ELASTICALLY REPLICATED INFORMATION SERVICES

By

Jose E. Torres-Berrocal

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
COMPUTER ENGINEERING

UNIVERSITY OF PUERTO RICO
MAYAGÜEZ CAMPUS

2004

Approved by:

Jaime Seguel, Ph.D. **Member, Graduate Committee**	**Date**
Manuel Rodriguez-Martinez, Ph.D. **Member, Graduate Committee**	**Date**
Bienvenido Velez-Rivera, Ph.D. **President, Graduate Committee**	**Date**
Haedeh Gooransarab, Ph.D. **Representative of Graduate Studies**	**Date**
Jorge L. Ortíz-Alvarez, Ph.D. **Chairperson of the Department**	**Date**

Abstract

This thesis introduces elastically replicated information systems (ERIS). ERIS are distributed storage clusters (DSC) capable of sustaining their availability over a threshold value even in the presence of topological changes to the configuration of the system by dynamically adjusting their replication level and object allocation scheme. Such topological changes may be caused by external factors such as changes in demand or dynamic repartitioning of resources, but also by internal factors such as storage node failures. Various replication methods are reviewed and compared to ERIS. A simple mathematical model of a DSC is introduced which forms the basis of several simulation results characterizing the availability of various replication schemes in response to changes in the number of nodes. We present experimental data from a DSC simulator used to compute the availability of a DSC under alternative replication schemes. These results demonstrate that availability decreases quickly enough to render elastic replication necessary even in DSC's with tens of nodes. The results also validate the hypothesis that static replication levels are not enough to guarantee a sustained level of availability. Finally, we exploit some observed patterns in the results in order to synthesize an elastically replicated scheme and demonstrate its ability to sustain availability above our target level as nodes are added to a DSC.

Resumen

Esta tesis introduce los Sistemas de Información Elásticamente Replicados (ERIS). ERIS son conjuntos distribuidos de almacenaje (DSC) capaces de sostener su disponibilidad sobre un valor de umbral incluso en la presencia de cambios topológicos a la configuración del sistema ajustando dinámicamente su nivel de replicación y esquema de localización de objetos. Tales cambios topológicos pueden ser causados por factores externos tales como cambios en demanda o repartición dinámica de recursos, pero también por factores internos tales como fallas en los nodos de almacenaje. Varios métodos de replicación son revisados y comparados con ERIS. Diseñamos un modelo matemático simple de un DSC que forma la base de varios resultados de simulaciones que caracterizan la disponibilidad de acuerdo a varios esquemas de localización de objetos en respuesta a cambios en el número de nodos en un DSC. Presentamos datos experimentales de un simulador utilizado para computar la disponibilidad de un DSC bajo esquemas alternativos de replicación. Estos resultados demuestran que la disponibilidad disminuye tan rápidamente que hace la replicación elástica necesaria, incluso en sistemas DSC con solo decenas de nodos en discordancia con sistemas de replicación fija como en sistemas RAID. Los resultados también validan la hipótesis que niveles constantes de replicación no son suficientes para garantizar un nivel sostenido de disponibilidad. Finalmente, explotamos algunos patrones observados en los resultados en orden de sintetizar un esquema elásticamente replicado y demostramos su habilidad de sostener la disponibilidad sobre el nivel del objetivo de disponibilidad mientras se añaden nodos al DSC.

Dedication

I want to dedicate this thesis work to my loving family. To my parents, Wilda Joan and Jose Rafael, for giving me their support and liberty to pursue my own path. To my grandmother, Gloria Zegarra, for having me on her prayers. And finally to my love and caring wife, Maria Ivelisse, for her support and patience.

Acknowledgements

I gratefully acknowledgement the financial support for this work from the Program for Research in Computing and Information Sciences and Engineering (PRECISE) under the NSF-EIA Grant 99-77071. To Dr. Bienvenido Velez, thanks a lot for giving me the opportunity to work in this project, and for the opportunity to be your teaching assistant in the Advanced Programming course. I would like to thank Dr. Edgar Acuña Fernandez and Dr. Robert W. Smith for their help on Stochastic Processes. Also, I want to thank William T. Vetterling for his consent to use part of the source code from the "Numerical Recipes in C" book. A special acknowledgement to Dr. Manuel Rodriguez Martinez and Dr. Jaime Seguel for accepting to be members of my graduate committee, and for a great experience during your courses. To Pablo J. Rebollo for his support and diligence in the PRECISE laboratory.

I also wish to thank the Department of Electrical and Computer Engineering for giving me the opportunity to pursue the master degree in Computer Engineering coming from Computer Science.

Finally, I wish to thank my dear wife Maria Ivelisse. Her enthusiasm helped me go through this hard period of my life and motivated me to finish this thesis report.

Table of Content

List of Tables

Table **Page**

List of Figures

Figure **Page**

Figure **Page**

Figure **Page**

List of Symbols and Abbreviations

Abbreviator	Meaning
ERIS	Elastically Replicated Information System
DSC	Distributed Storage Cluster
RAID	Redundant Arrays of Inexpensive Disks
MTTF	Mean Time To Failure
MTBF	Mean Time Before Failure
SM	Storage Matrix
MTTR	Mean Time To Repair
S	Probability of survival
S_N	Probability of survival for N nodes, where N is the number of nodes
P	Probability of failure
P_N	Probability of failure for N nodes, where N is the number of nodes
λ	Rate of failure of one node
λ_n	Rate of failure of the DSC
DSC_{MTTF}	MTTF for the DSC
ΔSn	MTTF difference between step n and step n-1, where n is the number of the step.
C	Number of total copies
D	Down tagging
pr	Percent of replication
MOD	Modulus operation
n_obj	Number of original objects
node_count	Number of nodes
RES	Relative Effective Space
ES	Effective Space
UES	Up effective space
DES	Down effective space

List of Appendixes

Chapter I
Introduction

State-of-the-art high-capacity storage systems are typically assembled by interconnecting several off-the-shelf disks through some storage area network (SAN) together with a layer of software implementing an image of a single storage system. We call this type of system a *distribute storage cluster* (DSC). To compensate for the potential reduction in reliability induced by the high numbers of components with independent failure modes, some of the storage units hold enough redundant information to reconstruct any information in the advent of a disk failure. It is common to encounter high-end storage systems that support the dynamic addition of storage units, also known as hot swapping, to the system without suspending its operation. The popularity of the Internet and the consequent flourishing of e-commerce, have dramatically increased the importance of a hot swapping capability from a requirement of few high end systems affordable only by large corporations, to a common necessity of any firm wishing to provide continuous 24/7 operation.

We are particularly interested in DSC's to which nodes could be added or removed at any time during the operation of the system and that must operate continuously. Such systems are becoming part of many IT departments offering services over the Internet, since there is always somebody awake somewhere in the world that can access the service. This design constraint forces the system to conduct as many

configuration changes as possible while the system is operating, thus there is a need to automate the process of determining and implementing any actions required to compensate for the consequent loss of data availability due to the topological changes. Availability in general terms is the capacity of a system to satisfy successfully the request for its resources at a particular moment in time.

The obvious way to achieve redundancy is by making copies or replicas of the existing objects; a process that is commonly called *replication*. Our research explores the use of replication as a means for sustaining the availability of a distributed storage cluster above some fixed threshold even while the system suffers dynamic changes to its configuration or *topology*. Even thought there are extensive works on applying replication to achieve reliability and availability, many of the systems proposed in the past either had a constant replication scheme or used replication in a somewhat ad-hoc or naïve manner. A constant replication scheme may result in a continuous decrease in availability as the number of components in the system increases. As the number of components increases so does the number of opportunities for the system to fail. In this research, I present experimental data sustaining this claim.

This work explores the feasibility of designing DSC's capable of sustaining their availability above a fix threshold value even in the presence of topological changes to the configuration of the system by dynamically adjusting their replication level and object allocation scheme. **Our central hypothesis is that distributed storage clusters (DSC) that suffer dynamic changes to their configuration or *topology* should dynamically**

adjust their replication levels in order to sustain data availability above a given threshold.

Several factors may induce dynamic changes in the topology of a DSC. Some *external* factors may include seasonal changes in storage demands, or dynamic repartitioning of resources due to load balancing, among others. However, we foresee that node failures by themselves will become a major internal factor driving the frequent reconfiguration of DSCs. When a node fails, the DSC should dynamically compensate for the potential loss of data, redundant or otherwise, in such a way as to bring the system back to a level of availability above the target threshold.

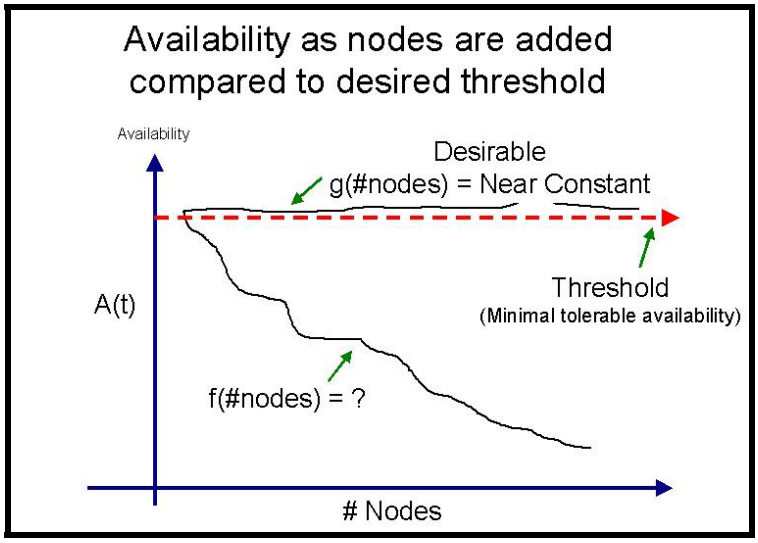


Figure I-1. f(#nodes) reflects the availability of a DSC with a constant replication scheme. Our goal is to design dynamic replication schemes yielding an availability response similar to g(#nodes).

Figure I-1 shows a hypothetical plot of DSC availability vs. number of nodes. It also shows three lines. The horizontal dotted line represents the minimal level of availability that a given DSC should provide at all times. We call this the *target*

availability. The curve *f(#nodes)* denotes the expected availability response of a DSC that uses a constant replication level. Our goal is to develop dynamic replication algorithms achieving a constant level of availability at all times as depicted by the *g(#nodes)*. In order to achieve this goal, we must first understand the functional relationship between availability and DSC topology for different replication schemes. An important contribution of our work consists of a mathematical/simulation framework that can be used to conduct explorations in this direction.

Availability is easy to achieve via replication if one is not concerned with storage utilization and performance. We can simply keep as many copies of each object as the number of failures we want to tolerate. However, redundant data takes room that could otherwise be used for non-redundant data. Space utilization thus decreases. Moreover, the amount of work necessary to maintaining consistency among all copies increases at least linearly with the number of copies of the data. Our goal is thus to understand what is the minimal redundancy that is necessary to achieve our target availability.

History has demonstrated that we always find ways to use any extra storage that we get. The "need" for storage capacity grows faster than the amount of storage per dollar. Everyday more and more sites connect into the global network, and each site demands more online storage capacity due to the addition of more users and services. This implies that, even thought storage is becoming cheaper and denser we cannot assume an unlimited online storage capacity. Effective storage utilization is desirable not only for write-intensive, but also for read-only workloads as well. Data cannot be naively

replicated in order to achieve availability without concern for performance and utilization. In fact, we demonstrate that there is an inherent trade off between utilization and availability. Maximum availability can always be obtained at the expense of minimal utilization and vice versa.

When new storage nodes are added to a DSC the additional storage capacity can be used to increase availability in one of two ways: replication or migration. As mentioned before, *replication* is the process of storing multiple redundant copies of objects on different storage nodes. *Migration,* on the other hand, is the process of relocating data objects among storage nodes in order to reduce the probability of losing the object due to a node failure. For instance, migration could be used to move objects to components that are more reliable.

Migration by itself does not increase the amount of redundant objects in a DSC, but it increases storage utilization by distributing objects among several available nodes. Replication reduces storage utilization because it increases the amount of space occupied by redundant copies. In write intensive systems, extensive use of replication may significantly degrade the system performance, since for each write event, a certain number of copies have to be updated in order to maintain consistency. **Our goal is designing DSC's that sustain their availability over a desired threshold by means of dynamically balancing replication and migration while simultaneously maximizing utilization.** We use the name *Elastically Replicated Information Service* (ERIS) to

systems that have the ability to sustain availability and maximize utilization using dynamic replication.

The need for ERIS arises in several different contexts and at different levels of abstraction. Examples of distributed storage systems include distributed file systems, RAID systems and distributed database systems, among others. In this work, we exploit a simple abstract model of a DSC and present data from simulations of the DSC's availability response under alternative replication and object allocation schemes. Our results indicate that some sort of elastic replication algorithm is needed to implement DSC's capable of sustaining a desired level of availability as the number of nodes in the DSC changes. For instance, a DSC with 35% replication and uniformly distributed objects within five (5) nodes will drop its reliability one nine (9) factor after increased to 10 nodes. Based on these simulations, we have found some availability patterns that we exploit to formulate one possible elastically replicated algorithm.

