2\,\_r\,M + 2\,G\,\_r + 2\,K\,\_r + 2\,J\,\_r \\
&\quad + 2\,N\,\_r + 2\,\_r\,H + K\,M + 3\,\_r^2 + K\,L + J\,H + J\,L + J\,M + G\,J + N\,M + G\,N + 2\,\_r\,L + N\,K + N\,H + G\,L + G\,M \\
&\quad + N\,L + K\,H)\Bigg),\ \mathrm{p10}(t) = \sum_{\_r = \%1} (G\,L + K\,H + K\,L + K\,M + K\,\_r)\,\mathbf{e}^{(\_r\,t)} \Big/ (2\,\_r\,M + 2\,G\,\_r + 2\,K\,\_r \\
&\quad + 2\,J\,\_r + 2\,N\,\_r + 2\,\_r\,H + K\,M + 3\,\_r^2 + K\,L + J\,H + J\,L + J\,M + G\,J + N\,M + G\,N + 2\,\_r\,L + N\,K + N\,H \\
&\quad + G\,L + G\,M + N\,L + K\,H),\ \mathrm{p01}(t) = \sum_{\_r = \%1} G\,(\_r + J + N)\,\mathbf{e}^{(\_r\,t)} \Big/ (2\,\_r\,M + 2\,G\,\_r + 2\,K\,\_r + 2\,J\,\_r \\
&\quad + 2\,N\,\_r + 2\,\_r\,H + K\,M + 3\,\_r^2 + K\,L + J\,H + J\,L + J\,M + G\,J + N\,M + G\,N + 2\,\_r\,L + N\,K + N\,H + G\,L + G\,M \\
&\quad + N\,L + K\,H),\ \mathrm{p00}(t) = \sum_{\_r = \%1} (H + L + M + \_r)\,(\_r + J + N)\,\mathbf{e}^{(\_r\,t)} \Big/ (2\,\_r\,M + 2\,G\,\_r + 2\,K\,\_r + 2\,J\,\_r \\
&\quad + 2\,N\,\_r + 2\,\_r\,H + K\,M + 3\,\_r^2 + K\,L + J\,H + J\,L + J\,M + G\,J + N\,M + G\,N + 2\,\_r\,L + N\,K + N\,H + G\,L + G\,M \\
&\quad + N\,L + K\,H)\}
\end{aligned}$$

$$\begin{aligned}
&\%1 := \mathrm{RootOf}(\_Z^3 + (K + J + H + L + N + M + G)\,\_Z^2 \\
&\quad + (G\,J + N\,K + G\,N + N\,M + K\,M + J\,M + K\,L + N\,L + K\,H + J\,L + G\,L + N\,H + J\,H + G\,M)\,\_Z + J\,G\,M \\
&\quad + N\,K\,H + N\,G\,L + N\,G\,M + N\,K\,L + N\,K\,M)
\end{aligned}$$

```
> #
> # Results of the equations are assigned to given variables.
> #
> assign(SolutionSet);
> #
> # Extract the parameters from the equations.
> #
> divisor:= 1/op(3,op(1,p00(t))):
> rof :=op(1,op(2,op(2,p00(t)))):
> p00_dividend := op(1,op(1,p00(t)))*op(2,op(1,p00(t))):
> p01_dividend := op(1,op(1,p01(t)))*op(2,op(1,p01(t))):
> p10_dividend := op(1,op(1,p10(t))):
> pf_dividend:=op(1,op(1,op(2,pf(t)))):
> #
> # Results of this process (variables p00_dividend,p01_dividend,
> # p10_dividend,pf_dividend,divisor,rof) are copied to file f_emm0_t.
> #
> save p00_dividend,p01_dividend,p10_dividend,pf_dividend,divisor,rof,f_emm0_t;
> #
> # Further processing is done in Maple V version 4.
> #
```

Analysis of EMM1 with Maple V version 4

```
> #
> # Solving EMM1 with exact analysis.
> #
> # This is Maple V version 4 part of the analysis. Here, the results of the
> # solved equations are manipulated and simplified.
> #
> # Undefine constants used in equations.
> #
> G:='G': # G:=(D+1)*S*lambda[s]
> H:='H': # H:=mu[s]
> K:='K': # K:=(D+1)*lambda[d]
> J:='J': # J:=mu[d]
> L:='L': # L:=lambda[d]
> M:='M': # M =D*(lambda[s]+lambda[d])
> N:='N': # N =D*(S*lambda[s]+lambda[df])
> #
> # Information, produced by Maple V version 2 is extracted from file F_EMM0_T.
> #
> p00_dividend := (_r+H+L+M)*(_r+J+N):
> p01_dividend := G*(_r+J+N):
> p10_dividend := G*L+K*_r+K*H+K*L+K*M:
> pf_dividend := -J*L-K*M-G*J-N*L-K*L-G*N-J*H-J*M-N*M-G*L-K*H-N*H-G*_r-
  _r*M-K*_r-N*_r-J*_r-_r*L-_r*H-_r^2:
> divisor := G*L+G*M+K*H+K*L+K*M+J*H+J*L+J*M+G*J+N*H+N*L+N*M+G*N+N*K+
  3*_r^2+2*N*_r+2*J*_r+2*_r*L+2*_r*H+2*_r*M+2*G*_r+2*K*_r:
> rof := _Z^3+(G+M+K+N+J+L+H)*_Z^2+(J*L+G*J+N*L+K*L+K*M+J*H+G*M+
  J*M+N*M+G*N+G*L+K*H+N*H+N*K)*_Z+J*G*M+N*K*L+N*G*L+N*K*H+N*K*M+N*G*M:
> #
> # Now, it is possible to change the parameters to Greek alphabets.
> #
> G:=(D+1)*S*lambda[s]:
> H:=mu[s]:
> K:=(D+1)*lambda[d]:
> J:=mu[d]:
> L:=lambda[d]:
> M:=D*(lambda[s]+lambda[d]):
> N:=D*(S*lambda[s]+lambda[df]):
> _r:=r[i]:
> #
> # p00
> #
> p00:=sum(p00_dividend/QI,r[i]);
```

$$p00 := \mathrm{sum}\left(\frac{(r_i+\mu_s+\lambda_d+D(\lambda_s+\lambda_d))\,(r_i+\mu_d+D(S\,\lambda_s+\lambda_{df}))}{QI},\, r_i\right)$$

```
> #
> # p01
> #
> p01:=sum(p01_dividend/QI,r[i]);
```

$$p01 := \mathrm{sum}\left(\frac{(D+1)\,S\,\lambda_s\,(r_i+\mu_d+D\,(S\,\lambda_s+\lambda_{df}))}{QI},\,r_i\right)$$

```
> #
> # p10
> #
> p10:=sum(p10_dividend/QI,r[i]);
```

$$\begin{aligned}&p10 := \mathrm{sum}((\\&\quad(D+1)\,S\,\lambda_s\,\lambda_d+(D+1)\,\lambda_d\,r_i+(D+1)\,\lambda_d\,\mu_s+(D+1)\,\lambda_d^2+(D+1)\,\lambda_d\,D\,(\lambda_s+\lambda_d)\\&\quad)\,/\,QI,\,r_i)\end{aligned}$$

```
> #
> # pf
> #
> pf:=sum(pf_dividend/QI,r[i]);
```

$$\begin{aligned}&pf := \mathrm{sum}((-\mu_d\,\lambda_d-(D+1)\,\lambda_d\,D\,\%2-(D+1)\,S\,\lambda_s\,\mu_d-D\,\%1\,\lambda_d-(D+1)\,\lambda_d^2\\&\quad-(D+1)\,S\,\lambda_s\,D\,\%1-\mu_d\,\mu_s-\mu_d\,D\,\%2-D^2\,\%1\,\%2-(D+1)\,S\,\lambda_s\,\lambda_d-(D+1)\,\lambda_d\,\mu_s\\&\quad-D\,\%1\,\mu_s-(D+1)\,S\,\lambda_s\,r_i-r_i\,D\,\%2-(D+1)\,\lambda_d\,r_i-D\,\%1\,r_i-\mu_d\,r_i-r_i\,\lambda_d-r_i\,\mu_s\\&\quad-r_i^2)\,/\,QI,\,r_i)\\&\quad\%1 := S\,\lambda_s+\lambda_{df}\\&\quad\%2 := \lambda_s+\lambda_d\end{aligned}$$

```
> #
> # Where the divisor (QI) is equal to
> #
> divisor;
```

$$\begin{aligned}&\lambda_d\,(D+1)\,S\,\lambda_s+(D+1)\,S\,\lambda_s\,D\,\%1+(D+1)\,\lambda_d\,\mu_s+\lambda_d^2\,(D+1)+(D+1)\,\lambda_d\,D\,\%1\\&\quad+\mu_d\,\mu_s+\mu_d\,\lambda_d+\mu_d\,D\,\%1+\mu_d\,(D+1)\,S\,\lambda_s+D\,\%2\,\mu_s+D\,\%2\,\lambda_d+D^2\,\%2\,\%1\\&\quad+(D+1)\,S\,\lambda_s\,D\,\%2+D\,\%2\,(D+1)\,\lambda_d+3\,r_i^2+2\,D\,\%2\,r_i+2\,\mu_d\,r_i+2\,r_i\,\lambda_d+2\,r_i\,\mu_s\\&\quad+2\,r_i\,D\,\%1+2\,(D+1)\,S\,\lambda_s\,r_i+2\,(D+1)\,\lambda_d\,r_i\\&\quad\%1 := \lambda_s+\lambda_d\\&\quad\%2 := S\,\lambda_s+\lambda_{df}\end{aligned}$$

```
> #
> # and r[i]'s are the roots of the following equation
> #
> rof;
```

$$\begin{aligned}
&\_Z^3 + ((D+1)\,S\,\lambda_s + D\,\%1 + (D+1)\,\lambda_d + D\,\%2 + \mu_d + \lambda_d + \mu_s)\,\_Z^2 + (\mu_d\,\lambda_d \\
&\quad + \mu_d\,(D+1)\,S\,\lambda_s + D\,\%2\,\lambda_d + \lambda_d^2\,(D+1) + (D+1)\,\lambda_d\,D\,\%1 + \mu_d\,\mu_s \\
&\quad + (D+1)\,S\,\lambda_s\,D\,\%1 + \mu_d\,D\,\%1 + D^2\,\%2\,\%1 + (D+1)\,S\,\lambda_s\,D\,\%2 + \lambda_d\,(D+1)\,S\,\lambda_s \\
&\quad + (D+1)\,\lambda_d\,\mu_s + D\,\%2\,\mu_s + D\,\%2\,(D+1)\,\lambda_d)\,\_Z + \mu_d\,(D+1)\,S\,\lambda_s\,D\,\%1 \\
&\quad + D\,\%2\,(D+1)\,\lambda_d^2 + D\,\%2\,(D+1)\,S\,\lambda_s\,\lambda_d + D\,\%2\,(D+1)\,\lambda_d\,\mu_s \\
&\quad + D^2\,\%2\,(D+1)\,\lambda_d\,\%1 + D^2\,\%2\,(D+1)\,S\,\lambda_s\,\%1 \\
&\quad \%1 := \lambda_s + \lambda_d \\
&\quad \%2 := S\,\lambda_s + \lambda_{df}
\end{aligned}$$

```
> #
> # End of EMM1 analysis.
> #
```

Analysis of EMM1A with Maple V version 2

```
> #
> # Solving EMM1 with approximation (EMM1A).
> #
> # This is Maple V version 2 part of the analysis. Here, EMM1A is solved in
> # the steady state using equation solving functions of Maple V.
> #
> # As Maple V version 2 is not capable to handle Greek alphabets nor
> # subscripts, they are replaced with normal letters as follows:
> #
> #      G:=(D+1)*S*lambda[s]
> #      H:=mu[s]
> #      K:=(D+1)*lambda[d]
> #      J:=mu[d]
> #      L:=lambda[d]
> #
> # where square brackets indicate the subscript;
> #
> # Steady state equations of EMM1A
> #
> SolutionSet := solve({
            -(G+K)*p00+H*p01+J*p10=0,
            -(H+L)*p01+G*p00=0,
            -(J)*p10+K*p00+L*p01=0,
            p00+p01+p10=1  },
  {p00,p01,p10}
  );
```

$$\mathit{SolutionSet} := \{\mathit{p10} = \frac{H\,K + L\,G + L\,K}{L\,G + L\,K + L\,J + G\,J + H\,J + H\,K},\ \mathit{p01} = \frac{G\,J}{L\,G + L\,K + L\,J + G\,J + H\,J + H\,K},$$

$$\mathit{p00} = \frac{J\,(H+L)}{L\,G + L\,K + L\,J + G\,J + H\,J + H\,K}\}$$

```
> #
> # Results of the equations are assigned to given variables.
> #
> assign(SolutionSet);
> #
> # The temporary variable is copied to new variable "divisor".
> #
> divisor := 1/op(2,p10):
> #
> # Results of this process (variables p00,p01,p10,divisor) are copied
> # to file f_emm1_t.
> #
> save p00,p01,p10,divisor,f_emm1_t;
> #
> #  Further processing is done in Maple V version 4.
> #
```