This dissertation is divided in the following chapters and sections. In chapter two, Previous Work, we present some basic mathematical background necessary to understand the rest of the document, as well as some previous methods used to sustain availability. In chapter three, we defined a mathematical model of a DSC then described the use of a DSC simulator for our research. Following is the Experimental Results chapter where it shows results based on the use of the DSC simulator and we explained a mathematical relationship between Availability, Replication and Tagging. In the same chapter, with the use of the mathematical relationship we described a way to estimate ERIS parameters

to obtain the object allocation scheme. Finally, we state our conclusions and propose future work in chapter five.

Chapter II
Previous Work

In this chapter, we present one section of methods currently used on DSC systems related with replication and then a section of mathematical formulations. The previous methods section is a brief summary among the extensive literature about replication, important to compare and to support this research. The mathematical background section describes various accepted definitions and formulas needed to understand this investigation. We refer to the reader to **[16]** and **[26]** for the basic definitions of the following concepts: Nondeterministic experiment, Random variable, and Probability.

A. Previous Methods

In general, data replication has been a subject of active research for over 30 years. The literature on the subject is extensive, but due to space limitations, only an extract of the most relevant previous work is presented here. Approaches to data replication can be classified into four different groups: Consensus Based, Data Trading, Caching, and RAID.

Consensus-based systems, also known as voting replication system, disseminate replicated data to various nodes in a network with a certain degree of correctness. In order to identify the correctness of the data that arrive to a given node during any event, the method requires comparing the data that arrived to that of the other nodes.

Since the correctness of the data at each receiving node in the network is uncertain, in order to decide whether the data is correct, the method uses consensus between the nodes. Each node casts a vote and the majority determines which data is correct. If a node does not belong to the majority, is either disabled or the data is resent to it. The availability of the system depends on the correctness of the data, which depends in turn, on the count of nodes involved in the voting process. The count of nodes has to be sufficiently large to overcome the "Byzantine failure assumption", that is, a majority of incorrect votes. Consensus-based replication is more concerned with the problem of coordinating object updates so that the latest update is always available; ERIS is more concerned with determining how many copies and where should they be stored in order to sustain a specific availability level. ERIS and consensus based systems complement each other. Examples of consensus-based systems are **[22]** and **[5]**.

The next replication method is Data Trading. This method is characterized, as the name suggests, by the exchange of data among nodes. For instance, if a node A has 1GB of data and 2GB of spare allocation space, and a node B, with a similar description, interchange part or all of their data, the result is that data from node A will be replicated in node B, and vice versa. The actual interchange is usually done through a communication network. This interchange is driven by the help of a set of particular policies. These policies determine the conditions under which the nodes form a relationship. The relationship can be a free exchange society or one based on economic factors. The rules involved in the relationship have to be determined, as well. For example, the amount of resources used in an interchange can be determined by a load

balancing scheme, or be restricted by time constraints, or by other type of policy. Although the level of replication in the system may vary dynamically, the system does not have enough control of how much its replication varies across the system. Data Trading was never designed with the same constraints as ERIS to sustain a specific level of availability. For instance, the system may reach a state in which some objects have no replication while other objects are excessively replicated. Examples of data trading systems include **[12]** and **[15]**.

In Caching, storage is partitioned so that the primary copies of objects remain separate from the replicas. Typically, the replicas are located closer to the data consumer and in smaller memories for faster access. This type of storage organization is primarily designed for performance instead of reliability. When the cached data is modified, the primary copies are modified according to a policy specifying how and when the change propagation is going to be performed. The Caching method is adequate for data that does not change too often. Otherwise, it requires a mechanism for getting consistent updates and conflict resolutions among the participating nodes. This is usually done by some consensus method. Two examples of caching systems are **[1]** and **[28]**.

Redundant arrays of inexpensive disks (RAID) are similar to ERIS in their use of replication to overcome the loss in reliability due to the use of multiple storage nodes with independent failure modes. Different RAID levels were originally proposed based on four parameters: the percentage of replication, the type of replication used (parity or full), the size of the data unit (striped or not striped), and the location or arrangement (distributed or not) of the units in the group of nodes (discs). What distinguishes RAID

from ERIS is that these parameters are normally fixed at the time the system is configured. Examples of RAID systems are: **[30]** and **[11]**.

A distributed storage system is said to be elastically redundant if it incorporates algorithms that automate the choice between migration and replication in order to maintain a desired level of availability in the presence of dynamic changes to the number of nodes. There have been distributed file systems that provide a level of replication that is fixed at system configuration time **[24]**. Others provide facilities to create objects with different levels of replication **[21]**. Deceit **[34]** provides operations to dynamically alter the level of replication of a file. However, none of these systems automatically adjusts the level of replication in response to a change in the number of nodes to sustain availability above a fix threshold.

B. Mathematical Background

Before we proceed to present the model of DSC used in the next chapter, we discuss some fundamental concepts of the theory of probabilities necessary to understand the remainder of the dissertation.

A *Bernoulli trial* is a nondeterministic experiment that results in one of two possible perpendicular or independent outcomes. The outcome of each trial depends on a predefined probability in such a way that its outcome can often be interpreted as either a *success* or a *failure*. A series of successive independent Bernoulli trials comprise a *Bernoulli process*. The probability $P[S_N = k]$ of obtaining k successes out of N

independent trials, given a probability p of success is:

$$P[S_N = k] = \binom{N}{k} p^k (1-p)^{N-k} \qquad (1)$$

where $\binom{N}{k}$ is the binomial coefficient, $\binom{N}{k} = \frac{N!}{k!(N-k)!}$

A Poisson process is a process satisfying the following properties. The numbers of changes in no overlapping intervals are independent for all intervals, the probability of exactly one change in a sufficiently small interval $t \equiv 1/n$ is $P = \lambda t \equiv \lambda/n$, where λ is the probability of one change and *n* is the number of trials, the probability of two or more changes in a sufficiently small interval ***t*** is essentially 0. In the limit of the number of trials becoming large, the resulting distribution is called a Poisson distribution.

A *Poisson process* differs from a Bernoulli process in that its outcomes describe the behavior of events occurring on a continuous time line. A Poisson process represents the limit, as $\Delta t \to 0$, of a Bernoulli process, taking one trial every Δt units of time and defining the probability p of a success as $p = \lambda \Delta t$. When a trial succeeds during one such trial, we say that an arrival has occurred. The probability $P(k)$ that k events arrive during a time interval t is:

$$P(k) = \frac{(\lambda t)^k e^{-\lambda t}}{k!} \qquad (2)$$

Availability $A(t)$ generally refers to the probability that a system is operating correctly at any given moment, while reliability is the property that a system can run continuously without failure. Reliability $R(t)$ is often measured in terms of "9's" (four 9's means that the system is available 99.99% of the time). When a system does not have repair time, $A(t) = R(t)$.

The mean time to failure (MTTF) defines the reliability property of a group of disks. It can have different values depending on the use of the disks in the group such that data can have different distributions, with or without redundancy, and with or without repair. The acronym MTBF (mean time before failure) is used interchangeably with MTTF, when the system does not have repair time ($A(t) = R(t)$).

The following formula is the currently accepted for the MTTF of a system compose of a group of disks, where the group can be divided itself in subgroups:

$$MTTF_{RAID} = \frac{(MTTF_{Disk})^2}{(D + C * n_G) * (G + C - 1) * MTTR}, \qquad (3)$$

where D is the total count of disks, C is the count of check disks or redundant disks per subgroup, G is the count of disks per subgroup, n_G is the count of subgroups, and $MTTR$ is the mean time to repair one disk for a system with repair time.

$$MTTF_{RAID} = \frac{(MTTF_{Disk})}{(D + C * n_G)}, \qquad (4)$$

for a system without repair time.

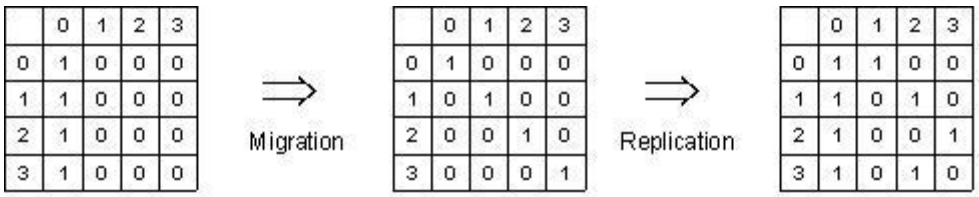
The previous concept can be translated to DSC systems changing the disks by any kind of node.

Chapter III
Modeling Distributed Storage Clusters

A. Mathematical Model of a DSC

For future reference, we will define the following terms. A *logical storage node* (or simply a node) is any indivisible unit capable of storing information. A *distributed storage cluster* (DSC) comprises two or more storage nodes. The member nodes of a DSC should function in a coordinated fashion to provide an illusion of a single storage system. We want to point out that these definitions apply to any storage system in question. In particular, we are interested in DSC's that allow dynamic reductions or additions of storage nodes with minimal system interruption.

We model a DSC as a Boolean O x D *storage matrix SM[o,d]*. The storage matrix relates objects to the storage nodes where they reside at any given point in time.



	0	1	2	3
0	1	0	0	0
1	1	0	0	0
2	1	0	0	0
3	1	0	0	0

	0	1	2	3
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1

	0	1	2	3
0	1	1	0	0
1	1	0	1	0
2	1	0	0	1
3	1	0	1	0

Figure III-1. Impact of Migration and Replication on DSC Matrix.

We set *SM[o,d]* to 1 when at least one copy of object *o* resides in node *d*. *SM[o,d]* equals zero otherwise (See Figure III-1 above).

We assume that all objects stored in the DSC are equally necessary. This implies that the system is available if at least one copy of each object is available at some node. Otherwise, we encounter a *system failure*. The assumption that all objects are equally necessary significantly simplifies our analysis, but may seem to induce some loss of generality. However, any storage system supports transactions that require a group of objects to be available in order to operate. Our assumption simply considers each such transactional group as having a separate availability goal. Different transactions may require different groups of objects, each of which may have a different availability target depending on the target probability that the corresponding transaction can take place.

The *Availability* $A(t)$ of a system at some point ***t*** in time refers to the probability that the system is operating at time ***t*** whether or not it has suffered from any individual node failure. In our case, the DSC is available until it encounters a system failure. In the introduction, we explained that replication can be used to avoid the failure of the system. Note that if one node of the DSC gets unreachable the objects in that node are also unreachable, but if those objects were replicated in other nodes of the DSC that remain reachable, the objects can still be accessed, and therefore the system as a whole remains available.

Another simplifying assumption is that all DSC storage nodes are connected to one another. We treat a lost connection to a node as a node failure. We leave the relaxation of this assumption for future work.

For some simple object allocation schemes, it is relatively easy to formulate their availability response function in closed mathematical form. For instance, for the special case of a DSC with a uniform distribution of objects and storing one original of each object, the mathematical formulations are the following. By taking equation (2), with k = 0, t = 1 (this implies that the system starts with a refresh set of nodes every time), and λ = 1/MTTF, we obtain the probability S of survival of one node, $1/e^{\lambda}$. The failure of one node P = 1 – S. Equation (1) with k = 0 and N equal to the number of nodes in the system gives $P_N = (1 – (1/e^{\lambda}))^N$. Since successive Poisson processes produce a Poisson process, we can take P_N as the probability failure of the DSC as a whole. The λ_N of the DSC equals $\ln(1/(1-P_N))$ and $MTTF_{DSC} = 1/ \lambda_N$. The availability probability S_N of the DSC equals $1 – P_N$.

We define *tagging* as the action of marking, placing, or allocating at least one object into a node. Tagging percent is then the percent of nodes that have at least one allocated object. In our work, we used the term *tagging* to imply node utilization to avoid confusion with the generalized meaning of the word "utilization" which is often used for total space utilized by data, redundant or otherwise.

The availability of a storage node is a function of the *mean time to failure* (MTTF) of the node. We selected the MTTF metric for experimentation purposes because it is the most commonly used mathematical model of system availability. Manufacturers of storage hardware often provide reliability data for their products in terms of their MTTF.

B. The DSC Simulator

In order to test our hypotheses we implemented a simulator for the DSC model previously described to execute simulations. We use the DSC simulator as a tool to find the relationship between different parameters involved in the DSC survival, in particular replication percent, allocation schemes, and the count of nodes. By running the simulator with a strategic variation of these parameters, we intend to find a mathematical relationship among them. In our case, since we are not experimenting with real DSC's, but using a simulator, the experiments are pseudo-empirical. We validated the simulator results against currently accepted mathematical formulations for object allocation schemes for which such formulations were tractable.

The simulator accepts a storage matrix *SM*, its dimensions *O* and *D*, and the individual *failure rate* common to all storage nodes (This implies another simplifying assumption). The simulation algorithm consists of a typical discrete time event loop where iterations model the passage of finite time lapses or *ticks*. Each time around the loop, the simulator flips a biased fail/not-fail coin for each storage node in the system, based on its failure rate. The list of non-failed nodes is used to determine if at least one copy of each object is available at some available node. If this is not the case, the simulator has detected a system failure. Otherwise, the simulation continues. The elapsed time from the beginning of the simulation up to the first system failure is the actual time to failure. The mean time to failure is simply the average of the actual times to failure taken over several runs.

In order to simulate the availability response of an object replication/migration scheme we run the simulation several times with different storage matrices representing the changes in object allocation induced by the scheme.

The object arrangement in the DSC and the total amount of replicas of each object is a crucial factor in determining the availability, and the various costs of the system. The costs can include the effective space used, the capacity of space required to allow the allocation of all objects, originals and replicas, and performance, among others.

As an important first step of any simulation research, we must refine and validate the model and simulator used. As mentioned before the simulator is run several times to take an average time to failure. During the validation and refinement process of the research (instead of finishing at a predetermined number of repetitions, 500 repetitions), the simulator finishes when the average time to failure converges as determined by a specific tolerance value.

In particular, the formula used into the loop code is:

$$IF\left(\left(\frac{abs(previous-current)}{((previous+current)/2)*100}\right)\leq Tolerance\right) stop\ loop \qquad (5)$$

This means that the percent (%) of difference between the current value and the previous value has to be less or equal to the tolerance in order to stop running the simulation. To obtain a good confidence value, we set the tolerance equal to 0.0001, or 99.99% of difference. The comparison value used in the formula is called the

convergence factor; in particular, we used the standard deviation or the average. For instance, the tenth and eleventh simulation run has a tenth and eleventh deviation value, and during the eleventh run, the current deviation value is the current convergence factor, and the tenth deviation value is the previous convergence factor.

During the validation and refinement process, the deviation value was taken to be the convergence factor in order to be able to refine and validate against the Poisson formulations used, and during the remainder of the research, the average value was taken as the convergence factor.

In order to be able to stop the running average or deviation with a convergence process we need to calculate the average and deviations values in a one pass style while not knowing a priory the repetitions or final sample size, N.

The following deviation formula is used:

$$S = \sqrt{S^2} = \sqrt{\frac{1}{N-1}\left\{\sum_{i=1}^{N} X_i^2 - N\overline{X}^2\right\}} \qquad (6),$$

which can be rewritten as:

$$\sqrt{\frac{1}{N-1}\left\{\sum_{i=1}^{N} X_i^2 - \frac{\left(\sum_{i=1}^{N} X_i\right)^2}{N}\right\}} \qquad (7),$$

to permit a one pass style calculation.

Finally, we want to point out that the results presented in this dissertation were obtained by fixing the number of objects to one hundred (100) and varying number of storage nodes within a range of one (1) to a hundred (100).

Chapter IV
Experimental Results

A. Simulation Results

In order to test our hypotheses we conducted several simulations. The first simulation validated the availability response function generated by the simulator against the mathematical formulations discussed in chapter three for the case in which objects are uniformly distributed among storage nodes and no replication is used. The second simulation tests different object allocation mechanisms and various static combinations of replication and migration values. The third simulation tests a simple elastic replication scheme that sustains the availability of the DSC above a threshold while maximizing tagging.

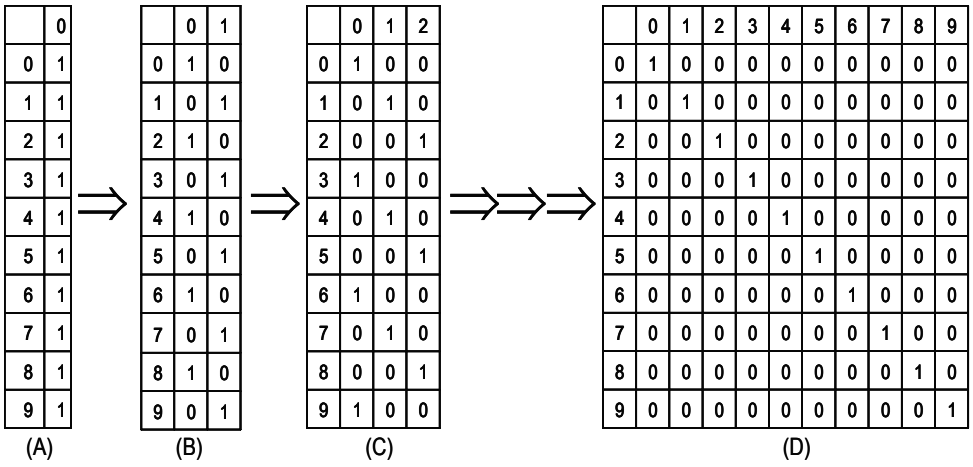


Figure IV-1. Uniform distribution. (A) DSC initial state. (B) DSC after adding one node. (C) DSC after adding next node. (D) Keep adding nodes until #nodes = #objects.

For validation purposes, we ran the simulator with a series of DSC's with increasingly higher numbers of nodes. Objects are uniformly distributed across all storage nodes and no redundant copies are kept. Figure IV-1 illustrates the storage matrix corresponding to this object arrangement. The theoretical and experimental results are depicted in Figure IV-2. The experimental line shows some variation around each point as illustrated by the box plots. However, both the theoretical and simulated results show remarkable similarity. Figure IV-2 is assembled with the data in Table IV-1 and the data in Table IV-2. Table IV-2 also shows that the deviation values correspond to the mathematical formulations used, in particular the curves does in fact trail a Poisson distribution. This result evidenced that our simulator was working as expected.

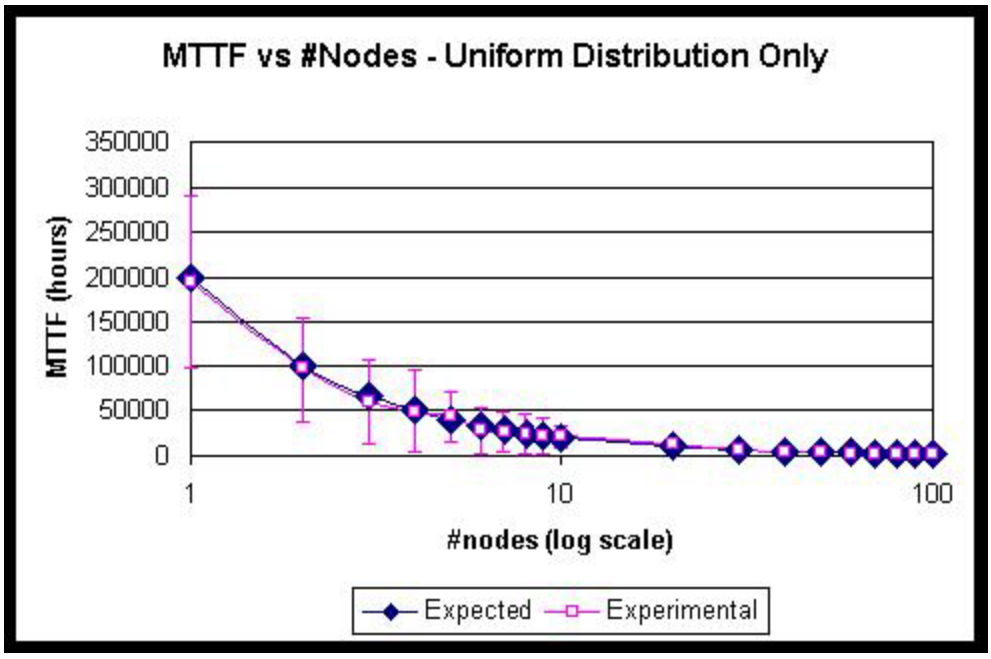


Figure IV-2. Validation Curve.

Another interesting observation that can be drawn from this result is that availability decreases by an order of magnitude even before the number of nodes reaches ten. This result suggests that the problem of dynamically sustaining availability is of importance even in rather small DSC's. The uniformly distributed arrangement achieves minimal availability since any node failure would make the system as a whole fail. However, uniform distribution achieves high storage utilization because all storage nodes are used to store objects.

Table IV-1. Calibration and System Confidence results used on Figure IV-2.

#objects	#discs	Average MTBF	Expected MTBF	%Error MTBF	System Confidence
100	1	194072	200000	2.964	0.999995
100	2	96688	100000	3.312	0.999990
100	3	60461	66666	9.307593	0.999985
100	4	49786	50000	0.428	0.999980
100	5	44196	39999	10.49276	0.999975
100	6	27837	33333	16.48816	0.999970
100	7	26832	28571	6.086591	0.999965
100	8	24680	24999	1.276051	0.999960
100	9	21764	22222	2.061021	0.999955
100	10	21178	19999	5.895295	0.999950
100	20	12205	9999	22.06221	0.999900
100	30	6271	6666	5.925593	0.999850
100	40	4900	4999	1.980396	0.999800
100	50	3426	3999	14.32858	0.999750
100	60	2808	3333	15.75158	0.999700
100	70	2298	2857	19.56598	0.999650
100	80	2485	2499	0.560224	0.999600
100	90	2174	2222	2.160216	0.999550
100	100	2001	1999	0.10005	0.999500

Table IV-2. Deviation results used on Figure IV-2.

#objects	#discs	Experimental Deviation	Expected Deviation	%Error Deviation
100	1	194330	194072	0.13294
100	2	95502	96688	1.226626
100	3	59337	60461	1.85905
100	4	46296	49786	7.010003
100	5	44460	44196	0.597339
100	6	27775	27837	0.222725
100	7	25574	26832	4.688432
100	8	22087	24680	10.50648
100	9	21757	21764	0.032163
100	10	20301	21178	4.14109
100	20	11556	12205	5.317493
100	30	5817	6271	7.239675
100	40	4699	4900	4.102041
100	50	3464	3426	1.109165
100	60	2684	2808	4.415954
100	70	2161	2298	5.961706
100	80	2050	2485	17.50503
100	90	2042	2174	6.071757
100	100	2040	2001	1.949025

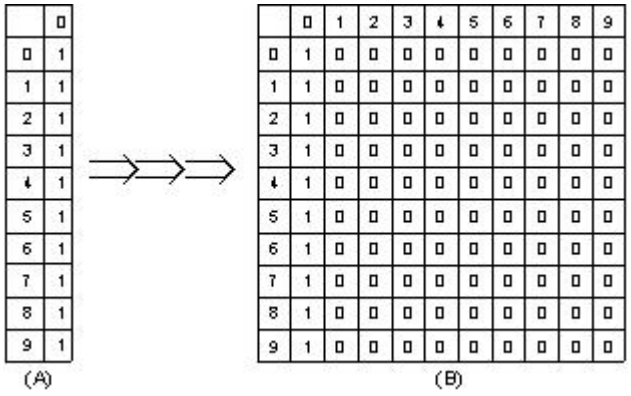


Figure IV-3. Centric mechanism. No distribution.
No matter how many nodes are added all objects are kept in the same node.

A second approach yields opposite results to uniform distribution. In this *centric* scheme, all objects are stored in a single node independently of how many nodes are available. Figure IV-3 illustrates the resulting storage matrix. Although this arrangement

is of little or no practical use, it does provide us with an upper bound of the availability that a system can achieve without replication. In this scenario, the system fails if one specific node of the system fails; the one that holds all the objects. Thus, the availability of the system is equal to the availability of a single node.

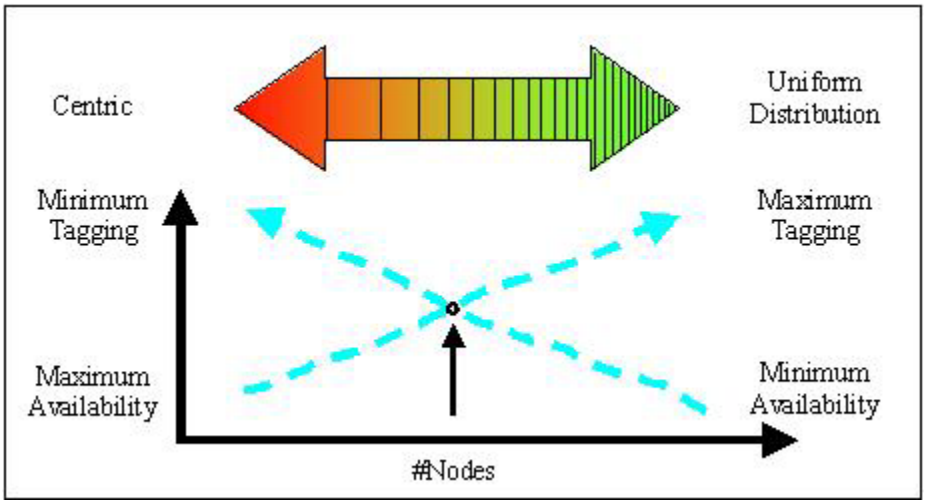


Figure IV-4. Availability and Tagging relationship.
The dotted curves are not meant to be understood as lines.

The uniform and centric approaches are opposite ends of the continuum of possibilities depicted in Figure IV-4. At one end of the spectrum (left) are those arrangements that maximize availability at the cost of space utilization. At the other end are those arrangements that maximize utilization at the expense of availability. Both objective functions are thus inherently opposed. Our goal is set at finding the point or points towards the center of the spectrum that sustains availability at some desired level, while simultaneously achieving maximal utilization across topological changes.