Analysis of EMM1A with Maple V version 4

```
> #
> # Solving EMM1 with approximation (EMM1A).
> #
> # This is Maple V version 4 part of the analysis. Here, results of the solved
> # equations are manipulated and simplified.
> #
> # Undefine constants used in equations:
> #
> G:='G': # G:=(D+1)*S*lambda[s]
> H:='H': # H:=mu[s]
> K:='K': # K:=(D+1)*lambda[d]
> J:='J': # J:=mu[d]
> L:='L': # L:=lambda[d]
> #
> # Information, produced by Maple V version 2 is extracted from file F_EMM1_T.
> #
> p00 := J*(H+L)/(L*G+L*K+L*J+G*J+H*J+H*K):
> p01 := G*J/(L*G+L*K+L*J+G*J+H*J+H*K):
> p10 := (L*K+L*G+H*K)/(L*G+L*K+L*J+G*J+H*J+H*K):
> divisor := L*G+L*K+L*J+G*J+H*J+H*K:
> #
> # Let q00 to be the dividend of p00 (and similarly for p01, p10)
> #
> q00:=p00*divisor:
> q01:=p01*divisor:
> q10:=p10*divisor:
> #
> # Now check the consistency: (q00+q01+q10) should be equal to divisor.
> #
> simplify(q00+q01+q10-divisor);
```

$$0$$

```
> #
> # Consistency checked and OK
> #
> # Now, it is possible to change the parameters to Greek alphabets.
> #
> G:=(D+1)*S*lambda[s]:
> H:=mu[s]:
> K:=(D+1)*lambda[d]:
> J:=mu[d]:
> L:=lambda[d]:
> #
> # p00
> #
> q00/QI;
```

$$\frac{\mu_d\,(\mu_s+\lambda_d)}{QI}$$

```
> #
> # p01
> #
> q01/QI;
```

$$\frac{(D+1)\,S\,\lambda_s\,\mu_d}{QI}$$

```
> #
> # p10
> #
> q10/QI;
```

$$\frac{\lambda_d{}^2\,(D+1)+\lambda_d\,(D+1)\,S\,\lambda_s+\mu_s\,(D+1)\,\lambda_d}{QI}$$

```
> #
> # Where QI is the divisor, equals
> #
> divisor;
```

$$\lambda_d\,(D+1)\,S\,\lambda_s+\lambda_d{}^2\,(D+1)+\lambda_d\,\mu_d+(D+1)\,S\,\lambda_s\,\mu_d+\mu_s\,\mu_d+\mu_s\,(D+1)\,\lambda_d$$

```
> #
> # Final check (dividends divided by divisor should equal one).
> #
> simplify((q00+q01+q10)/divisor);
```

$$1$$

```
> #
> # End of EMM1A analysis.
> #
```

Analysis of EMM2A with Maple V version 2

```
> #
> # Solving EMM2 with approximation (EMM2A).
> #
> # This is Maple V version 2 part of the analysis. Here, EMM2A is solved in
> # the steady state using equation solving functions of Maple V.
> #
> # As Maple V version 2 is not capable to handle Greek alphabets nor
> # subscripts, they are replaced with normal letters as follows:
> #
> #      G:=(D+1)*S*lambda[s]
> #      H:=mu[s]
> #      K:=(D+1)*lambda[d]
> #      J:=mu[d]
> #      L:=lambda[d]
> #      P:=lambda[sd]
> #      Q:=mu[sd]
> #      R:=mu[dr]
> #
> # where square brackets indicate the subscript.
> #
> # Steady state equations of EMM2A
> #
> SolutionSet := solve({
             -p000*(G+K+P)+p001*H+p100*Q=0,
             -p001*(H+L+P)+p000*G+p101*Q=0,
             -p010*(J+L)+p000*K+p001*L+p110*R=0,
             -p100*(G+K+Q)+p000*P+p010*J+p101*H=0,
             -p101*(H+L+Q)+p001*P+p100*G=0,
             -p110*(R)+p010*L+p100*K+p101*L=0,

             p000+p001+p010+p100+p101+p110=1 },
  {p000,p001,p010,p100,p101,p110}
  );
```

$$\mathit{SolutionSet} := \{\mathit{p000} = \frac{Q J R (G H + H^2 + 2 H L + Q H + P H + L^2 + Q L + P L)}{\%1}, \mathit{p100} = J R (L G P + Q P H$$

$$+ L^2 P + K Q H + K P H + 2 H P L + H G L + 2 L K H + Q P L + Q G L + Q K L + G H P + P H^2 + H P^2 + L P^2$$

$$+ K P L + L^2 K + L^2 G + K H^2) / (\%1), \mathit{p101} = \frac{R G J (G L + G P + K H + K L + K P + P H + P L + P^2 + Q P)}{\%1},$$

$$\mathit{p010} = (G^2 Q L + Q^2 G L + Q^2 K L + L^2 Q K + G L P^2 + G L^2 P + Q L^2 G + Q H G L + 2 L Q K H + Q^2 K H$$

$$+ K L P^2 + G^2 L^2 + K^2 H^2 + K^2 H P + 2 Q P K H + G Q K H + K H G P + K H P^2 + K^2 Q H + Q H^2 K$$

$$+ K H^2 P + K^2 L^2 + G^2 L P + 2 G L^2 K + K^2 Q L + 2 K^2 H L + K^2 L P + K L^2 P + 2 G L K H + 2 G L K P$$

$$+ G L P H + 2 G L Q P + 2 K Q G L + 2 K Q P L + 2 K H P L) R / (\%1), \mathit{p110} = (L^2 Q^2 K + L^2 Q^2 G$$

$$+ 2 K L^3 G + 2 K^2 L^2 H + K^2 L^2 P + K L^3 P + K L^2 P^2 + Q L^3 G + Q L^3 K + G^2 L^2 P + G L^3 P + G L^2 P^2$$

$$+ L^2 G^2 Q + L K^2 H^2 + L^2 K^2 Q + J G^2 L^2 + J K^2 H^2 + J K^2 L^2 + K^2 L^3 + G^2 L^3 + 2 L^2 Q P G$$

$$+ 2 L Q P K H + 2 L^2 Q P K + L Q^2 K H + 2 K L^2 G P + 2 K L^2 P H + 2 Q L^2 K H + 2 G L^2 K H + G L^2 P H$$

$$+ L G Q K H + 2 L^2 G Q K + L K H G P + L K^2 H P + L K H^2 P + L K H P^2 + L K^2 Q H + L^2 Q H G$$

$$+ L Q H^2 K + J G^2 L P + 2 J G L K H + 2 J G L^2 K + 2 J G L K P + J G L P H + J G L^2 P + J G L P^2$$

$$+ J G L Q P + J K Q G L + J K^2 Q H + J K^2 Q L + J K Q P H + J K Q P L + J K H G P + 2 J K^2 H L + J K^2 H P$$

$$+ J K H^2 P + 2 J K H P L + J K H P^2 + J K^2 L P + J K L^2 P + J K L P^2) / (\%1),$$

$$\mathit{p001} = \frac{Q J G R (G + K + H + L + Q + P)}{\%1}\}$$

$$\%1 := L^2 Q^2 K + L^2 Q^2 G + 2 K L^3 G + 2 K^2 L^2 H + K^2 L^2 P + K L^3 P + K L^2 P^2 + Q L^3 G + Q L^3 K$$

$$+ G^2 L^2 P + G L^3 P + G L^2 P^2 + R G^2 L^2 + R K^2 H^2 + R K^2 L^2 + L^2 G^2 Q + L K^2 H^2 + L^2 K^2 Q + J G^2 L^2$$

$$+ J K^2 H^2 + J K^2 L^2 + K^2 L^3 + G^2 L^3 + 2 R Q P G L + 2 R Q P K H + 2 R Q P K L + G^2 Q J R + K Q J G R$$

$$+ 2 Q H J G R + 2 Q L J G R + Q^2 J G R + 2 Q P J G R + J G^2 R L + J G^2 R P + J G R K H + J G R K L$$

$$+ J G R K P + 2 J G R P H + 2 J G R P L + J G R P^2 + J R L^2 G + 2 J R L K H + J R L^2 K + J R L K P$$

$$+ 2 J R L P H + J R L^2 P + J R L P^2 + 2 J R L Q P + 2 L^2 Q P G + 2 L Q P K H + 2 L^2 Q P K + J R H G L$$

$$+ J R H^2 K + J R H K P + J R H^2 P + J R H P^2 + 2 J R H Q P + L Q^2 K H + 2 K L^2 G P + 2 K L^2 P H$$

$$+ 2 Q L^2 K H + 2 G L^2 K H + G L^2 P H + R Q^2 G L + R Q^2 K H + R Q^2 K L + R G^2 L P + 2 R G L K H$$

$$+ 2 R G L^2 K + 2 R G L K P + R G L P H + R G L^2 P + R G L P^2 + R G^2 Q L + R G Q K H + 2 R G Q K L$$

$$+ R K H G P + 2 R K^2 H L + R K^2 H P + R K H^2 P + 2 R K H P L + R K H P^2 + R K^2 L P + R K L^2 P$$

$$+ R K L P^2 + R K^2 Q H + R K^2 Q L + R Q H G L + R Q H^2 K + 2 R Q H K L + R Q L^2 G + R Q L^2 K$$

$$+ L G Q K H + 2 L^2 G Q K + L K H G P + L K^2 H P + L K H^2 P + L K H P^2 + L K^2 Q H + L^2 Q H G$$

$$+ L Q H^2 K + J G^2 L P + 2 J G L K H + 2 J G L^2 K + 2 J G L K P + J G L P H + J G L^2 P + J G L P^2$$

$$+ J G L Q P + J K Q G L + J K^2 Q H + J K^2 Q L + J K Q P H + J K Q P L + J K H G P + 2 J K^2 H L + J K^2 H P$$

$$+ J K H^2 P + 2 J K H P L + J K H P^2 + J K^2 L P + J K L^2 P + J K L P^2 + K Q J R H + K Q J R L + Q^2 J R H$$

$$+ Q H^2 J R + 2 Q H J R L + Q L^2 J R + Q^2 J R L$$

```
> #
> # Results of the equations are assigned to given variables.
> #
> assign(SolutionSet);
> #
> # The temporary variable is copied to new variable "divisor".
> #
> divisor := %1:
> p000_dividend := op(1,p000)*op(2,p000)*op(3,p000)*op(4,p000):
> p001_dividend := op(1,p001)*op(2,p001)*op(3,p001)*op(4,p001)*op(5,p001):
> p010_dividend := op(1,p010)*op(2,p010):
> p100_dividend := op(1,p100)*op(2,p100)*op(3,p100):
> p101_dividend := op(1,p101)*op(2,p101)*op(3,p101)*op(4,p101):
> p110_dividend := op(1,p110):
> #
> # Results of this process (dividends of p000,p001,p010,p100,p101,p110,
> # and divisor) are copied to file f_emm2_t.
> #
> save
  p000_dividend,p001_dividend,p010_dividend,
  p100_dividend,p101_dividend,p110_dividend,divisor,f_emm2_t;
> #
> #  Further processing is done in Maple V version 4.
> #
```