Figure IV-5 compares the experimental results obtained for the uniform and centric approaches. The centric curve shows quite some variation, but on average stays at the level of availability of a single node, which in this experiment was considered 1190 weeks or 200,000 hrs. **[12]**. We need to point out that due to those variations we decided to use the average value instead of the standard deviation value as the convergence factor in all our experiments.

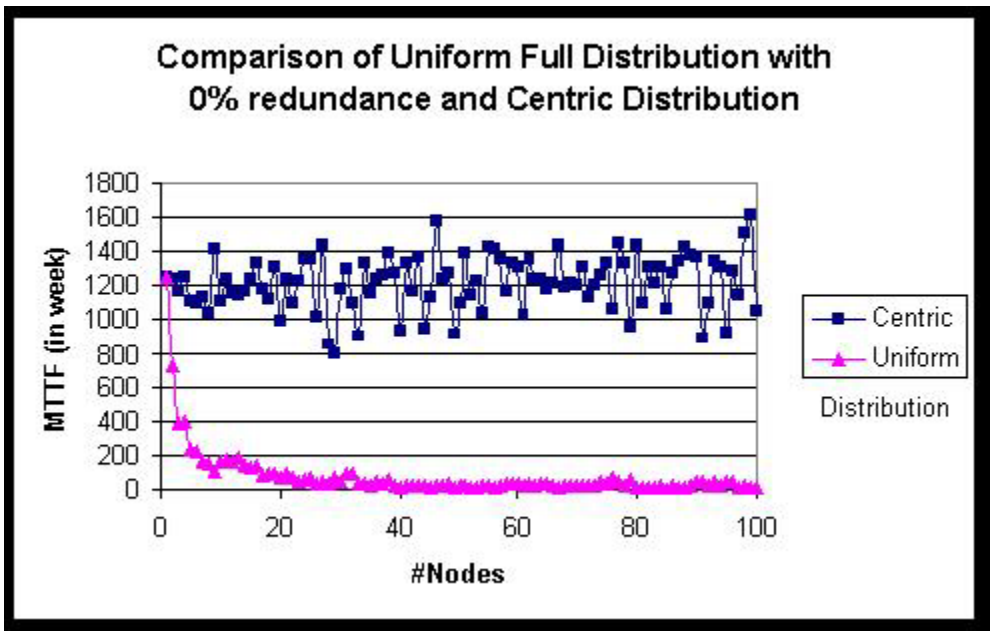


Figure IV-5. Comparison curves for Extreme algorithms

The next simulation compares the availability achieved by uniformly distributed scenarios with fixed replication levels. Replication is calculated as R = C/(C + O)*100, that is the percentage of objects that are redundant, where C is the number of object copies and O is the number of original objects. For instance, a replication level of 50% indicates that 50% of the objects are redundant copies. Figure IV-6 below shows the availability results of a uniformly distributed DSC with varying static level of replication.

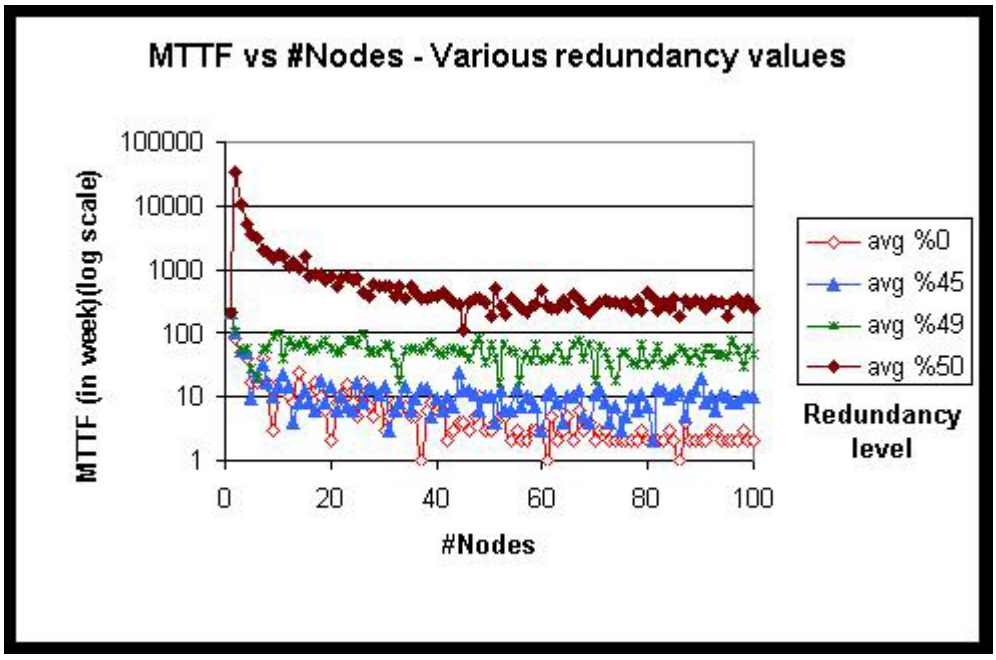


Figure IV-6. Uniform Distribution with various replication levels in logarithmic scale.

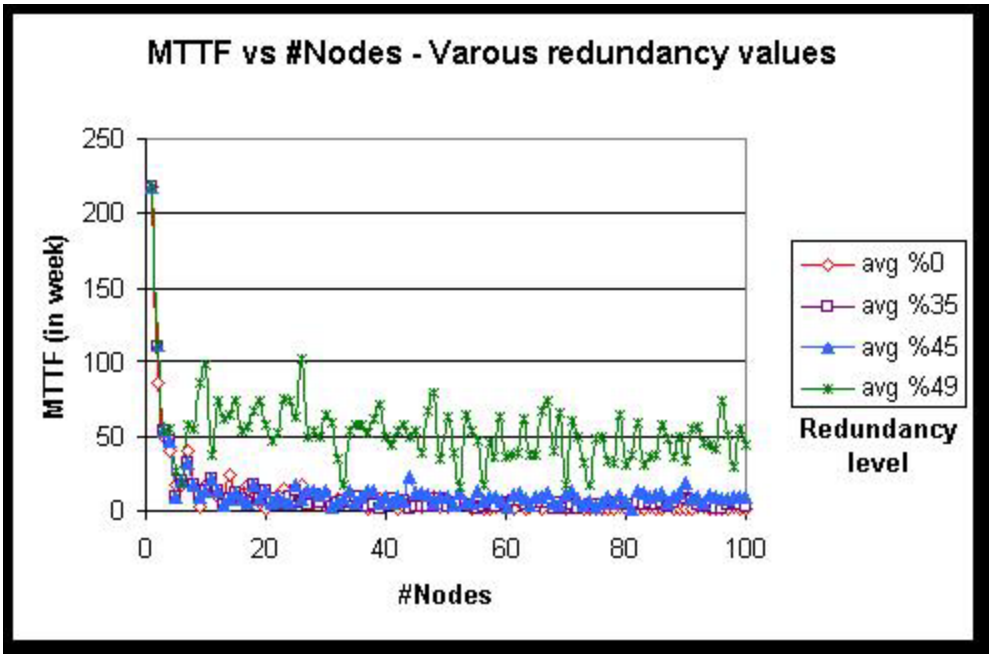


Figure IV-7. Uniform Distribution with various replication levels in normal scale.

These results indicate that even at fixed levels of replication, availability decreases rapidly as the number of storage nodes increases. If we fix the level of replication too high, we may achieve more availability than needed at the expense of a decrease in space utilization and performance due to the unnecessary additional copies. If we set it too low, we may not achieve the desired level of availability as the number of nodes increases. This result clearly evidences the need for some form of elastic replication.

In addition to the centric and uniform approaches, we tested some hybrid mechanisms as well. We partitioned the set of objects into two different levels of replication. More specifically one group of objects is replicated and another group is not. The resulting storage matrix can be viewed as divided into two regions. The *up* region holds the objects that are replicated while the *down* region holds the objects that have not been replicated. During the object allocation of original objects and its replicas, if the total replicas are less than an integral multiple of the total count of original objects, the reminder replicas are allocated top to bottom. This forms two regions in the matrix. The up region will have at least one additional redundant copy of each object than the down region. Each region can have different allocation schemes, which may give different tagging percentages. The upper region is said to have an *Up tagging* and the lower region is said that has a *Down tagging*. For simplicity and without loss of generality in both cases, objects are assigned to nodes in *left-to-right* fashion. Figure IV-8 (below) illustrates two views of such a matrix. The left view (a) shows how replication is calculated for the matrix while the right view (b) shows how tagging is calculated for the

up and down regions. The left matrix has six out of its ten objects replicated for a resulting replication level of 6/16 (about 37.5%). The objects 0-5 are stored in nodes 0-3, and objects 6-9 are stored in nodes 0-1. The matrix has an up region in which four out of ten nodes hold original copies for a resulting tagging level of 40%. The down region achieves a tagging level of 20%. In overall, the matrix has a 40% tagging since the up region include the nodes in the down region.

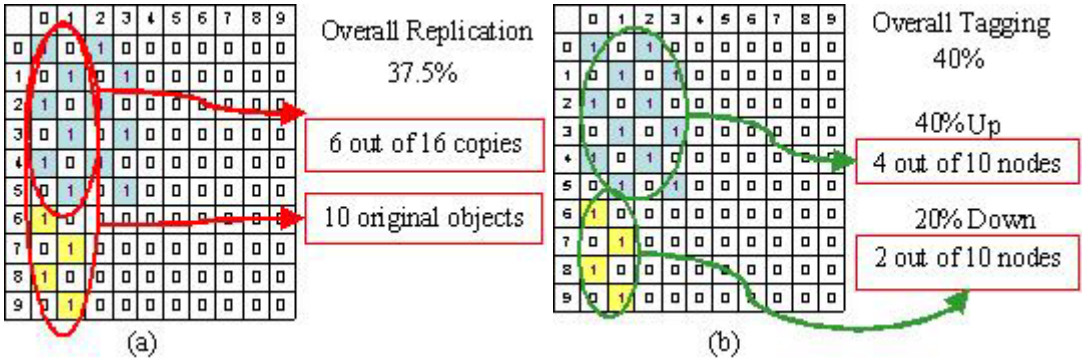


Figure IV-8. (a) Left to Right - Tagging sample DSC matrix. Replication calculation example (b) Left to Right - Tagging sample DSC matrix. Tagging calculation example.

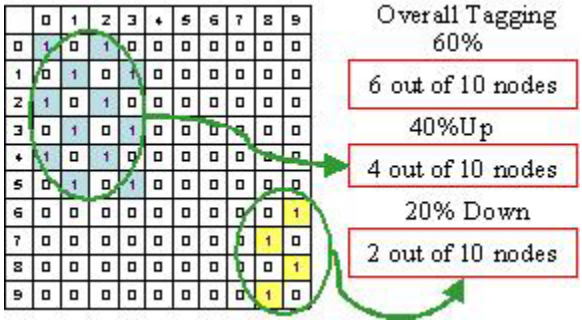


Figure IV-9. Balance - Tagging sample DSC matrix. Tagging calculation example.

Figure IV-9 shows an alternative arrangement in which the set of objects with replicas and the set of objects without replicas are stored in disjoint sets of storage nodes. Objects 0-5 are stored in nodes 0-3, and object 6-9 are stored in nodes 8-9, or in a

balanced arrangement. The replication factor and the tagging level of each region is the same as in Figure IV-8, but this balanced arrangement achieves an overall tagging of 60%, since the up and down regions use non-overlapping sets of nodes, and thus achieves higher utilization. In cases where the number of nodes is small or the number of objects is large, the balanced arrangement may yield sets of nodes used by the up and down regions that have some overlap. Although this object allocation scheme is not in itself an algorithm, it is not hard to visualize that a family of algorithms can be devised that simply maintain this particular object arrangement as an invariant. For this reason we use the terms elastically replicated *scheme* and *algorithm* interchangeably during the remainder of the dissertation.