Analysis of EMM2A with Maple V version 4

```
> #
> # Solving EMM2 with approximation (EMM2A).
> #
> # This is Maple V version 4 part of the analysis. Here,
> # results of the solved equations are manipulated and simplified.
> #
> # Undefine constants used in equations:
> #
> G:='G': # G:=(D+1)*S*lambda[s]
> H:='H': # H:=mu[s]
> K:='K': # K:=(D+1)*lambda[d]
> J:='J': # J:=mu[d]
> L:='L': # L:=lambda[d]
> P:='P': # P:=lambda[sd]
> Q:='Q': # Q:=mu[sd]
> R:='R': # R:=mu[dr]
> #
> # Information, produced by Maple V version 2 is extracted from file F_EMM2_T.
> #
> p000_dividend := R*J*Q*(Q*L+L^2+Q*H+H^2+G*H+2*L*H+H*P+L*P):
> p001_dividend := Q*G*J*R*(K+H+G+Q+L+P):
> p010_dividend := (K*P*H^2+2*K*H*P*L+2*K*Q*P*H+K*L*P^2+K*Q^2*H+K*P^2*H+K*Q*H^2+
  K^2*Q*H+K^2*L*P+K^2*H*P+2*K^2*L*H+K^2*L^2+K^2*H^2+2*G*L*Q*P+
  G*Q*H*L+G*H*P*L+G*L^2*P+G*L*Q^2+G*L*P^2+2*G*H*K*L+2*G*K*P*L+
  G*K*H*P+G*K*Q*H+2*G*K*L^2+G^2*Q*L+G^2*L*P+G^2*L^2+2*K*Q*H*L+
  2*K*Q*G*L+2*K*Q*P*L+K^2*Q*L+K*Q*L^2+K*P*L^2+Q*L^2*G+K*Q^2*L)  *R:
> p100_dividend := R*J*(Q*P*H+P^2*H+P*H^2+L*Q*P+K*P*L+G*P*L+G*H*P+K*Q*L+H*G*L+
  K*L^2+L*P^2+L^2*G+L^2*P+2*H*K*L+2*H*P*L+Q*G*L+K*H^2+K*Q*H+K*H*P):
> p101_dividend := R*J*G*(K*P+L*P+H*P+G*P+P^2+Q*P+L*K+L*G+K*H):
> p110_dividend := P*L^2*G^2+Q^2*L^2*G+K^2*L^2*P+K*L^3*Q+L^3*G*P+L^3*G*Q+
  Q*L^2*G^2+Q*K^2*L^2+K*P*L^3+K^2*L*H^2+K*P^2*L^2+K^2*J*L^2+P^2*L^2*G+
  L^2*G^2*J+Q^2*K*L^2+2*K^2*L^2*H+K^2*L^3+2*Q*K*H*P*L+Q*K*H*G*L+Q^2*K*H*L+
  2*Q*K*L^2*G+L^3*G^2+K^2*J*H^2+2*Q*K*L^2*P+K^2*J*H*P+K^2*H*P*L+K^2*J*P*L+
  2*K^2*J*H*L+K*L*Q*H^2+2*K*L^2*Q*H+K*J*Q*H*P+K*P*L*H^2+2*K*P*L^2*H+2*K*P*L^2*G+
  2*K*L^2*G*J+2*K*L^3*G+K^2*Q*J*H+K^2*Q*J*L+K*Q*G*J*L+2*K*G*J*P*L+K*P*J*L^2+
  K*P*J*H^2+2*K*P*J*H*L+K*P^2*J*L+2*K*L^2*G*H+K*P*L*G*H+K*G*J*H*P+K*J*Q*P*L+
  Q*K^2*L*H+K*P^2*L*H+2*K*G*J*H*L+K*P^2*J*H+G*J*Q*P*L+L^2*G*Q*H+2*P*L^2*Q*G+
  G^2*J*P*L+L^2*G*H*P+L*G*J*H*P+P^2*J*G*L+L^2*G*J*P:
> divisor := P*L^2*G^2+Q^2*L^2*G+K^2*L^2*P+K*L^3*Q+L^3*G*P+L^3*G*Q+Q*L^2*G^2+
  Q*K^2*L^2+K*P*L^3+K^2*L*H^2+K*P^2*L^2+K^2*J*L^2+P^2*L^2*G+L^2*G^2*J+
  Q^2*K*L^2+2*K^2*L^2*H+K^2*L^3+2*Q*K*H*P*L+Q*K*H*G*L+Q^2*K*H*L+2*Q*K*L^2*G+
  L^3*G^2+K^2*J*H^2+2*Q*K*L^2*P+K^2*J*H*P+K^2*H*P*L+K^2*J*P*L+2*K^2*J*H*L+
  K*L*Q*H^2+2*K*L^2*Q*H+K*J*Q*H*P+K*P*L*H^2+2*K*P*L^2*H+2*K*P*L^2*G+2*K*L^2*G*J+
  2*K*L^3*G+K^2*Q*J*H+K^2*Q*J*L+K*Q*G*J*L+2*K*G*J*P*L+K*P*J*L^2+K*P*J*H^2+
  2*K*P*J*H*L+K*P^2*J*L+2*K*L^2*G*H+K*P*L*G*H+K*G*J*H*P+K*J*Q*P*L+Q*K^2*L*H+
  K*P^2*L*H+2*K*G*J*H*L+K*P^2*J*H+G*J*Q*P*L+L^2*G*Q*H+2*P*L^2*Q*G+G^2*J*P*L+
  L^2*G*H*P+L*G*J*H*P+P^2*J*G*L+L^2*G*J*P+H*G*R*Q*K+Q^2*R*K*H+K^2*R*Q*H+
  2*P*R*K*Q*H+2*L*G*R*Q*K+2*P*R*K*Q*L+Q^2*R*K*L+K^2*R*Q*L+K*Q*G*J*R+G*J*Q^2*R+
  K^2*R*L^2+R*L^2*G^2+K^2*R*H^2+Q*G^2*J*R+2*Q*P*J*G*R+2*Q*L*G*J*R+2*Q*H*G*J*R+
  2*K*R*H*G*L+H*P^2*R*K+2*P*R*Q*G*L+P^2*R*G*L+Q*R*G^2*L+Q*R*K*H^2+Q*R*H*G*L+
  Q^2*R*G*L+2*Q*R*K*L*H+P*R*K^2*H+P*R*K^2*L+P*R*K*H^2+2*K^2*R*L*H+P*R*K*G*H+
  P*R*G^2*L+P*R*H*G*L+2*P*R*K*L*H+2*P*R*K*G*L+P^2*R*K*L+Q*R*L^2*K+2*K*R*L^2*G+
  2*R*J*Q*P*L+J*H*P^2*R+J*P*R*H^2+P*R*L^2*G+P*R*L^2*K+Q*R*L^2*G+2*J*P*R*L*H+
  G*J*R*L*H+2*G*J*P*R*L+2*G*J*P*R*H+Q^2*J*L*R+K*R*J*Q*H+K*J*R*L^2+K*R*G*J*L+
  K*G*J*R*P+2*K*J*R*L*H+K*J*P*R*L+K*J*P*R*H+K*J*R*H^2+2*J*H*L*Q*R+J*H^2*Q*R+
  J*L^2*Q*R+Q^2*J*H*R+K*R*J*G*H+2*R*J*Q*H*P+P*J*R*L^2+P^2*J*R*L+L^2*G*J*R+
  K*R*J*Q*L+G^2*J*R*L+G^2*J*R*P+P^2*J*G*R:
> #
> # Now check the consistency (p000_dividend+p001_dividend+p010_dividend+
> #  p100_dividend+p101_dividend+p110_dividend) should be equal to divisor.
> #
> simplify(
> p000_dividend+p001_dividend+p010_dividend+
> p100_dividend+p101_dividend+p110_dividend -divisor);
                                     0
> #
> # Consistency checked and OK
> #
> # Simplify the terms (p000_dividend ... p110_dividend). This must be done
> # manually, as Maple V doesn't do it automatically. Call the simplified
> # dividends as q000_dividend ... q110_dividend. Also, check that results are
> # still correct by comparing the p???_dividend with q???_dividend.
> #
> q000_dividend := R*J*Q*((P+H+Q+L)*(L+H)+G*H):
> simplify(p000_dividend-q000_dividend);
                                     0
> q001_dividend := R*J*G*Q*(L+P+K+G+Q+H):
> simplify(p001_dividend-q001_dividend);
                                     0
> q010_dividend := R*(G*L+K*L+K*H)*((L+P+Q)*(G+P+K+Q)+H*(K+P+Q)):
> simplify(p010_dividend-q010_dividend);
                                     0
> q100_dividend := R*J*((L+H)*((K+P)*(Q+H+L+P)+(P+L)*G)+G*L*Q):
> simplify(p100_dividend-q100_dividend);
                                     0
> q101_dividend := R*J*G*((P+K)*(L+P+H)+P*(G+Q)+G*L):
> simplify(p101_dividend-q101_dividend);
                                     0
> q110_dividend :=
  ((K*H+K*L+G*L)*(P*J))*((H+G+Q+P))+
  L*L*(G+K)*(P+Q)*(P+H+Q+K+L+G)+
  (K*L)*((Q+P+K+H)*(H*Q+P*H))+
  (K*L)*((Q+P)*(G*H+H*L+J*G))+
  L*L*G*(J*(P+G)+(G+K)*L)+
  (K*H+K*L)*(J*K*(Q+H+P)+L*L*(G+K))+
  (K*L)*(
  (J+L)*(H*G)+
  (K*H)*(J+H+L)+
  (J*L)*(G+K)+
  (J*G)*(H+L)+
  J*P*(H+L)):
> simplify(p110_dividend-q110_dividend);
                                     0
> #
> # Also simplify the divisor with sdivisor
> #
> sdivisor :=
  R*J*Q*(L*L+L*P+2*L*H+L*Q+P*H+G*H+H*Q+H*H)+
  R*J*G*Q*(L+P+K+G+Q+H)+
  R*((L*G+K*L+K*H)*((L+P+Q)*(G+P+K+Q)+H*(K+P+Q)))+
  R*J*((L+H)*((K+P)*(Q+H+L+P)+(P+L)*G)+G*L*Q)+
  R*J*G*((P+K)*(L+P+H)+P*(G+Q)+G*L)+
  ((K*H+K*L+G*L)*(P*J))*((H+G+Q+P))+
  L*L*(G+K)*(P+Q)*(P+H+Q+K+L+G)+
  (K*L)*((Q+P+K+H)*(H*Q+P*H))+
  (K*L)*((Q+P)*(G*H+H*L+J*G))+
  L*L*G*(J*(P+G)+(G+K)*L)+
  (K*H+K*L)*(J*K*(Q+H+P)+L*L*(G+K))+
  (K*L)*((J+L)*(H*G)+
  (K*H)*(J+H+L)+
  (J*L)*(G+K)+
  (J*G)*(H+L)+
  J*P*(H+L)):
> simplify(sdivisor-divisor);
```

$$0$$

```
> #
> # Now, it is possible to change the parameters to Greek alphabets.
> #
> G:=(D+1)*S*lambda[s]:
> H:=mu[s]:
> K:=(D+1)*lambda[d]:
> J:=mu[d]:
> L:=lambda[d]:
> P:=lambda[sd]:
> Q:=mu[sd]:
> R:=mu[dr]:
> #
> # p000
> #
> q000_dividend/QII;
```

$$\frac{\mu_{dr}\,\mu_d\,\mu_{sd}\,((\lambda_{sd}+\mu_s+\mu_{sd}+\lambda_d)\,(\lambda_d+\mu_s)+(D+1)\,S\,\lambda_s\,\mu_s)}{QII}$$

```
> #
> # p001
> #
> q001_dividend/QII;
>
```

$$\frac{\mu_{sd}\,(D+1)\,S\,\lambda_s\,\mu_d\,\mu_{dr}\,((D+1)\,\lambda_d+\mu_s+(D+1)\,S\,\lambda_s+\mu_{sd}+\lambda_d+\lambda_{sd})}{QII}$$

```
> #
> # p010
> #
> q010_dividend/QII;
```

$$\mu_{dr}\,(\lambda_d\,(D+1)\,S\,\lambda_s+\lambda_d^2\,(D+1)+(D+1)\,\lambda_d\,\mu_s)\,($$
$$(\lambda_d+\lambda_{sd}+\mu_{sd})\,((D+1)\,S\,\lambda_s+\lambda_{sd}+(D+1)\,\lambda_d+\mu_{sd})+\mu_s\,((D+1)\,\lambda_d+\lambda_{sd}+\mu_{sd})$$
$$)\,/\,QII$$

```
> #
> # p100
> #
> q100_dividend/QII;
```

$$\mu_{dr}\,\mu_d\,((\lambda_d+\mu_s)\,(((D+1)\,\lambda_d+\lambda_{sd})\,(\lambda_{sd}+\mu_s+\mu_{sd}+\lambda_d)+(\lambda_{sd}+\lambda_d)\,(D+1)\,S\,\lambda_s)$$
$$+\mu_{sd}\,(D+1)\,S\,\lambda_s\,\lambda_d)\,/\,QII$$

```
> #
> # p101
> #
> q101_dividend/QII;
```

$$\mu_{dr}\,\mu_d\,(D+1)\,S\,\lambda_s$$

$$(((D+1)\,\lambda_d+\lambda_{sd})\,(\lambda_d+\lambda_{sd}+\mu_s)+\lambda_{sd}\,((D+1)\,S\,\lambda_s+\mu_{sd})+\lambda_d\,(D+1)\,S\,\lambda_s)\,/\,QII$$

```
> #
> # p110
> #
> q110_dividend/QII;
```

$$($$

$$(\lambda_d\,(D+1)\,S\,\lambda_s+\lambda_d^2\,(D+1)+(D+1)\,\lambda_d\,\mu_s)\,\lambda_{sd}\,\mu_d\,(\mu_s+(D+1)\,S\,\lambda_s+\mu_{sd}+\lambda_{sd})$$

$$+\lambda_d^2\,\%1\,(\lambda_{sd}+\mu_{sd})\,((D+1)\,\lambda_d+\mu_s+(D+1)\,S\,\lambda_s+\mu_{sd}+\lambda_d+\lambda_{sd})$$

$$+(D+1)\,\lambda_d^2\,(\mu_{sd}+\lambda_{sd}+(D+1)\,\lambda_d+\mu_s)\,(\mu_{sd}\,\mu_s+\mu_s\,\lambda_{sd})$$