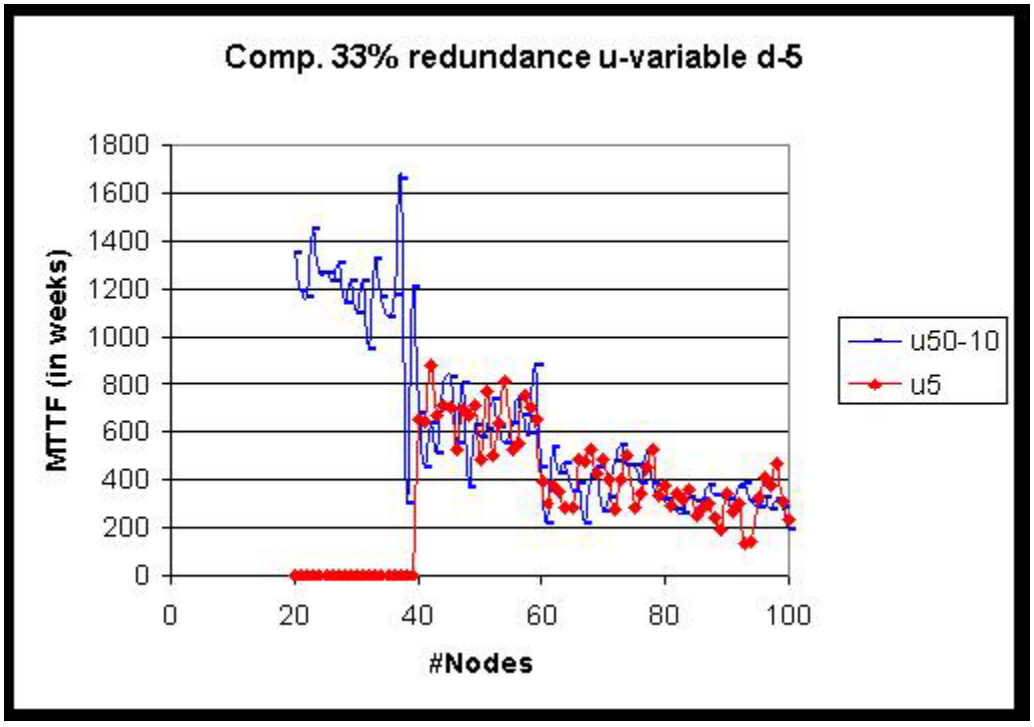


Figure IV-10. Hybrid algorithm setting variable node utilization or tagging in the Up region and fixed in the lower or down region.

Figure IV-10 (above) shows the availability of the hybrid algorithm with 33% replication with 5% tagging in the down region, but varying the Up tagging. (We used tagging factors from 5% to 50%) The factors 10% to 50% produced exactly the same MTTF response and the 5% factor responds with different values but with the same trend to the point that is hard to distinguish between the curves. This means that varying the tagging in the Up region does not appear to change the MTTF response as the number of nodes increases.

The following statements are an informal proof of why the Up region does not have an appreciable impact on the MTTF of the DSC:

Let A_p be the probability of survival of the DSC system.

Let A_1 be the probability of survival of the group of nodes in the DSC with its objects not replicated.

Let A_2 be the probability of survival of the group of nodes in the DSC with its objects replicated with one (1) redundant object each.

Let N_p be the total number of nodes of the DSC system with objects.

Let N_1 be the number of nodes of the group of nodes in the DSC with A_1.

Let N_2 be the number of nodes of the group of nodes in the DSC with A_2.

Let $N_p = N_1 + N_2$ then, $A_p = A_1 \bullet A_2$.

Since A_1 tends to zero (0) for $N_1 >> 1$, A_p tends to zero (0) for $N_1 >> 1$.

Then, the Up region does not have an appreciable impact because the Up region contains the nodes with the objects that are replicated with one (1) redundant object each, while the Down region contains the nodes with the objects without replication. Then $A_p = A_1 \bullet A_2$ where N_1 is the Down region and N_2 is the Up region, thus if $N_1 >> 1$ the Up region does not have an appreciable impact in the MTTF.

The MTTF is zero for the curve with 33% replication, 5% Up tagging, and 5% Down tagging for less than 40 nodes because the SM requires at least 40 nodes in order to be able to allocate all the objects in the DSC. Since tagging is an integer term, the nodes allowed to use by the tagging percent have to be an integer number of nodes.

The formula used to calculate the allowed nodes is the following:

Nodes_allowed_up = total_nodes * Up_tagging% (8a)

Nodes_allowed_down = total_nodes * Down_tagging% (8b)

Using the previous formulas, we obtain on integer calculations:

Nodes_allowed_up = 40 * 5/100 = 2; Nodes_allowed_down = 40 * 5/100 = 2

Nodes_allowed_up = 39 * 5/100 = 1; Nodes_allowed_down = 39 * 5/100 = 1

Nodes_allowed_up = 20 * 5/100 = 1; Nodes_allowed_down = 20 * 5/100 = 1

For 33% replication to allocate the original objects plus the objects copy, we need at least two (2) nodes. According to the calculations above, we need at least 40 total nodes in order to have two (2) allowed nodes. For the range of 1 to 19 nodes, the simulator would not even run because for less than 20 total nodes, the nodes allowed will be zero.

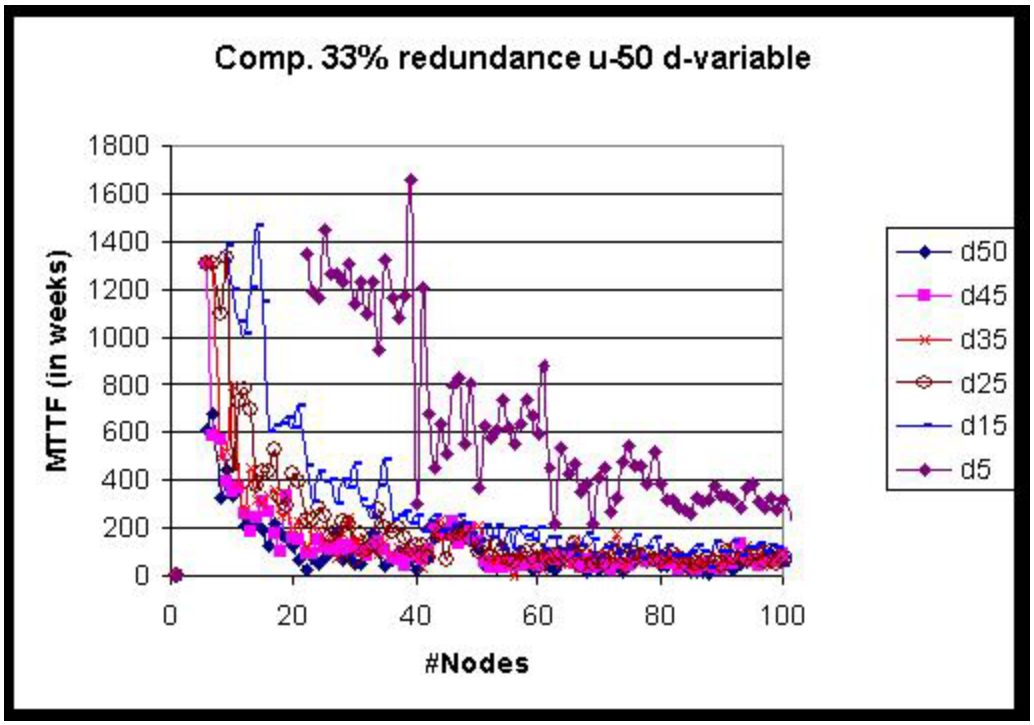


Figure IV-11. Availability of hybrid algorithm for variable down tagging and fixed up tagging at 33% overall replication.

Figure IV-11 (above) shows an example curve that uses the hybrid algorithm with 33% replication and 50% tagging in the Up region, but varying the down tagging. (We used down tagging factors from 50% to 5% at 5% increments) Each factor produced a different MTTF response. This means that by reducing the tagging in the down region the MTTF increases as the number of nodes increases. This result confirms prediction

that reducing the down tagging reduces the probability that a failed node holds objects without replicas.

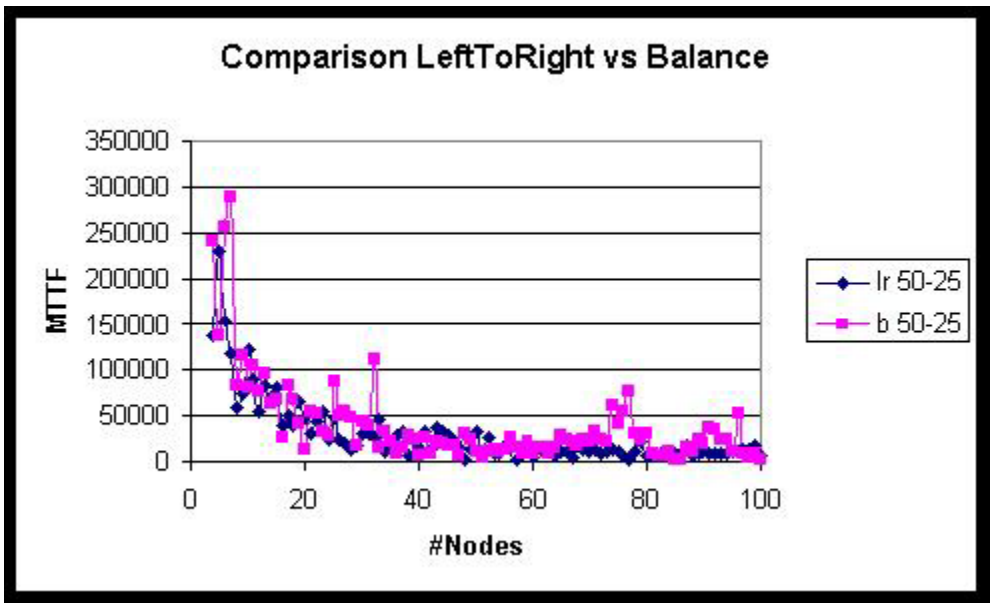


Figure IV-12. Left-to-right(lr) and Balance(b) mechanisms with 35% replication, 50% Up tagging, and 25% down tagging.

We compared both the *left-to-right* and the *balanced* arrangements by running several simulations with the same tagging factors on the corresponding regions. The results consistently demonstrated that the availability of both arrangements was very similar, as can be seen from the example shown in Figure IV-12. The left-to-right allocation differs with the balance allocation in the way the objects with 0% replication is allocated. The objects with 50% replication are allocated equally in both schemes. Since the region or group of nodes with 0% replication objects is the critical factor, as explained earlier, the two schemes behave the same. Therefore, we decided to use the balanced mechanism since it achieves higher overall tagging than the left-to-right mechanism.

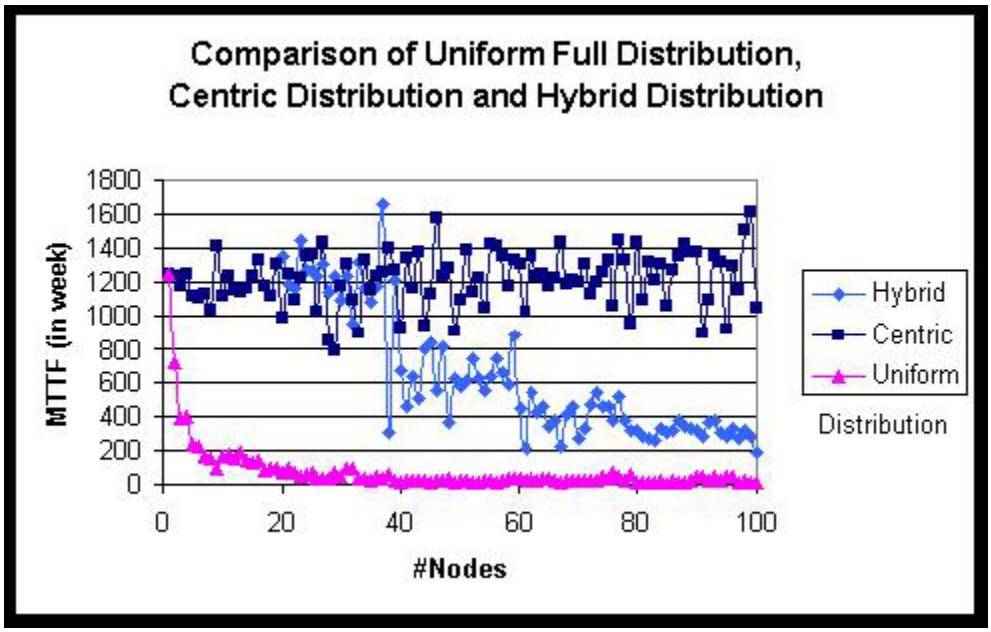


Figure IV-13. MTTF comparison between the uniform and centric algorithms and a hybrid algorithm. Hybrid plot is for Up tagging of 50%, down tagging of 5% and 33% replication.

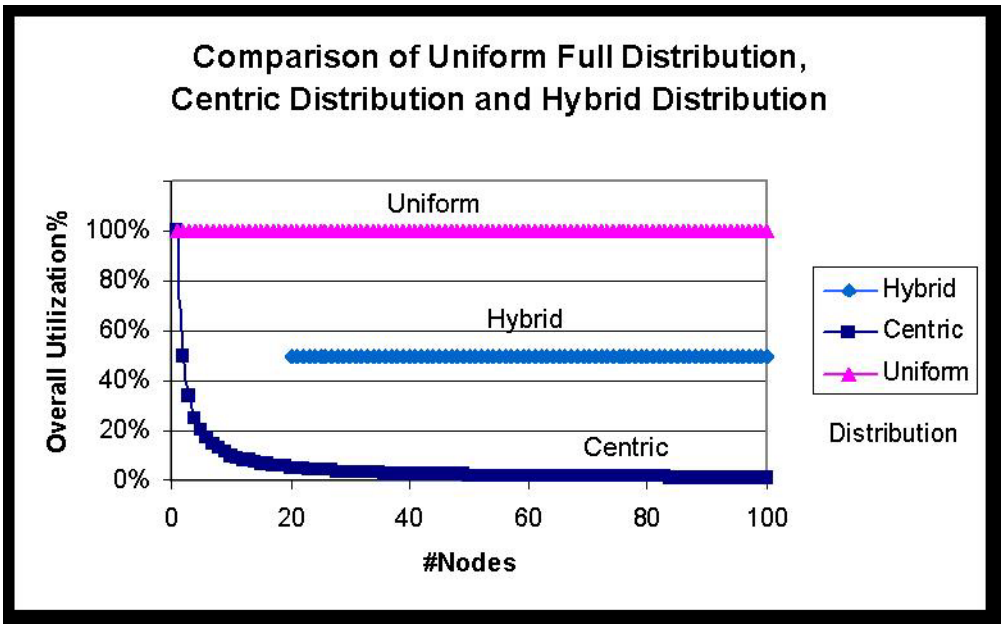


Figure IV-14. Tagging or overall utilization% comparison between the basic algorithms and the hybrid algorithm. Hybrid plot is for Up tagging of 50%, down tagging of 5% and 33% replication.