$$+(D+1)\,\lambda_d^2\,(\lambda_{sd}+\mu_{sd})\,((D+1)\,S\,\lambda_s\,\mu_s+\lambda_d\,\mu_s+\mu_d\,(D+1)\,S\,\lambda_s)$$

$$+\lambda_d^2\,(D+1)\,S\,\lambda_s\,(\mu_d\,(\lambda_{sd}+(D+1)\,S\,\lambda_s)+\%1\,\lambda_d)$$

$$+((D+1)\,\lambda_d\,\mu_s+\lambda_d^2\,(D+1))\,(\mu_d\,(D+1)\,\lambda_d\,(\mu_{sd}+\mu_s+\lambda_{sd})+\lambda_d^2\,\%1)+(D+1)$$

$$\lambda_d^2\,((\mu_d+\lambda_d)\,(D+1)\,S\,\lambda_s\,\mu_s+(D+1)\,\lambda_d\,\mu_s\,(\mu_d+\mu_s+\lambda_d)+\mu_d\,\lambda_d\,\%1$$

$$+\mu_d\,(D+1)\,S\,\lambda_s\,(\lambda_d+\mu_s)+\mu_d\,\lambda_{sd}\,(\lambda_d+\mu_s)))\,/\,QII$$

$$\%1 := (D+1)\,S\,\lambda_s+(D+1)\,\lambda_d$$

```
> #
> # Where QII is the divisor and equals
> #
> sdivisor;
```

$$\mu_{dr}\,\mu_{d}\,\mu_{sd}\,(\mu_{sd}\,\lambda_{d}+\lambda_{d}^{2}+\mu_{sd}\,\mu_{s}+\mu_{s}^{2}+(D+1)\,S\,\lambda_{s}\,\mu_{s}+2\,\lambda_{d}\,\mu_{s}+\mu_{s}\,\lambda_{sd}+\lambda_{d}\,\lambda_{sd})$$

$$+\mu_{sd}\,(D+1)\,S\,\lambda_{s}\,\mu_{d}\,\mu_{dr}\,((D+1)\,\lambda_{d}+\mu_{s}+(D+1)\,S\,\lambda_{s}+\mu_{sd}+\lambda_{d}+\lambda_{sd})+\mu_{dr}$$

$$(\lambda_{d}\,(D+1)\,S\,\lambda_{s}+\lambda_{d}^{2}\,(D+1)+(D+1)\,\lambda_{d}\,\mu_{s})\,($$

$$(\lambda_{d}+\lambda_{sd}+\mu_{sd})\,((D+1)\,S\,\lambda_{s}+\lambda_{sd}+(D+1)\,\lambda_{d}+\mu_{sd})+\mu_{s}\,((D+1)\,\lambda_{d}+\lambda_{sd}+\mu_{sd})$$

$$)+\mu_{dr}\,\mu_{d}\,($$

$$(\lambda_{d}+\mu_{s})\,(((D+1)\,\lambda_{d}+\lambda_{sd})\,(\lambda_{sd}+\mu_{s}+\mu_{sd}+\lambda_{d})+(\lambda_{sd}+\lambda_{d})\,(D+1)\,S\,\lambda_{s})$$

$$+\mu_{sd}\,(D+1)\,S\,\lambda_{s}\,\lambda_{d})+\mu_{dr}\,\mu_{d}\,(D+1)\,S\,\lambda_{s}$$

$$(((D+1)\,\lambda_{d}+\lambda_{sd})\,(\lambda_{d}+\lambda_{sd}+\mu_{s})+\lambda_{sd}\,((D+1)\,S\,\lambda_{s}+\mu_{sd})+\lambda_{d}\,(D+1)\,S\,\lambda_{s})+$$

$$(\lambda_{d}\,(D+1)\,S\,\lambda_{s}+\lambda_{d}^{2}\,(D+1)+(D+1)\,\lambda_{d}\,\mu_{s})\,\lambda_{sd}\,\mu_{d}\,(\mu_{s}+(D+1)\,S\,\lambda_{s}+\mu_{sd}+\lambda_{sd})$$

$$+\lambda_{d}^{2}\,\%1\,(\lambda_{sd}+\mu_{sd})\,((D+1)\,\lambda_{d}+\mu_{s}+(D+1)\,S\,\lambda_{s}+\mu_{sd}+\lambda_{d}+\lambda_{sd})$$

$$+(D+1)\,\lambda_{d}^{2}\,(\mu_{sd}+\lambda_{sd}+(D+1)\,\lambda_{d}+\mu_{s})\,(\mu_{sd}\,\mu_{s}+\mu_{s}\,\lambda_{sd})$$

$$+(D+1)\,\lambda_{d}^{2}\,(\lambda_{sd}+\mu_{sd})\,((D+1)\,S\,\lambda_{s}\,\mu_{s}+\lambda_{d}\,\mu_{s}+\mu_{d}\,(D+1)\,S\,\lambda_{s})$$

$$+\lambda_{d}^{2}\,(D+1)\,S\,\lambda_{s}\,(\mu_{d}\,(\lambda_{sd}+(D+1)\,S\,\lambda_{s})+\%1\,\lambda_{d})$$

$$+((D+1)\,\lambda_{d}\,\mu_{s}+\lambda_{d}^{2}\,(D+1))\,(\mu_{d}\,(D+1)\,\lambda_{d}\,(\mu_{sd}+\mu_{s}+\lambda_{sd})+\lambda_{d}^{2}\,\%1)+(D+1)$$

$$\lambda_{d}^{2}\,((\mu_{d}+\lambda_{d})\,(D+1)\,S\,\lambda_{s}\,\mu_{s}+(D+1)\,\lambda_{d}\,\mu_{s}\,(\mu_{d}+\mu_{s}+\lambda_{d})+\mu_{d}\,\lambda_{d}\,\%1$$

$$+\mu_{d}\,(D+1)\,S\,\lambda_{s}\,(\lambda_{d}+\mu_{s})+\mu_{d}\,\lambda_{sd}\,(\lambda_{d}+\mu_{s}))$$

$$\%1 := (D+1)\,S\,\lambda_{s}+(D+1)\,\lambda_{d}$$

```
> #
> # Final check (dividends divided by divisor should equal one)
> #
> simplify(
> (q000_dividend+q001_dividend+q010_dividend+
>  q100_dividend+q101_dividend+q110_dividend)/sdivisor);
```

$$1$$

```
> #
> # End of EMM2A analysis.
> #
```

Appendix B: Comparison with results in technical literature

In Appendix B, the reliability models of EMM1 and EMM2A are compared with TMM that is the reliability model presented in technical literature [Schwarz 1994, Hillo 1993a, Gibson 1991]. The default and the extreme values are listed in the following table.

Table B-1. Default and extreme parameters for comparison of TMM, EMM1, and EMM2A

Parameter	Value range	Default value in TMM	Default value in EMM1	Default value in EMM2A
D	1-100	50	50	50
S	1 000 000	1 000 000	1 000 000	1 000 000
$1/\lambda_d$	10 000h - 100 000 000h	200 000h	200 000h	200 000h
$1/\lambda_{df}$	100h - 10 000 000h	200 000h	200 000h	200 000h
$1/\lambda_s$	1 000h * S - 10 000 000h * S	-	200 000h * S	200 000h * S
$1/\lambda_{sd}$	10 000h - 10 000 000h	-	2 000 000h	2 000 000h
$1/\mu_d$	1h - 1 000h	24h	24h	24h
$1/\mu_s$	1h - 1 000h	-	24h	24h
$1/\mu_{sd}$	0/24h	-	-	0/24h
$1/\mu_{dr}$	0/24h	-	-	0/24h

Comparison as a function of disk unit reliability

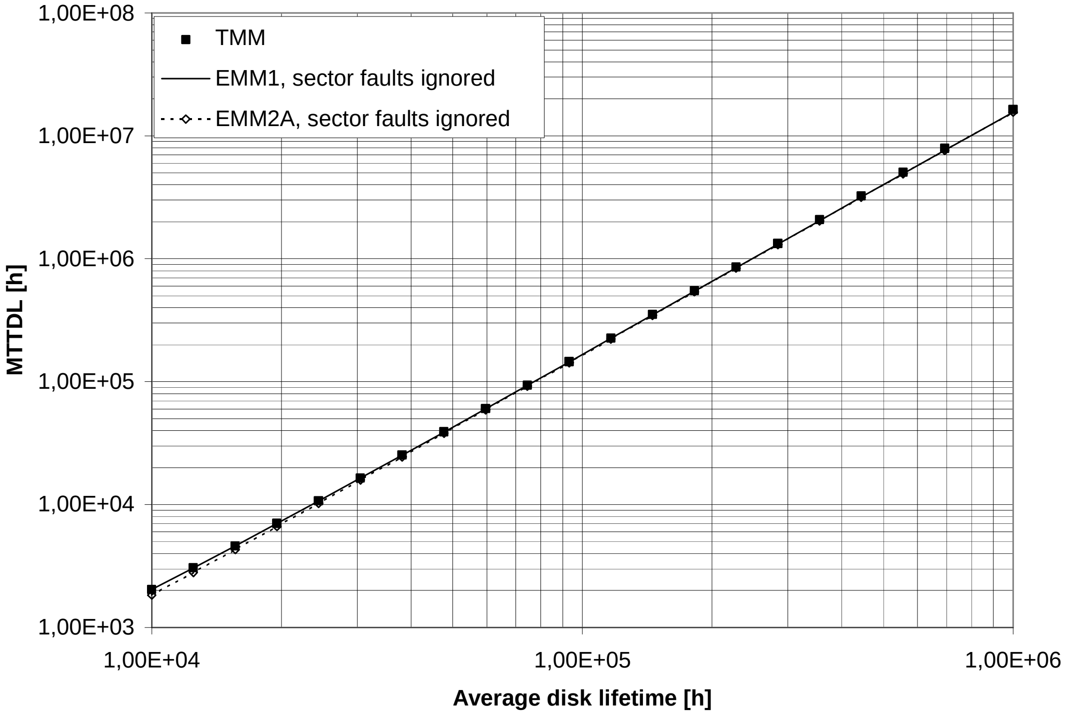

Figure B-1: MTTDL as a function of disk unit reliability

D	=	50
S	=	1 000 000
$1/\lambda_d$	=	10 000 - 1 000 000h
$1/\lambda_{df}$	=	10 000 - 1 000 000h
$1/\lambda_s$	=	ignored
$1/\lambda_{sd}$	=	10 000 - 1 000 000h
$1/\mu_d$	=	24h
$1/\mu_s$	=	ignored
$1/\mu_{sd}$	=	0h
$1/\mu_{dr}$	=	0h

Comparison as function of disk repair rate

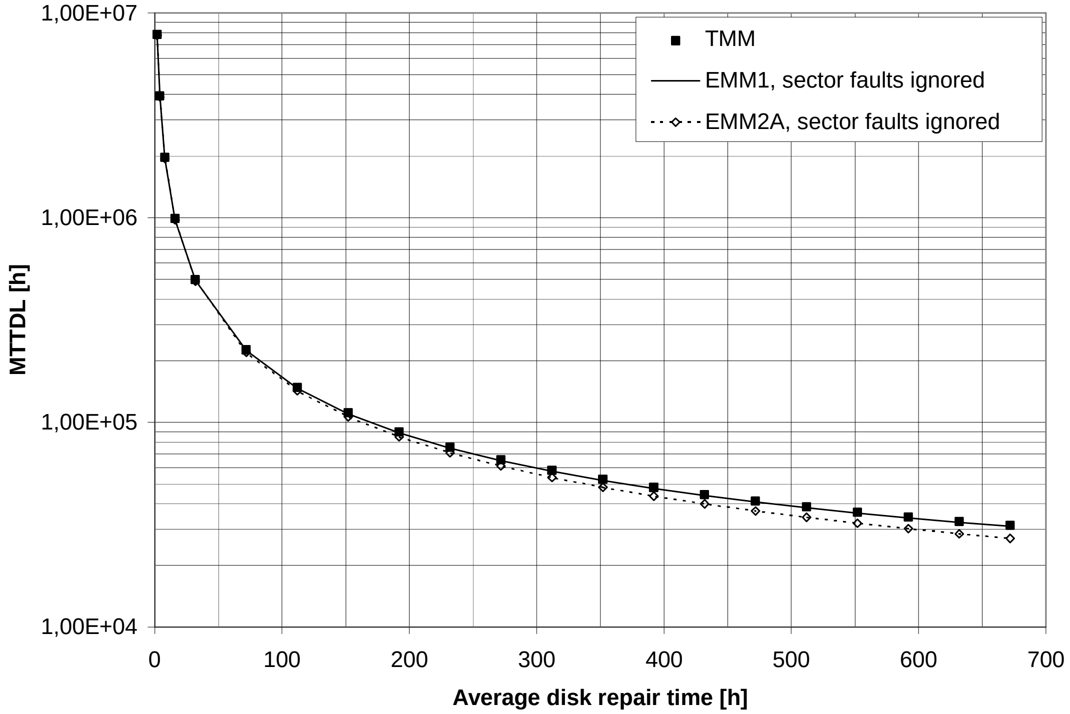

Figure B-2: MTTDL as a function of disk repair time

D	=	50
S	=	1 000 000
$1/\lambda_d$	=	200 000h
$1/\lambda_{df}$	=	200 000h
$1/\lambda_s$	=	ignored
$1/\lambda_{sd}$	=	200 000h
$1/\mu_d$	=	1 - 700h
$1/\mu_s$	=	ignored
$1/\mu_{sd}$	=	0h
$1/\mu_{dr}$	=	0h

Comparison as function of the number of disks in the array

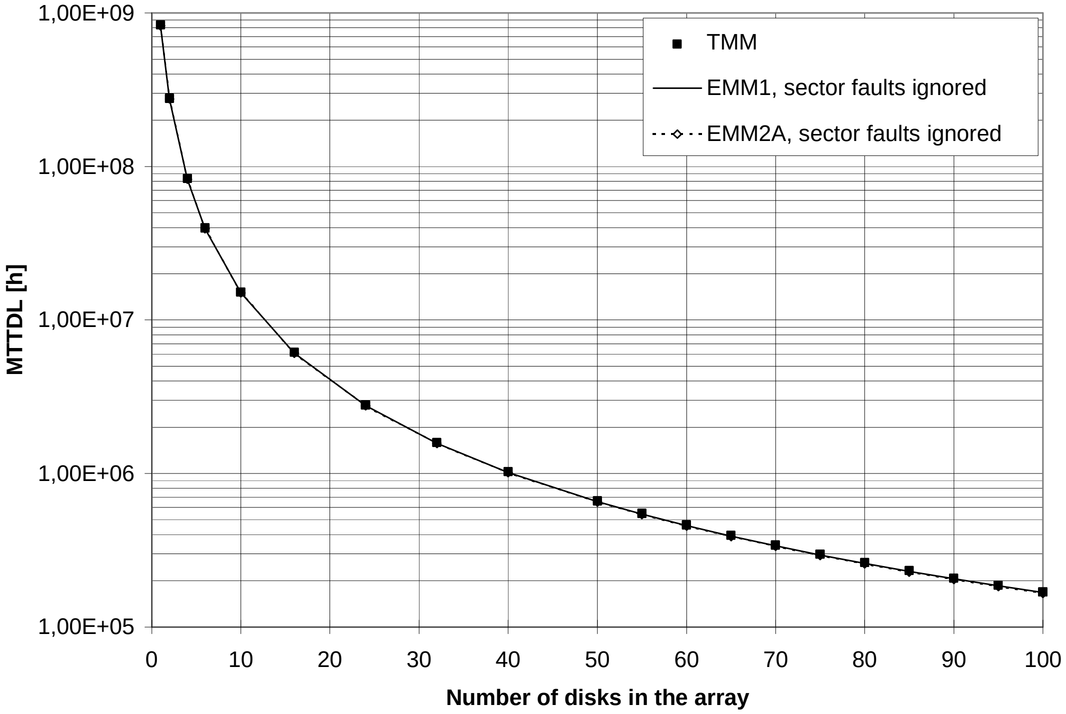

Figure B-3: MTTDL as a function of the number of disks in the array

D	=	1 - 100
S	=	1 000 000
$1/\lambda_d$	=	200 000h
$1/\lambda_{df}$	=	200 000h
$1/\lambda_s$	=	ignored
$1/\lambda_{sd}$	=	200 000h
$1/\mu_d$	=	24h
$1/\mu_s$	=	ignored
$1/\mu_{sd}$	=	0h
$1/\mu_{dr}$	=	0h

Comparison as function of disk unit reliability

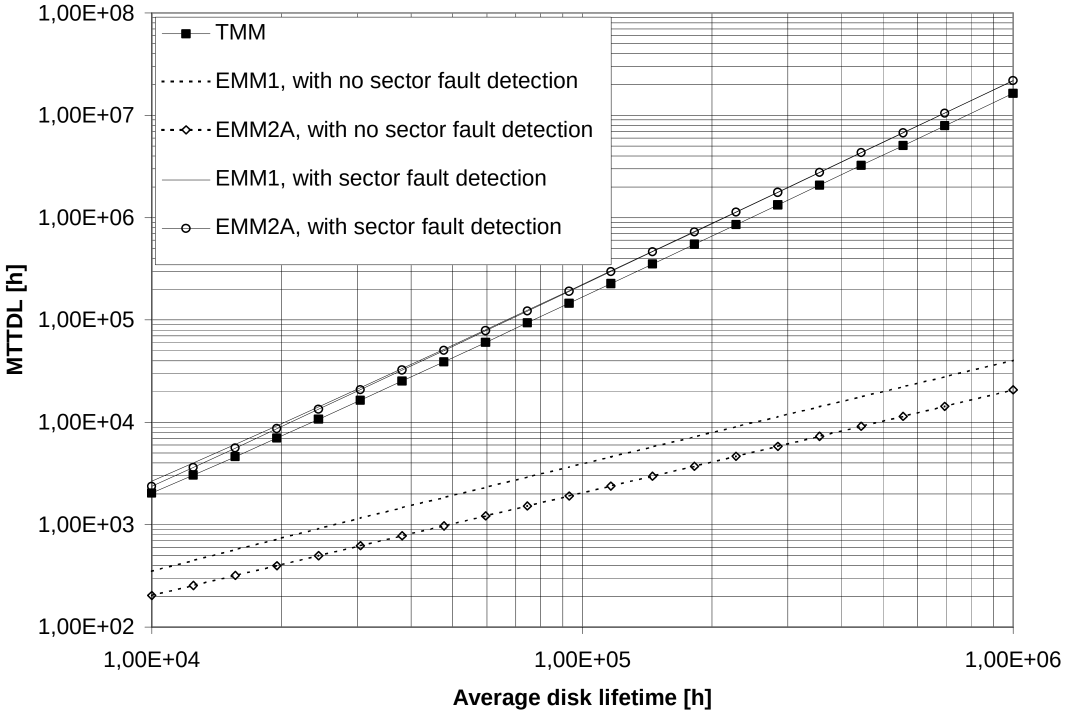

Figure B-4: MTTDL as a function of disk unit reliability

D	=	50
S	=	1 000 000
$1/\lambda_d$	=	10 000 - 1 000 000h (for TMM)
$1/\lambda_d$	=	20 000 - 2 000 000h (for EMM1 and EMM2A)
$1/\lambda_{df}$	=	10 000 - 1 000 000h (for TMM)
$1/\lambda_{df}$	=	20 000 - 2 000 000h (for EMM1 and EMM2A)
$1/\lambda_s$	=	20 000 - 2 000 000h
$1/\lambda_{sd}$	=	10 000 - 1 000 000h (for TMM)
$1/\lambda_{sd}$	=	20 000 - 2 000 000h (for EMM2A)
$1/\mu_d$	=	24h
$1/\mu_s$	=	24h
$1/\mu_{sd}$	=	24h
$1/\mu_{dr}$	=	24h

Comparison as function of disk repair rate

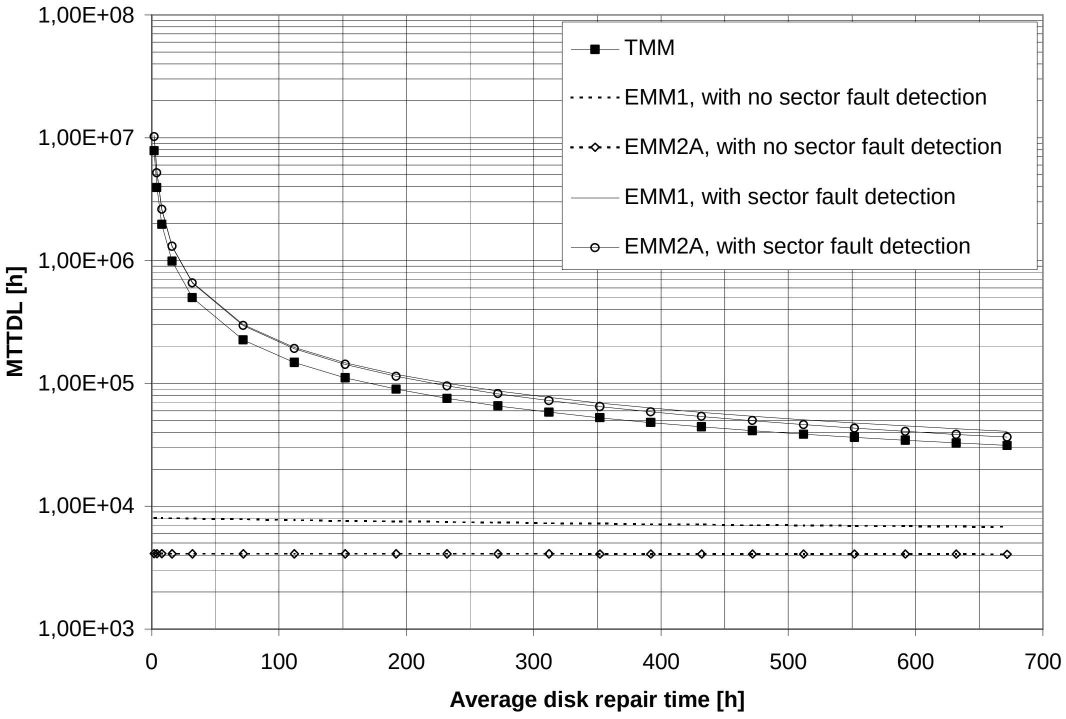

Figure B-5: MTTDL as a function of disk repair time

D	=	50	
S	=	1 000 000	
$1/\lambda_d$	=	200 000h (for TMM)	400 000h (for EMM1 and EMM2A)
$1/\lambda_{df}$	=	200 000h (for TMM)	400 000h (for EMM1 and EMM2A)
$1/\lambda_s$	=		400 000h (for EMM1 and EMM2A)
$1/\lambda_{sd}$	=		400 000h (for EMM2A)
$1/\mu_d$	=	1 - 700h	
$1/\mu_s$	=	1 - 700h	
$1/\mu_{sd}$	=	24h	
$1/\mu_{dr}$	=	24h	

Comparison as function of the number of disks in the array

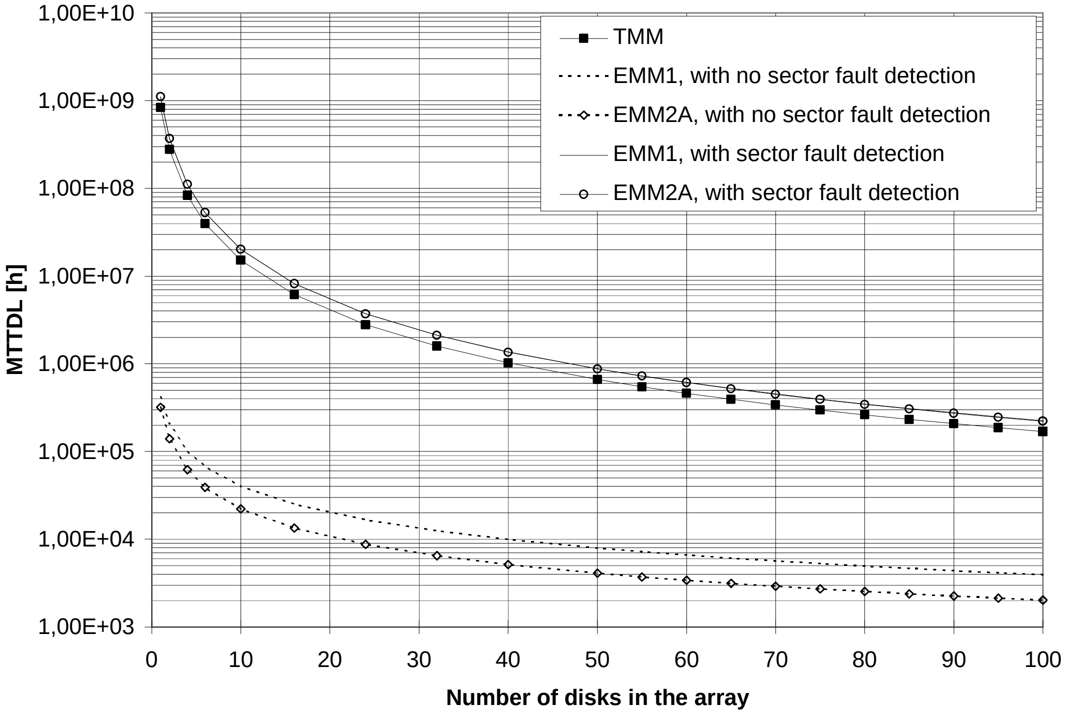

Figure B-6: MTTDL as a function of the number of disks in the array

D	=	1-100	
S	=	1 000 000	
$1/\lambda_d$	=	200 000h (for TMM)	400 000h (for EMM1 and EMM2A)
$1/\lambda_{df}$	=	200 000h (for TMM)	400 000h (for EMM1 and EMM2A)
$1/\lambda_s$	=		400 000h (for EMM1 and EMM2A)
$1/\lambda_{sd}$	=		400 000h (for EMM2A)
$1/\mu_d$	=	24h	
$1/\mu_s$	=	24h	
$1/\mu_{sd}$	=	24h	
$1/\mu_{dr}$	=	24h	

Appendix C: Sensitivity analysis of the parameters

In Appendix C, the reliability models EMM1 and EMM2A are illustrated in respect of sensitivity of various parameters in the following pages. The default and the extremes values are listed in the following table.