Figure IV-13 and Figure IV-14 plot the availability and tagging achieved by the centric, uniform and hybrid approaches. Overall utilization decreases with the centric approach while the MTTF decreases in the uniform approach as the node count increases. The hybrid algorithm falls in the middle of both extreme approaches as expected in terms of both utilization and MTTF, and sustains the availability of the DSC longer than the uniform full distribution algorithm. In particular, the hybrid curve shown, which is for a curve that uses Up tagging of 50%, Down tagging of 5% and 33% replication, sustains the availability up to 40 nodes to the same level of the Centric algorithm, and continues holding up availability significantly higher than the Uniform approach all the way up to 100 nodes.

In Figure IV-15(a), we show a DSC example after the algorithm has been applied, with one family of values and a set of parameters for tagging and replication. Note that object 5 is tagged in node 0 and node 5, but it is only counted once. Also, note that node 8 is tagged by objects 4, 8 and 9, but it is only counted once. Then in this case, all nodes are tagged at least once, giving 100% general tagging. The space that appears unoccupied is not wasted. As shown in Figure IV-15(b), this space can be used by other groups of objects (denoted by letters A-D) that use a complementary form of the algorithm.

nodes/ objects	0	1	2	3	4	5	6	7	8	tags	untag
0	1	0	0	0	1	0	0	0	0	2	7
1	0	1	0	0	0	1	0	0	0	2	7
2	0	0	1	0	0	0	1	0	0	2	7
3	0	0	0	1	0	0	0	1	0	2	7
4	0	0	0	0	1	0	0	0	1	2	7
5	1	0	0	0	0	1	0	0	0	2	7
6	0	1	0	0	0	0	1	0	0	2	7
7	0	0	1	0	0	0	0	1	0	2	7
8	0	0	0	0	0	0	0	0	1	1	8
9	0	0	0	0	0	0	0	0	1	1	8

(a)

nodes/ objects	0	1	2	3	4	5	6	7	8
0	A	B	C	D	A	B	C	D	0
1	0	A	B	C	D	A	B	C	D
2	D	0	A	B	C	D	A	B	C
3	C	D	0	A	B	C	D	A	B
4	B	C	D	0	A	B	C	D	A
5	A	B	C	D	0	A	B	C	D
6	D	A	B	C	D	0	A	B	C
7	C	D	A	B	C	D	0	A	B
8	0	0	0	0	0	0	0	0	A
9	0	0	0	0	0	0	0	0	A

(b)

Figure IV-15. (a) d = 12%, u = 100%, pr = 48%, n = 9, o = 10; SM with 9 nodes and 10 objects. d - Down region tagging%. u - Up region tagging%. pr - replication%. n - node count. o - object count. General Tagging = 100%. (b) The same SM as Figure IV-15(a) but with the effective space used.

The group objects is a collection of objects that share the same allocation scheme. Each group object differs in the starting column in the SM. That way the effective space is used while keeping the same probability of loosing any of the objects. The table shows an example SM with groups labeled A to D. Each object A belongs to group object A, objects B belong to group object B, and so on. To clarify, the fact that the table shows all objects in its group object with the same "letter", that does not mean that the objects are replicas of each other, they are just labels to identify that an object belongs to a particular group object, or the same, to a particular allocation scheme. Each group of objects has to have equal ratio of replication. For instance, objects zero (0) to seven (7) have groups A to D with one copy each. Remember the Up regions have 100% tagging. The lowest region has only one family, because in this example the D-tagging allow only one node to be used of the nine (9) nodes.

For the example in Figure IV-15(b) we calculated the *relative effective space* (*RES*) from the following equations: *RES = (ES/(node_count * n_obj))*100*, where *ES* stands for Effective Space. *ES = UES + DES*, where *UES* stands for Up effective space and *DES* stand for Down effective space. *UES* can be calculated from the example DSC in Figure IV-15(b), as 8 objects * 8 columns. Again, *DES* can be calculated from the example DSC as 2 objects * 1 column. Both, *UES* and *DES*, have a general equation that we used to calculate the values for different DSCs.

B. Finding a Mathematical Relationship between Availability, Replication and Tagging

An elastic replication algorithm could be developed based on all the previous results once we obtain a mathematical relationship between availability, redundancy and tagging. The algorithm would simply select the minimal replication level and maximum tagging level that achieves the desired availability and will replicate and migrate the necessary objects to bring the allocation scheme to within these parameters.

In order to uncover a relationship between availability, tagging and replication we conducted various simulations varying the replication and tagging parameters, to obtain families of curves of availability. From these curves, we planned to perform curve fitting and empirical experimentation to obtain a generalized formula that relates replication, tagging and availability. We found that the Up region tagging does not significantly influence the availability of the DSC and decided to set the up tagging to 100%. Figure IV-16 and Figure IV-17 show that the availability of a DSC with 1% replication

and another with 49% replication is not significantly different over the MTTF range of 50,000 hrs to 200,000 hrs. The dominant factor is the down tagging. Both curves differ under 50,000 hrs where the curves of Figure IV-16 continue its availability decrease as the node count increases. In contrast, the curves in Figure IV-17 sustain the availability at a higher level.

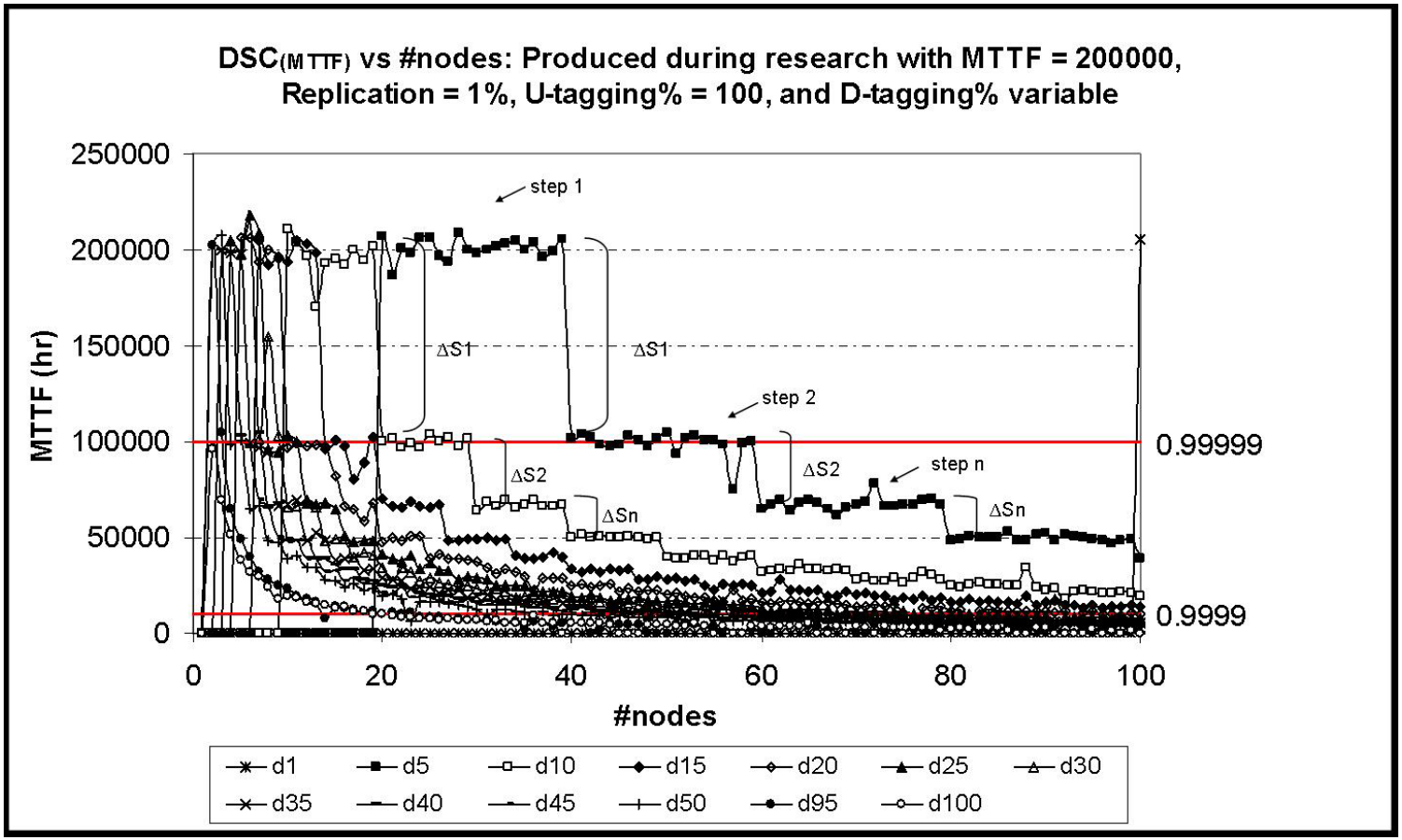


Figure IV-16. DSC curve family for 1% replication.

To sustain the availability of the DSC over 50,000 hrs we need to set the replication level to 49%. In terms of availability 200,000 hrs translates to 0.99999 (five nines) available time. Since we are using the Poisson distribution the probability of survival is given by $P=\lambda\Delta t$. We set $\Delta t=1$ and $\lambda=1/MTTF$, so $P=1/MTTF$. Then the number of 9's is obtained by calculating the inverse of the MTTF value. Thus, 200,000 hrs and 100,000 hrs have the same number of 9's, being 100,000 hrs the lower bound to the corresponding range of number of 9's.

To sustain the availability of the DSC at 0.99999 we could set the replication from 1% to 49% and it would make no difference. Recall that A_1 tends to zero (0) for $N_1 >> 1$, A_p tends to zero (0) for $N_1 >> 1$. That is the case for 1% to 49% replication. Recall that $A_p = A_1 \bullet A_2$. In the case of 50% replication, $N_1 = 0$, thus $A_1=1$ giving $A_p = A_2$. Then, since A_2 tends to the same order of magnitude of the probability of survival of one node, there is a big difference between 49% and 50% replication.

Another effect of our algorithm is that the availability of the DSC takes a stepwise form. Each step is repeated on each curve and they are a function of the tagging percentage. Recall that the DSC is modeled with the SM where the objects are represented by a single value. Each object is indivisible so the object exists or not. You cannot have parts of an object, to recover a lost object a complete redundant copy of the same object has to exist. This means that when you replicate and object, it has to be an object of integral size. For the same reason, if you want to allocate an object it has to be allocated in whole. At the instance of filling the SM with a particular replication percent, the redundant copies have to be an integer number of objects. Recalling that replication

percent, R% is defined as R%=C/(C+O)*100, then solving for C gives C = round ((O*R)/ (100-R)), using the round function to get integer objects. Thus in order to have a change in availability a modulus factor of the number of nodes is required to accommodate an integral count of replicated objects.

From these curves, we can try to obtain a relationship between replication, tagging, and availability.