Table C-1. Default and extreme parameters for sensitivity analysis

Parameter	Value range[*]	Default value in EMM1	Default value in EMM2A
D	1-100	50	50
S	1 000 000	1 000 000	1 000 000
$1/\lambda_d$	10 000h - 100 000 000h	200 000h	200 000h
$1/\lambda_{df}$	100h - 10 000 000h	200 000h	200 000h
$1/\lambda_s$	1 000h * S - 10 000 000h * S	200 000h * S	200 000h * S
$1/\lambda_{sd}$	10 000h - 10 000 000h	2 000 000h	2 000 000h
$1/\mu_d$	1h - 1 000 000h	24h	8h
$1/\mu_s$	1h - 1 000 000h	24h	24h
$1/\mu_{sd}$	32h - 1 000 000h	-	44h
$1/\mu_{dr}$	1h - 1 000 000h	-	32h

[*] The upper values for some of the parameters are out of the normal range of these parameters. However, such high upper values are used in analyzing the sensitivity of the equations.

Number of disks (D) with respect to disk unit failure rate

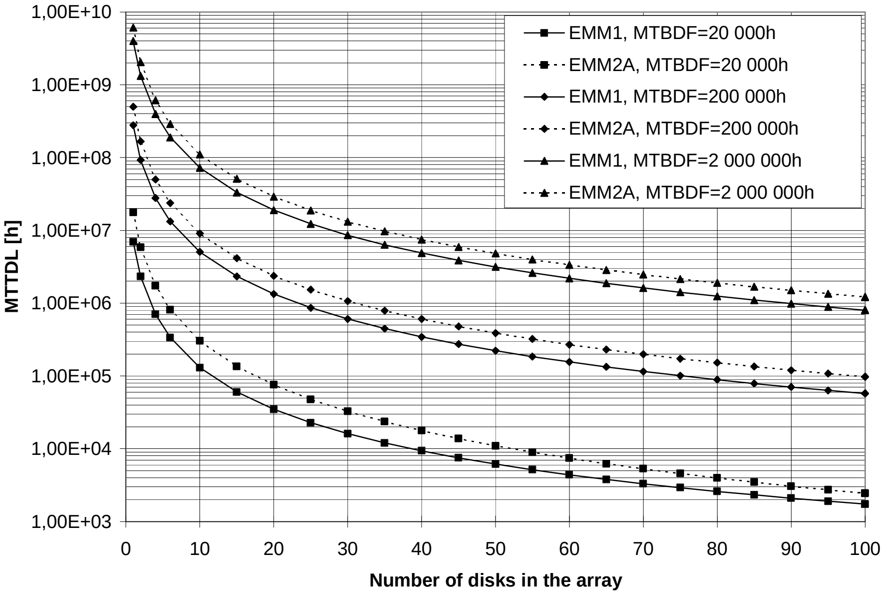

Figure C-1: MTTDL as a function of the number of disks and disk unit reliability

Number of disks (D) with respect to sector failure rate

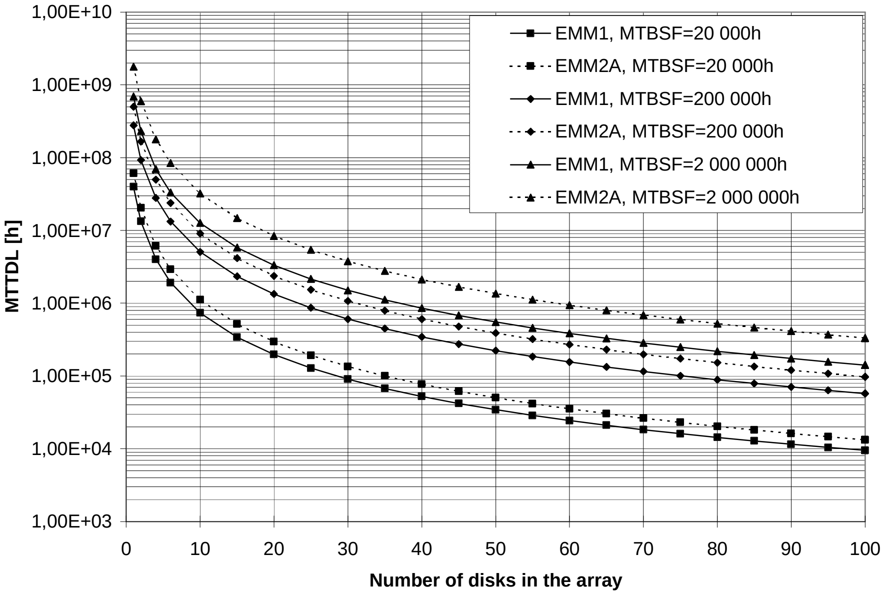

Figure C-2: MTTDL as a function of the number of disks and sector failure rate

Number of disks (D) with respect to disk unit repair rate

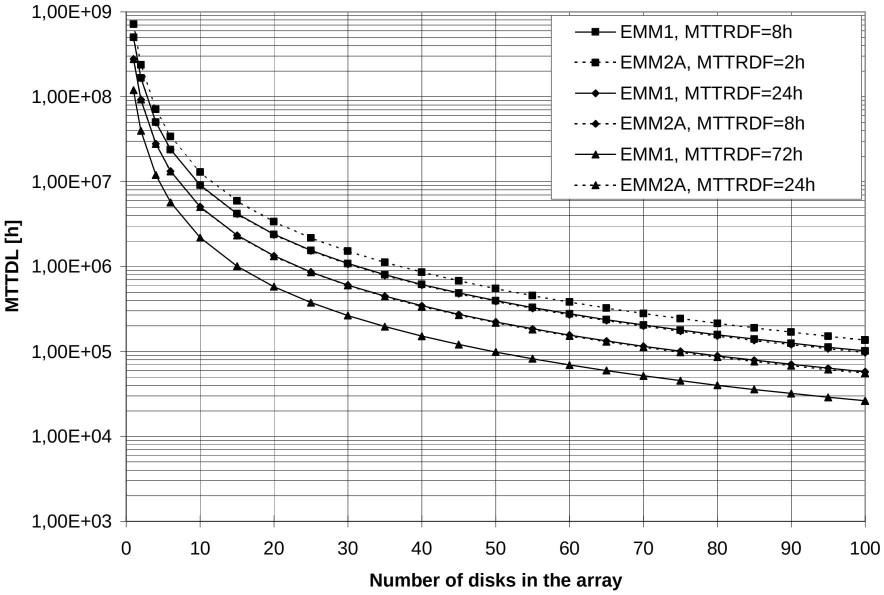

Figure C-3: MTTDL as a function of the number of disks and disk repair rate

Number of disks (D) with respect to sector repair rate

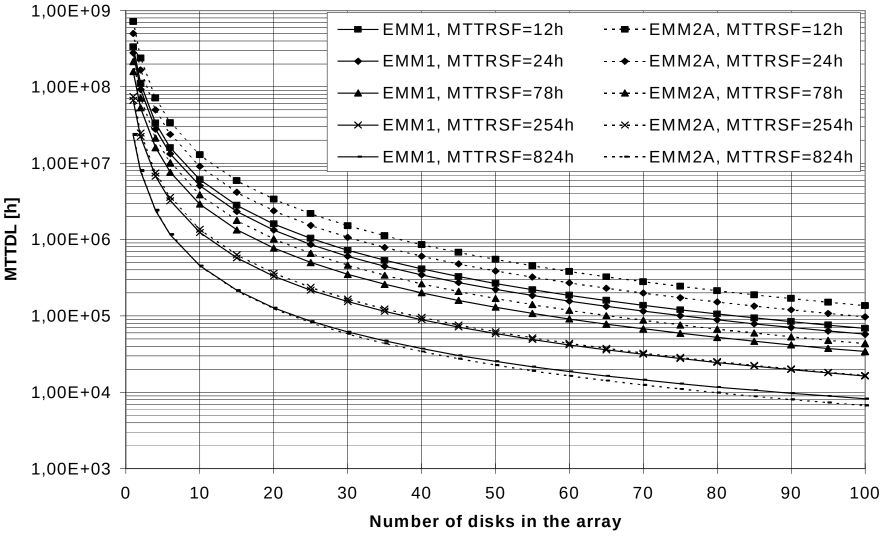

Figure C-4: MTTDL as a function of the number of disks and sector repair rate

Number of disks (D) with respect to disk repair rate

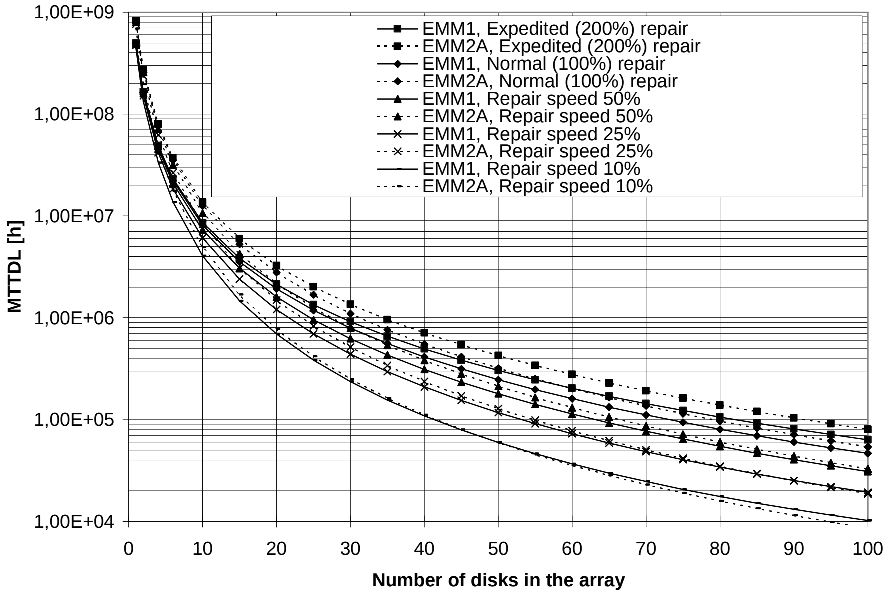

Figure C-5: MTTDL as a function of the number of disks and relative repair time

Disk unit failure rate (λ_d)

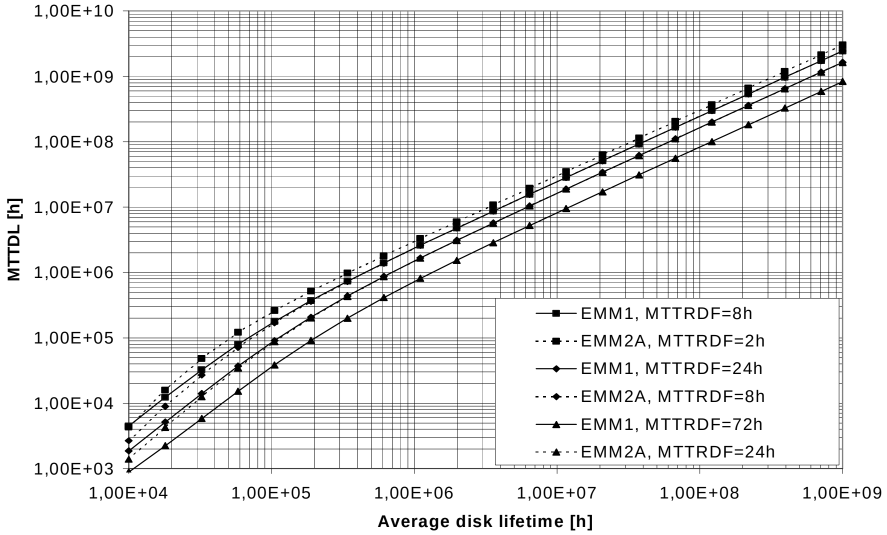

Figure C-6: MTTDL as a function of disk unit reliability

Sector failure rate (λ_s)

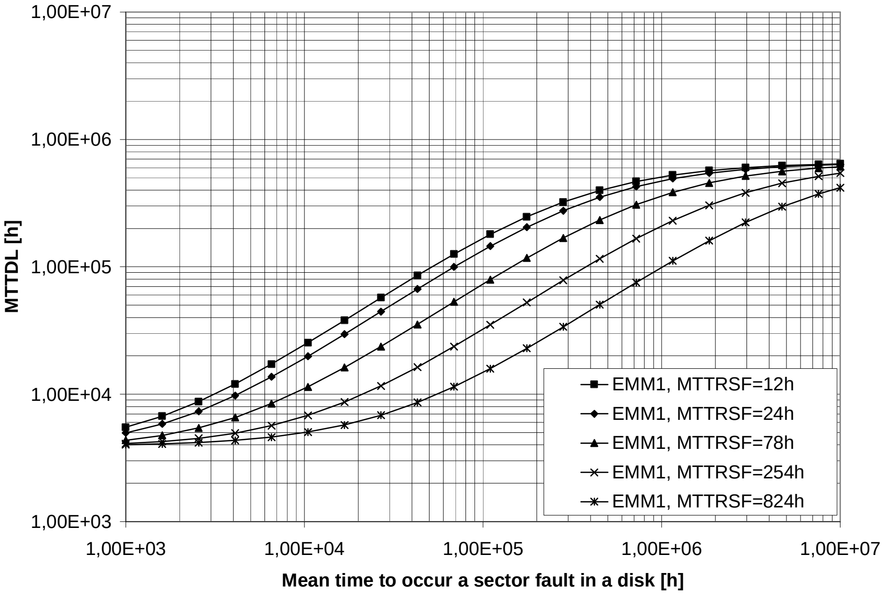

Figure C-7: MTTDL for EMM1 as a function of sector failure rate

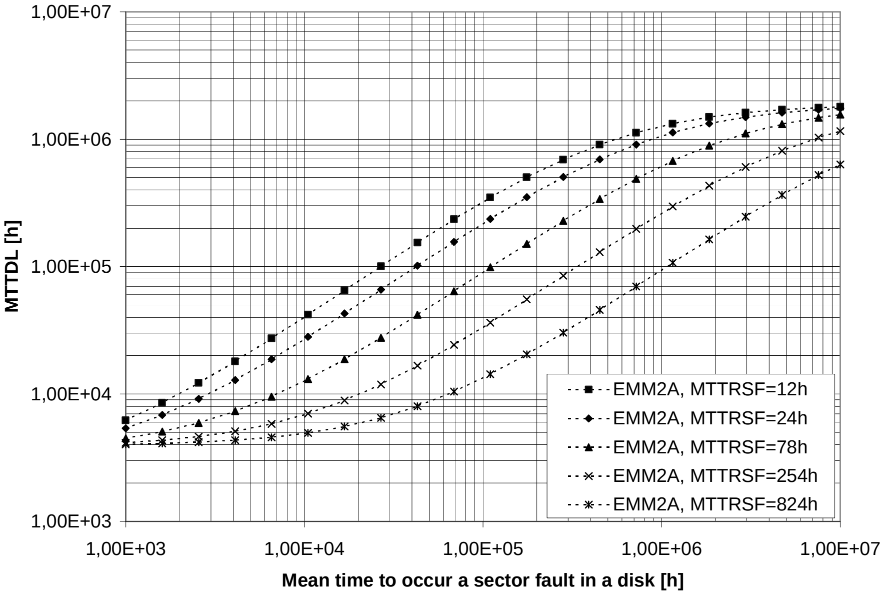

Figure C-8: MTTDL for EMM2A as a function of sector failure rate

Second disk unit failure rate (λ_{df})

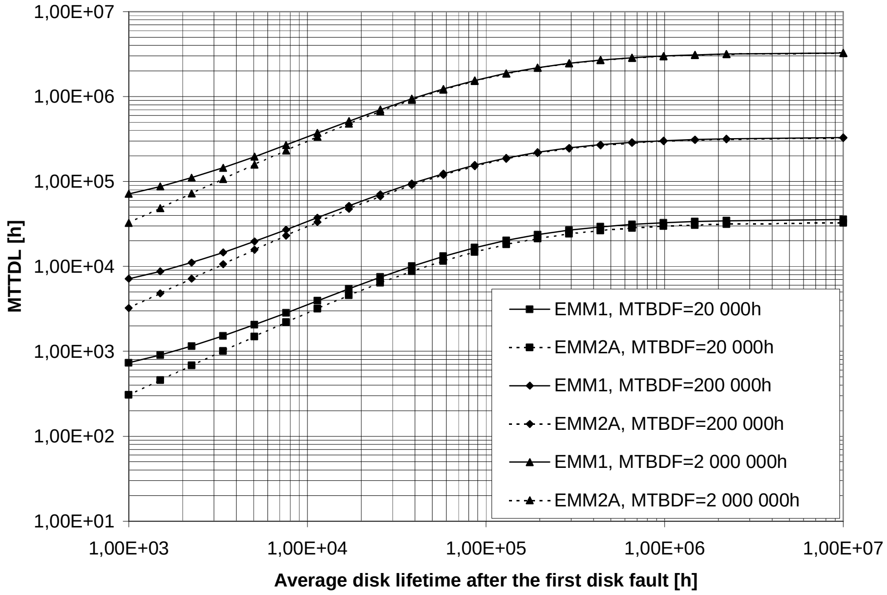

Figure C-9: MTTDL as a function of second disk failure rate

Spare disk unit failure rate (λ_{sd})

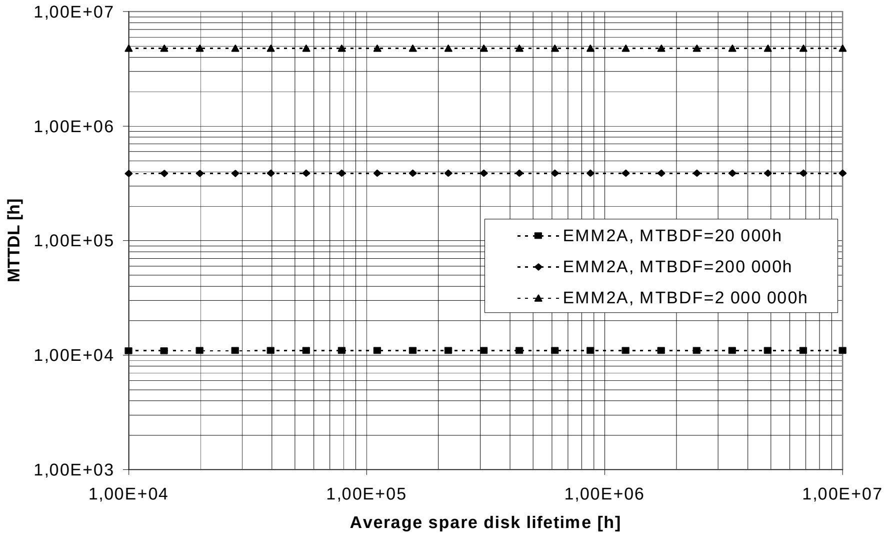

Figure C-10: MTTDL as a function of spare disk reliability

Disk unit repair rate (μ_d)

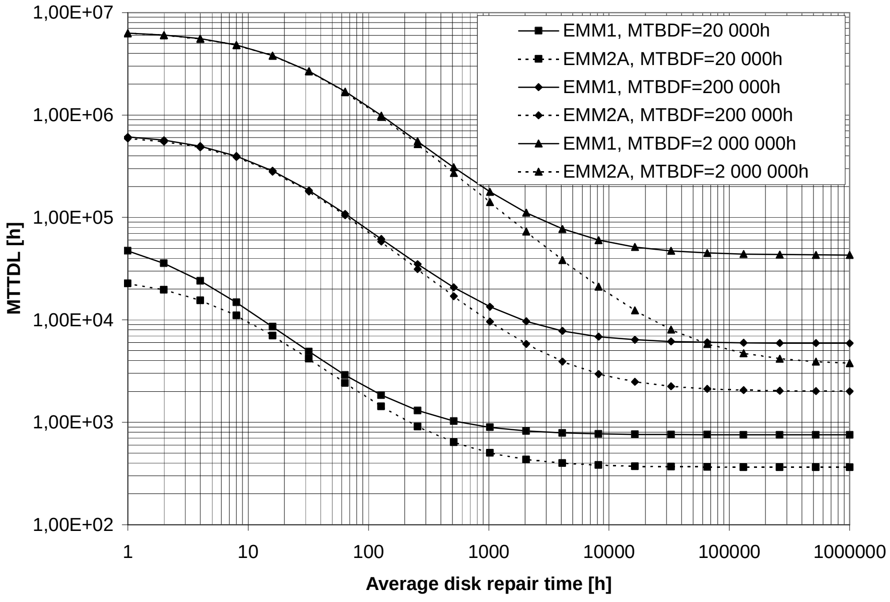

Figure C-11: MTTDL as a function of disk repair rate

Sector fault detection and repair rate (μ_s)

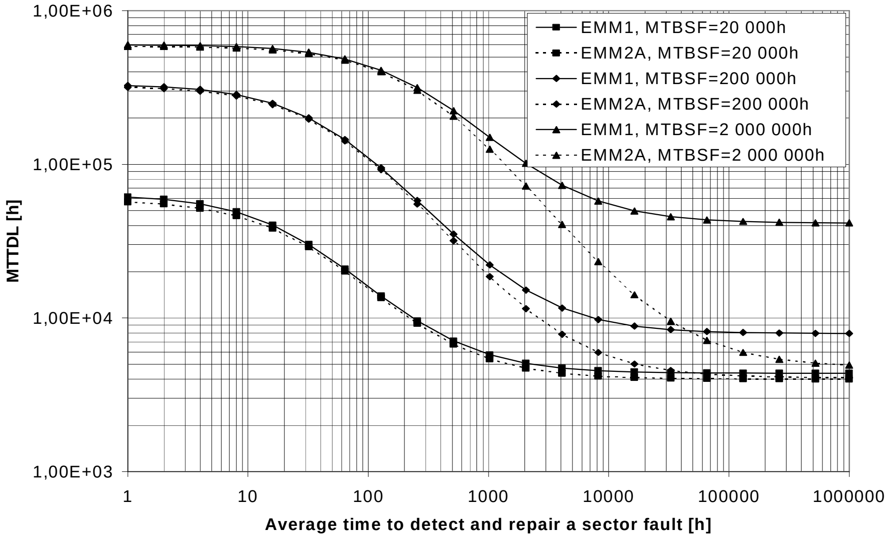

Figure C-12: MTTDL as a function of sector repair rate

Spare disk unit repair rate (μ_{sd})

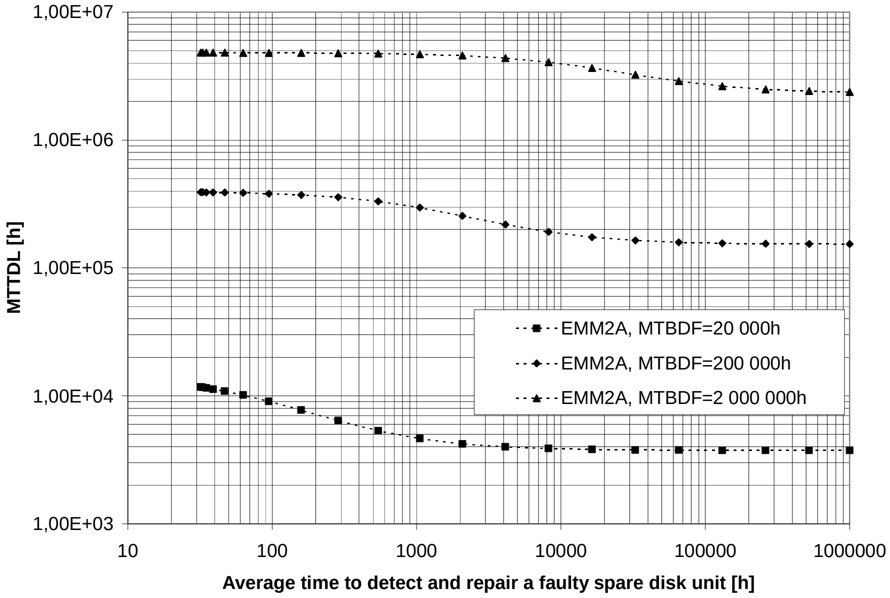

Figure C-13: MTTDL as a function of spare disk fault detection and repair rate

Spare disk unit repair rate (μ_{dr})

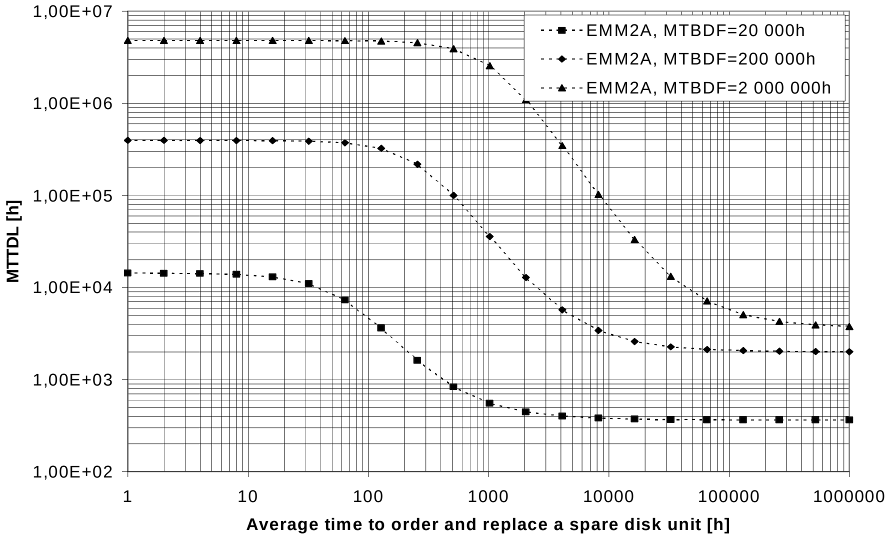

Figure C-14: MTTDL as a function of spare disk order and repair rate

Appendix D: Analysis for the approximation accuracy

In Appendix D, the exact reliability model EMM1 and two approximation models, EMM1A and EMM2A, are compared. The figures in the following pages illustrate the reliability of the disk array as a function of each of the corresponding parameters. The default and the extremes values are listed in the following table.