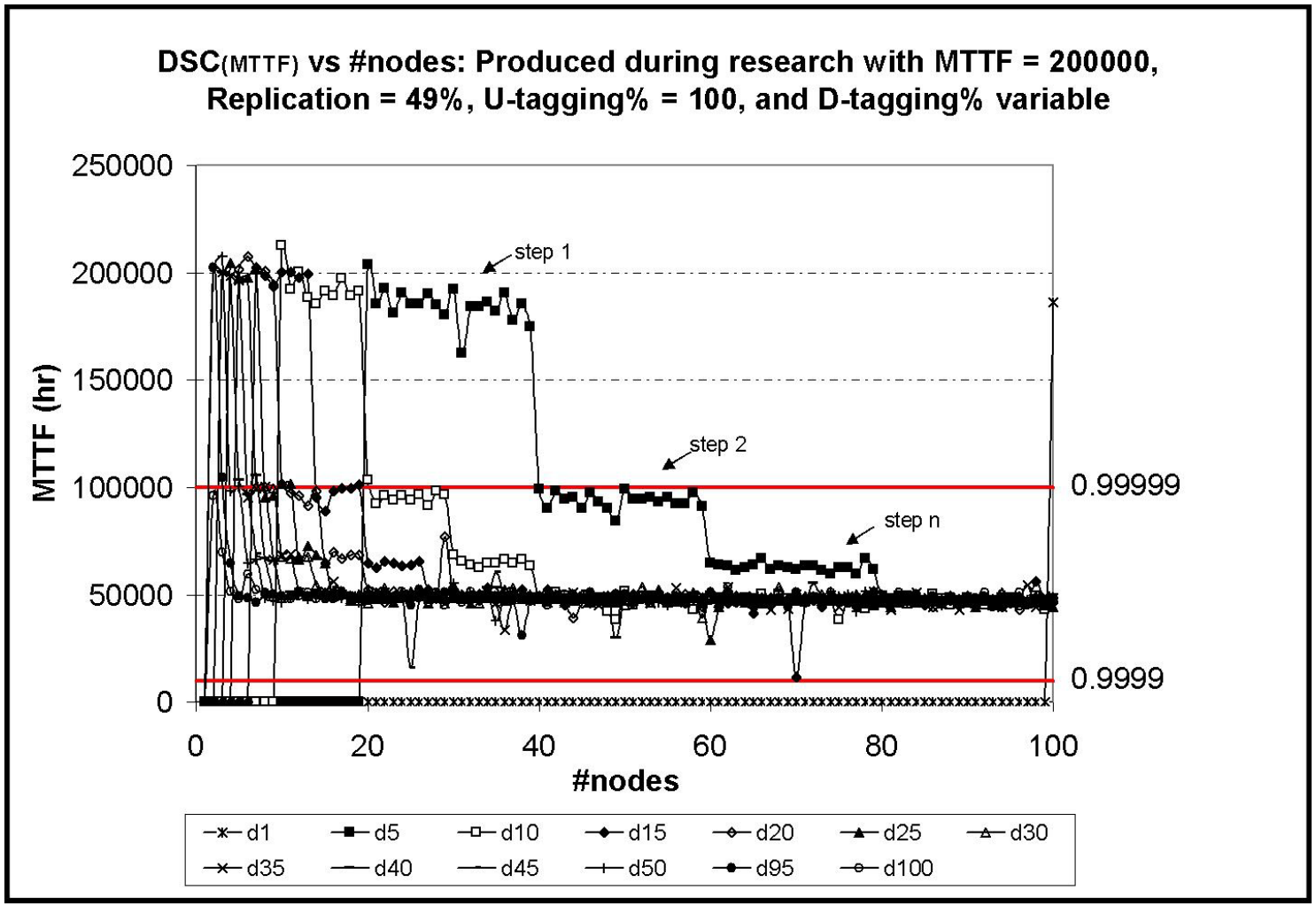
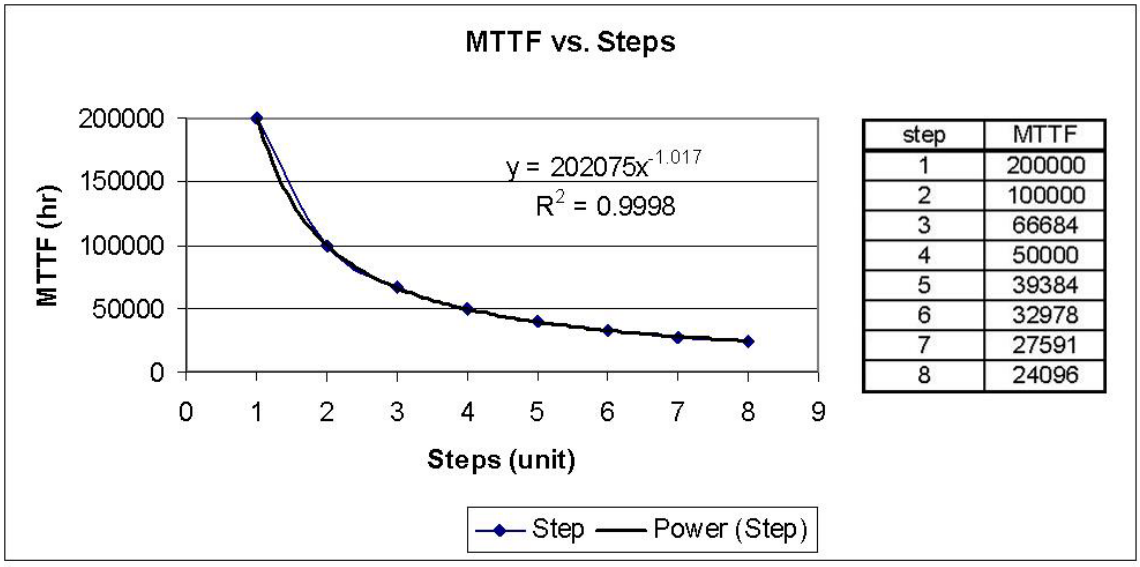


Figure IV-17. DSC curve family for 49% replication.

C. ERIS Parameter Estimation

Based on the curves in Figure IV-16 and Figure IV-17 we obtained the tagging percent in the lower region needed to get a particular MTTF for a DSC of a given node count. Note on Figure IV-16 that for each down region tagging percentage there is a step curve as the node count increases, and each step has the same level of availability or MTTF. Each level step fall a ΔSn difference that is related mathematically with the availability of the DSC, as the duration or length of the step is related to the "Nodes_allowed" formulas (8a) and (8b). These steps can be obtained from the function *x = (MTTF/a)^(1/b)*, where *x* is the step number, *a* and *b* are constants (in particular *a*=202075, and *b* = -1.017 for 200,000 hrs.), and *MTTF* is the desired availability threshold (DSC_{MTTF}). The constants are obtained from the curve fitting of the steps vs. the corresponding MTTF, as shown on Figure IV-18.



step	MTTF
1	200000
2	100000
3	66684
4	50000
5	39384
6	32978
7	27591
8	24096

Figure IV-18. Step vs. MTTF curve fitting, to obtain *a* and *b* constants.

Following is a word description for the ERIS algorithm; code included in Figure IV-19 and Figure IV-20. The relationship between the down tagging and the step number is given by the function *step = floor (node_count/(100/d))*, where *d* is the down tagging percent. By evaluating for *d*, we obtain the function *d = ceiling ((x * 100)/node_count)*. Since the *step* value can be repeated for different *d* and *node_count* values, a confirmation is needed. If *x* does not equal *step* then *d* has to be recalculated for a higher *step* where *x = (step - 2)*. If *x* is less than 1 then there is no *step* higher defined for the curves on Figure IV-16 and Figure IV-17, or for 1% to 49% replication; then we set *d* = 100 and the replication percent to 50%. If *x* equals *step* or *x* is not less than 1, then the *d* value is found but the replication percent still has to be calculated.

To calculate the replication percent, *pr*, we need the definition *C = round ((n_obj * pr)/(100 - pr))*, where *n_obj* is the count of original objects in the DSC, and *C* is the count of copies or redundant objects in the DSC, based on the replication percent or *pr*. Since objects cannot be fractioned, the calculation includes the rounding function. With the value of *C*, we defined the mathematical concept of general tagging without replication, *tag*. General tagging without replication means the percent of total nodes that have unique objects. Becomes *tag = (C + (node_count * d) MOD (n_obj - C))/node_count * 100*, where *MOD* is the modulus operation. Then to calculate the *pr* value, we set a desired tagging to a value to be compared to the calculation of *tag*. Later is done a loop of *pr* from zero (0) to fortynine (49), to stop on the first value of *pr* that gives a *tag* equal to the desire tagging,

remember that *tag* is related to *pr* through *C*. If the loop goes beyond 49, then *pr* is set to 50%, and *d* is reset to 100.

The previous algorithm obtains the parameters needed to sustain the desired DSC_{MTTF} threshold with a desired tagging, for 100 objects as the count of nodes increases. By integrating the algorithm into the simulator, replacing the static configuration parameters with the calculated parameters, the DSC is able to sustain the availability over the desired threshold.

The parameter estimation can be easily applied to other storage node MTTF values by applying a correction factor into the calculation of *d*, but not in the calculation of *pr* since this parameter is independent of the MTTF in the case where the threshold value in 9's is equal to the availability of one node . The correction factor is given by the relation $f = MTTF/MTTF_b$, where $MTTF_b$ is the base case value used to obtain the experimental values of the parameters (200,000 hrs). *MTTF* is the new availability of each node, and *f* is the correction factor. We then obtain *x = (MTTF/(a*f))^(1/b)*, where again, *x* is the step order, *a* and *b* are constants (in particular *a* = 202075, and *b* = -1.017 for 200,000 hrs), *f* is the correction factor, and *MTTF* is replaced by the threshold value.

```
//clrate - MTTF, n_disc - #nodes, goalnines - threshold in # of 9s,
//p_d - tagging down section
//Return -1 on error, 0 for out of range, 1 for normal calculation
int get_p_d(long clrate, int n_disc, int goalnines, int& p_d) {

  float base = 200000.0; //MTTF base
  long thres0;
  long thres1;

  long a = 202075; //step function values
  float b = -1.017;

  float factor;

  //x = (MTTF/(a*f))^1/b where f = MTTF/MTTF_b
  int x;
  //step = floor(n_disc/(100/d))
  int step;
  //d = ceiling((x * 100) / n_disc)
  int d;

  int d_if = 0;

  if (goalnines == 0) goalnines = nines(clrate);
  thres0 = threshold(goalnines,0);

  //  factor = float(thres0)/base;
  factor = float(clrate)/base;
  x = round( pow( (float(thres0)/(float(a)*factor)),(1.0/b) ) );
  d = (int)ceil(((float)(x * 100))/(float)n_disc);
  if ((d > 100) || (d == 0)) return -1; //Is not defined
  step = (int)floor(((float)n_disc)/(100.0/(float)d));
  if (x != step) d_if = -1;
  else p_d = d;

  if (d_if == -1) {
    thres1 = long(factor * a * pow(float(step-2),b));
    if (goalnines != nines(thres1) ) return 0;
      //There is no prior step
    else {
      x = step -2;
      d = (int)ceil(((float)(x * 100))/(float)n_disc);
      p_d = d;
    }
  }

  return 1; //Calculation completed
}
```

Figure IV-19. ERIS algorithm part I. Down tagging calculation.

```
void get_p_r(int n_disc, int n_obj, int p_d, int tagging, int&
p_r) {

  int C;
  int tag;
  int pr;
  bool found = false;
  pr = 0;
  do {
    C = round((n_obj * pr)/(100 - pr));
    tag = (C + (n_disc * p_d)%(n_obj - C))/n_disc * 100;
    if (tag >= tagging) found = true;
    else pr++;
  } while ((!found) && (pr <= 49));

  p_r = pr;
}

int nines(int mttf) {

double p;
double lambda;
int temp = 0;
int mul;

lambda = 1.0/double(mttf);
p = 1.0/double(exp(lambda));
mul = int(p*10);
p = p*10 - mul;

while (mul == 9) {
      temp++;
      mul = int(p*10);
      p = p*10 - mul;
}

return temp;
}

//level = 1 if threshold ~ n - 1,
//level = 0 if threshold ~ one node MTTF
long threshold(int nine, int level) {

int tn = nine - level;
long temp;
double p;

p = 1 - 1.0/double(pow(10,tn));
temp = (int long)(-1/log(p));

return temp;

}
```

Figure IV-20. ERIS algorithm part II. Replication, threshold, and number of nines calculation.

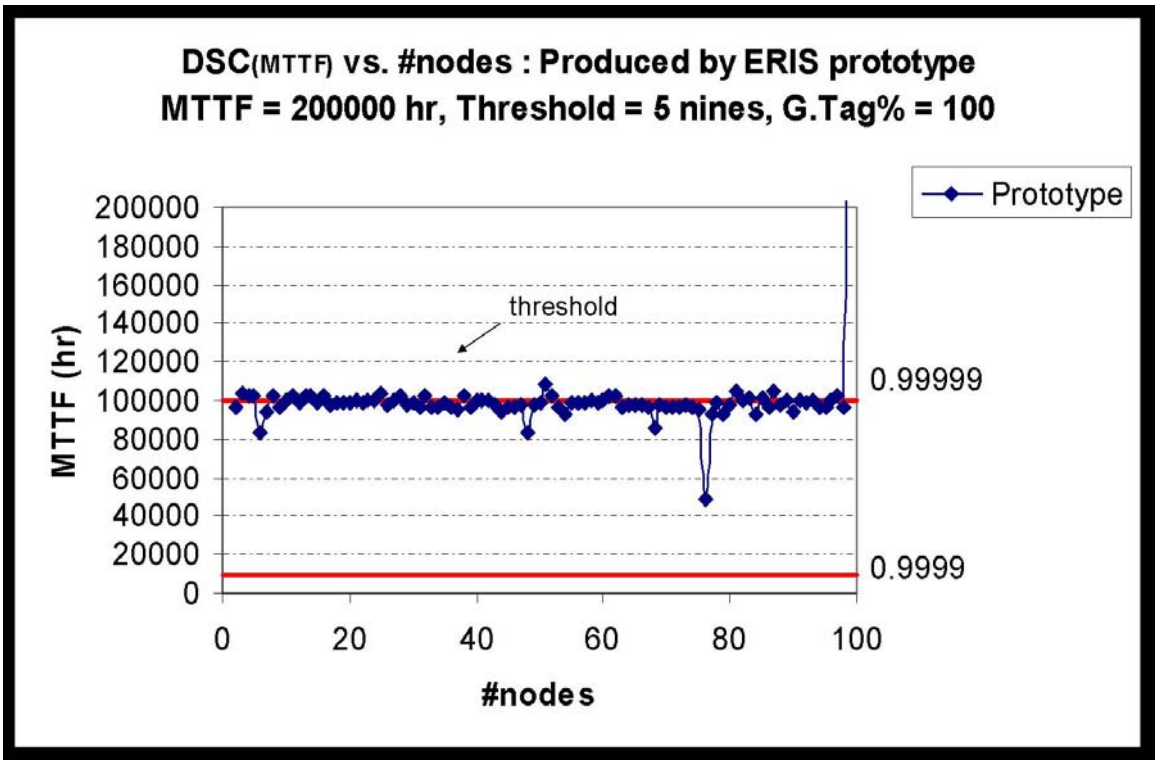


Figure IV-21. Availability result using the very first ERIS Prototype. The continuous line is the reliability for 5 and 4 nines.