Table D-1. Default and extreme parameters for approximation accuracy

Parameter	Value range[*]	Default value in EMM1, EMM1A, and EMM2A[†]
D	1-100	50
S	1 000 000	1 000 000
$1/\lambda_d$	10 000h - 100 000 000h	200 000h
$1/\lambda_{df}$	100h - 10 000 000h	200 000h
$1/\lambda_s$	1 000h * S - 10 000 000h * S	200 000h * S
$1/\lambda_{sd}$	10 000h - 10 000 000h	2 000 000h
$1/\mu_d$	1h - 1 000 000h	24h
$1/\mu_s$	1h - 1 000 000h	24h
$1/\mu_{sd}$	0h	0h
$1/\mu_{dr}$	0h	0h

[*] The upper values for some of the parameters are out of the normal range of these parameters. However, such high upper values are used in analyzing the approximation models.

[†] All three models are using exactly the same parameters and the spare disk repair is assumed to be immediate. Hence, all three models should provide same results. Deviation in the results is a sign of approximation error.

Number of disks (D)

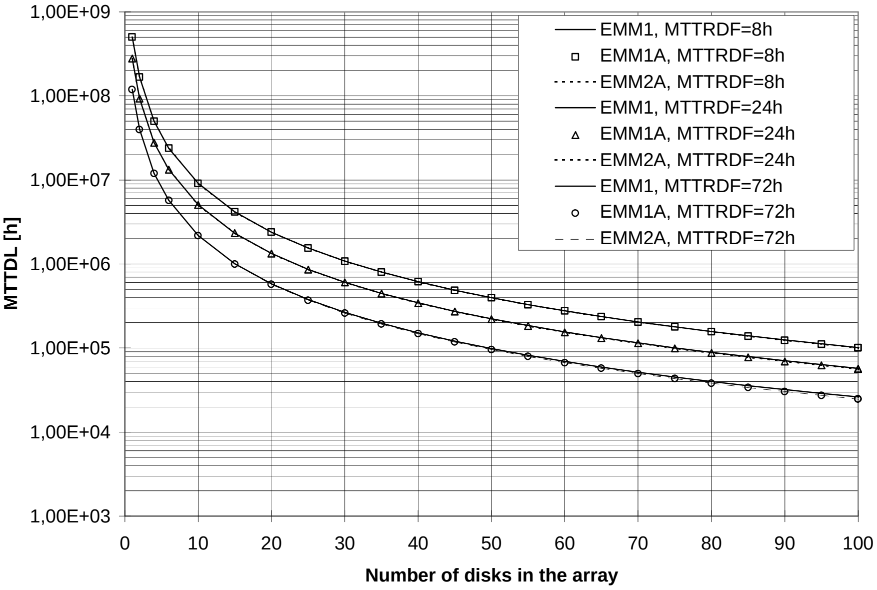

Figure D-1: MTTDL as a function of the number of disks

Disk unit failure rate (λ_d)

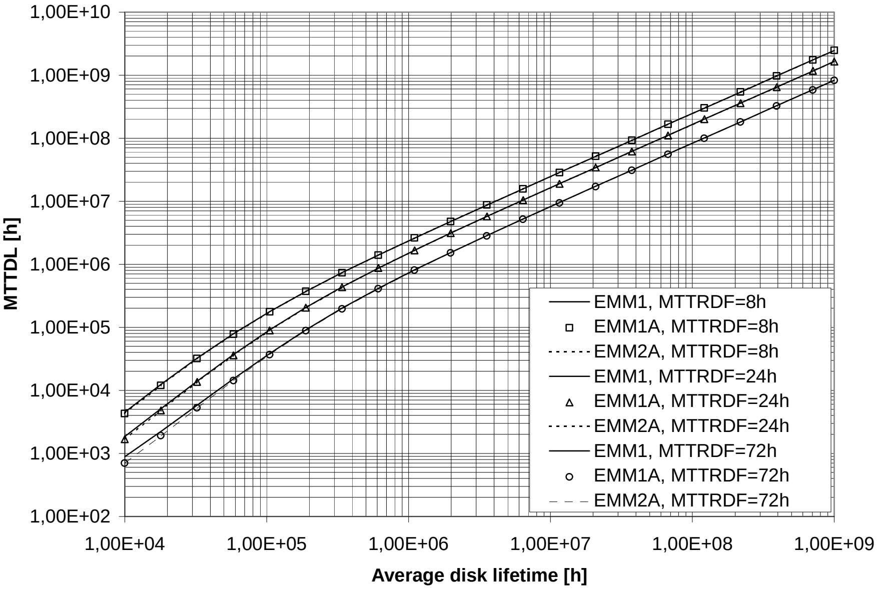

Figure D-2: MTTDL as a function of disk unit reliability

Sector failure rate (λ_s)

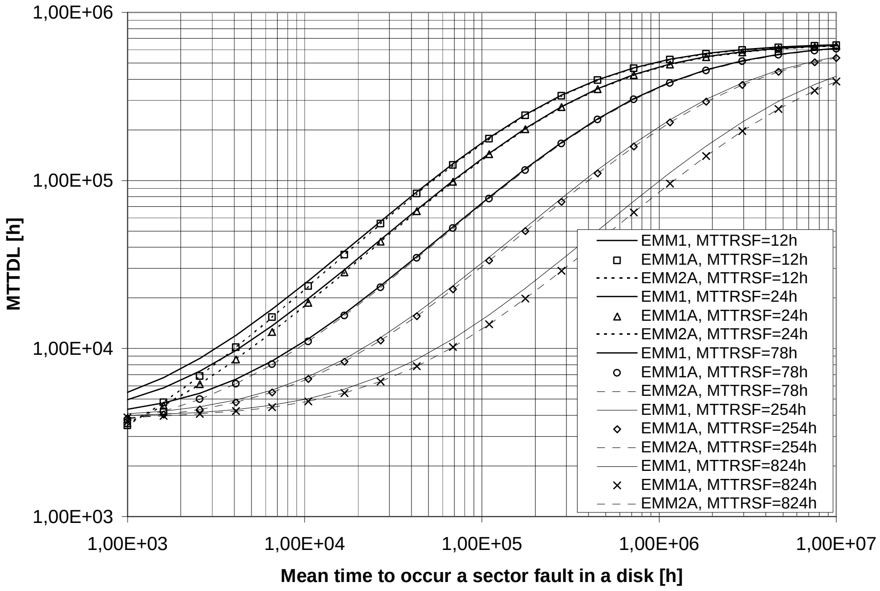

Figure D-3: MTTDL as a function of sector failure rate

Second disk unit failure rate (λ_{df})

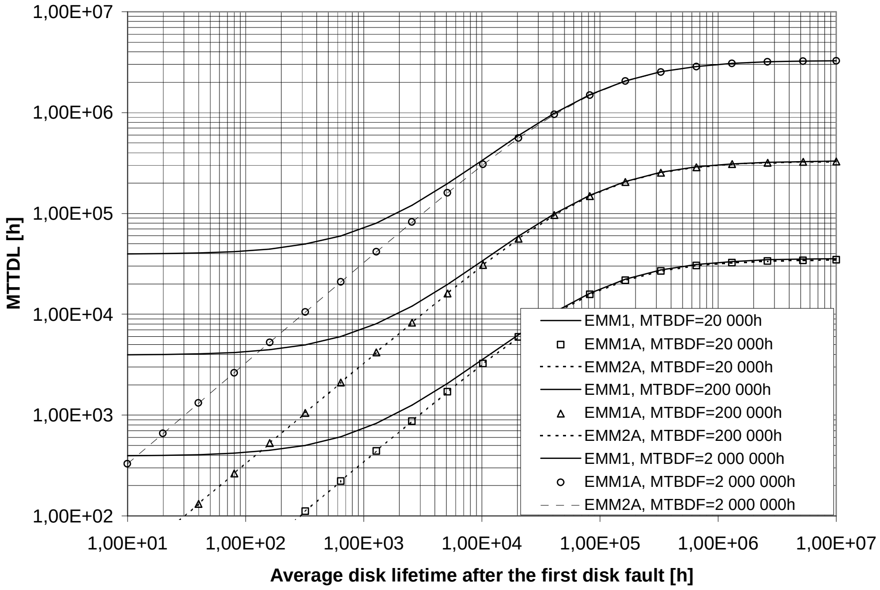

Figure D-4: MTTDL as a function of second disk failure rate

Disk unit repair rate (μ_d)

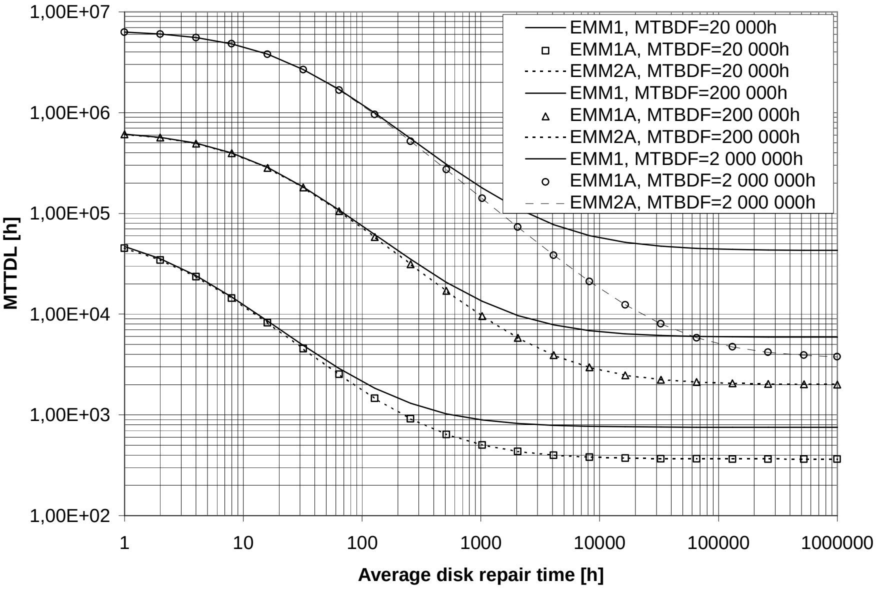

Figure D-5: MTTDL as a function of disk unit repair rate

Sector fault detection and repair rate (μ_s)

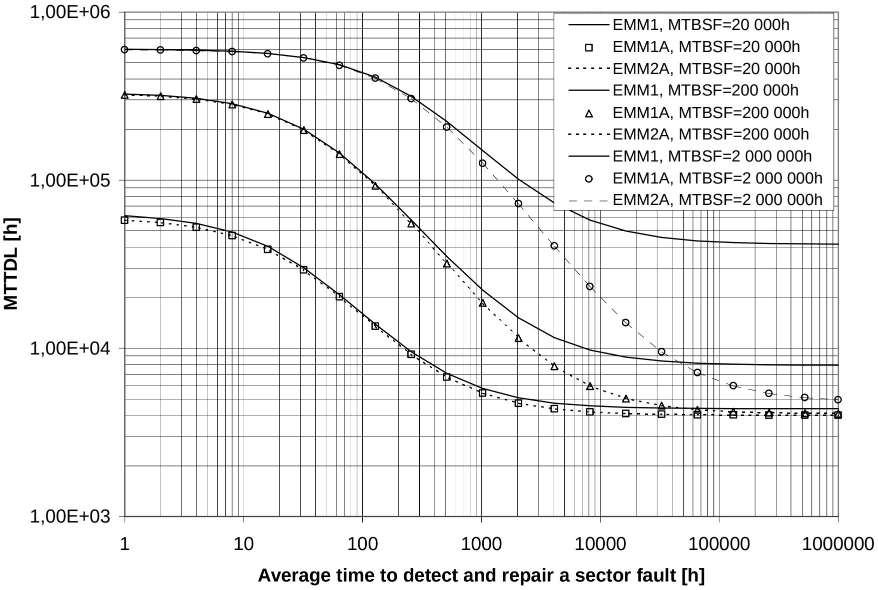

Figure D-6: MTTDL as a function of sector fault detection and repair rate