Figure IV-21 illustrates how the availability is successfully sustained above a threshold as the node count increases. In the curve, around 78 nodes, appears a point value that goes below the threshold. This point is not significant due that it is a random phenomenon. We know that because the point changes its location when the simulation is run with a different seed to the random number generator used. Table IV-3 show the actual parameters of Down Tagging, Up Tagging, Replication, according to the count of nodes that produced the curve.

Table IV-3. ERIS calculated parameters.

Down Tagging%	Up Tagging%	Replication%	#nodes	#objects
100	100	1	2	100
67	100	1	3	100
50	100	2	4	100
40	100	2	5	100
34	100	1	6	100
29	100	2	7	100
25	100	3	8	100
23	100	1	9	100
20	100	4	10	100
19	100	1	11	100
17	100	3	12	100
16	100	2	13	100
15	100	2	14	100
14	100	2	15	100
13	100	3	16	100
12	100	5	17	100
12	100	1	18	100
11	100	4	19	100
10	100	7	20	100
10	100	4	21	100
10	100	1	22	100
9	100	6	23	100
9	100	3	24	100
8	100	9	25	100
8	100	6	26	100
8	100	4	27	100
8	100	2	28	100
7	100	9	29	100
7	100	7	30	100
7	100	5	31	100
7	100	3	32	100
7	100	1	33	100
6	100	10	34	100
6	100	9	35	100
6	100	7	36	100
6	100	5	37	100
6	100	4	38	100
6	100	2	39	100
5	100	13	40	100
5	100	11	41	100
5	100	10	42	100
5	100	10	43	100
5	100	8	44	100

Down Tagging%	Up Tagging%	Replication%	#nodes	#objects
5	100	7	45	100
5	100	6	46	100
5	100	4	47	100
5	100	3	48	100
5	100	2	49	100
4	100	15	50	100
4	100	14	51	100
4	100	14	52	100
4	100	13	53	100
4	100	12	54	100
4	100	11	55	100
4	100	10	56	100
4	100	10	57	100
4	100	9	58	100
4	100	8	59	100
4	100	7	60	100
4	100	6	61	100
4	100	5	62	100
4	100	4	63	100
4	100	3	64	100
4	100	2	65	100
4	100	1	66	100
3	100	19	67	100
3	100	19	68	100
3	100	18	69	100
3	100	17	70	100
3	100	17	71	100
3	100	16	72	100
3	100	16	73	100
3	100	16	74	100
3	100	15	75	100
3	100	14	76	100
3	100	14	77	100
3	100	14	78	100
3	100	13	79	100
3	100	13	80	100
3	100	12	81	100
3	100	11	82	100
3	100	11	83	100
3	100	10	84	100
3	100	10	85	100
3	100	10	86	100
3	100	9	87	100
3	100	8	88	100

Down Tagging%	Up Tagging%	Replication%	#nodes	#objects
3	100	8	89	100
3	100	7	90	100
3	100	6	91	100
3	100	6	92	100
3	100	5	93	100
3	100	4	94	100
3	100	4	95	100
3	100	3	96	100
3	100	2	97	100
3	100	21	98	100
3	100	50	99	100
2	100	50	100	100

Figure IV-22 shows the trends for the parameters calculated by the ERIS algorithm as well as the relative effective space (RES) offered by the algorithm. Since we are minimizing the replication, the RES resulted in a relatively small percentage of total raw space. However, this is the RES that can be achieved by keeping at most two copies of each object. Higher replication levels can be used to achieve even higher RES.

It is important to notice that the down tagging maximum decreases exponentially as the node count increases in order to sustain the availability. In fact, the down tagging has a much more significant effect than the replication level in the DSC's availability. If your application requires high availability, the down tagging cannot exceed a relatively small percentage. For instance, if you have a DSC with 20 nodes, it cannot exceed 10% down tagging or 2 nodes.

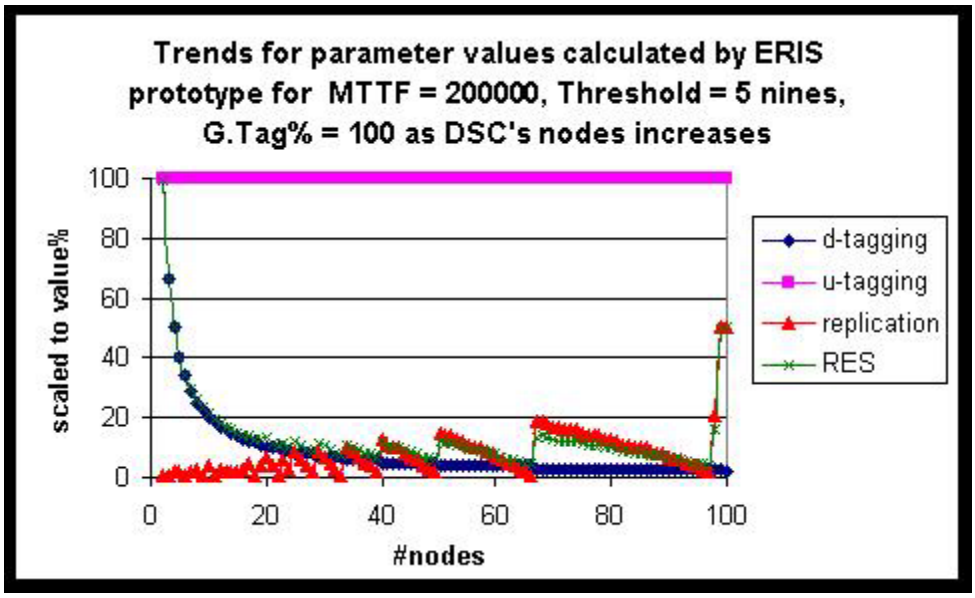


Figure IV-22. Comparison of trends for parameters calculated by ERIS.

Chapter V
Conclusions and Future Work

A. Conclusions

1. We have introduced a simple model of an abstract distributed storage system and used this model to uncover the mathematical relationship between storage system availability, replication and tagging under our simplified model.

2. Based on the results obtained during the validation phase of the research, illustrated with Figure IV-2 and shown by Table IV-2, that the simulation program yields valid results.

3. One important empirical observation is the fact that the availability of a DSC decreases rapidly as the number of nodes increases.

4. Another important finding is that the availability of a DSC system decreases as the number of nodes increases even using 50% of replication to within an order of magnitude of the availability of one node.

5. Our results evidence that some sort of elastic replication is needed to implement DSC's capable of sustaining a desired level of availability as the number of nodes in the DSC changes. A fixed level of replication appears no to suffice.

6. In our hybrid approach, the experiments demonstrate that the group of nodes with fewer replicas is the predominant point of failure of the DSC system (The chain breaks on the weakest link). In some terms, the upper and down regions could represent two different priorities. The upper region represents the high priority objects and the down region represents the low priority objects. You could have more than two priorities in the DSC by creating for example a region with three copies instead of two copies, and the traditional two regions. However, the DSC total availability will still be limited by the lower priority objects due that the DSC responds to the law of the chain breaks on the weakest link, and eventually those objects will be required.

7. The availability of a DSC system decreases as the down tagging increases, and increases as the down tagging decreases. Even thought it seems a contradiction because by increasing the tagging the objects are parallelized, the availability decreases because the system becomes vulnerable, since the more places the objects are distributed to the more opportunity there is for losing an object.

8. We have also proposed an algorithm for dynamically estimating the parameters of a DSC, necessary to re-configure the object allocation scheme that maintains the availability threshold with minimal storage overhead.

Summarizing, the assumptions made during this research are:

1. All objects are important to the survival of the DSC.
2. All nodes have the same failure rate or MTTF value.
3. On every loop of the simulation, we start with a fresh set of nodes.
4. All nodes are connected peer to peer, and in the event that any node gets disconnected, it is considered a decrease of one node in the DSC with no loss of any object.

If we eliminate the assumptions, conclusions 1 to 7 still hold. The conclusion 8 will hold if we eliminate assumption 2, with little modification on the algorithm.

B. Future Work

We have found an ERIS algorithm that can sustain the availability of a DSC while the node count increases, but since the replication is minimized, the relative effective space (RES) is reduced by the algorithm found. In order to increase the RES we need to find a better way to set the replication value to a higher value without affecting performance reductions caused by write events. Besides finding a better setting for the replication value to increase the relative effective space, there is a need to focus on the relaxation of some of the simplifying assumptions made so far. For example, there is a need to take in consideration the frequency of access for the objects. Normally in a database only the 20% of the objects is accessed the 80% of the time. The assumption that all objects have the same priority for the survival of the DSC and that each node has

the same individual MTTF value should be relaxed. In addition, the ERIS algorithm should have factor corrections for variable object counts, and simultaneously should be allowed to let the node count to go over the object count, which also might require a correction factor. Another interesting research avenue consists of analyzing the availability response of various SAN topologies.

We hope that all the lessons that we have learned during this work will be put to practice by designing a real storage system based on the principles of elastic replication.

Bibliography

[1] Alonso, R. et al. Data Caching Issues in an information retrieval system. ACM Transactions on Database Systems, 15(3), 359-384. Sep 1990

[2] Alsberg, P. A. and Day, J. D. A principle of resilient sharing of distributed resources. In Proceedings of the 2nd International Conference on Software Engineering, page-, 627-644, October 1976.

[3] Anderson, T. E., Culler, D. E. and Patterson, D. A. The case for now (networks of workstations). IEEE Micro Magazine, February 1995.

[4] Anderson, T. E., Dahlin, M. D. et al. Serverless network file systems. In Proceedings of the 15th ACM Symposium on Operating Systems Principles, pages 15 28, December 1993.

[5] Babaogölu, Ö. On the Reliability of Consensus-Based Fault-Tolerant Distributed Computing Systems. ACM Transactions on Computer Systems. 5(3): 394-416. Nov 1987

[6] Barak, A., Guday, S. and Wheeler, R. C... The MOSIX Distributed Operating System. Lecture Notes in Computer Science. Springer-Verlag, 1993.

[7] Barak, A. and La'adan, O. The mosix multicompuer operating system for high performance cluster computing. Journal of Future Generation Computer Systems. 13(45):361-372. March 1998.

[8] Bartlett, J. A non-stop kernel. In Proceedings of the 8th ACM Symposium on Operating Systems Principles, pages 22-29, December 1981.

[9] Bernstein, P. A., Hadzilacos, V. and Goodman, N... Concurrency Control and Recovery in. Database Systems. Addison-Wesley Publishing Company, Reading, Massachusetts, 1987.

[10] Birman, K. P. and Rennesse, R. V. Reliable Distributed Computing with the Isis Toolkit, chapter 5, pages 79-100. IEEE Computer Society Press, Los Alamitos, California, 1993.

[11] Chen, P.M. et al. RAID: High-Performance, Reliable Secondary Storage. ACM Computing Surveys, 26(2), 145-185. June 1994.

[12] Cheetah 73FC Product Manual,Rev.C
http://www.seagate.com/support/disc/manuals/fc/29482c.pdf

[13] Cole,G. Estimating Drive Reliability in DesktopComputers and Consumer Electronics Systems. 2000. http://www.seagate.com/docs/pdf/newsinfo/disc/drive_reliability.pdf

[14] Cooper, B.F. and Garcia-Molina, H. Peer-to-peer Resource Trading in a Reliable Distributed System. Electronic Proceedings for the 1st International Workshop on Peer-to-Peer Systems, Cambridge, USA. 2002.

[15] Douglis, F. and Ousterhout, J. K. Transparent process migration: Design alternatives and the sprite implementation. Software Practice and Experience, 21(8):757-85, August 1991.

[16] Drake, Alvin W. *Fundamentals of Applied Probability Theory* (New York: McGraw-Hill Inc., 1988).

[17] Eicken, T.V., Culler, D.E., Goldstein, S.C. and Schauser, K.E. Active messages: A mechanism for integrated communication and computation. In Proceedings of the 19th ACM International Symposium on Computer Architecture, Gold Coast, Australia, May 1992.

[18] El Abbadi, A. and Toueg, S. Availability in a portioned replicated database. In Proceedings of the fifth ACM Symposium on Principles of Database Systems, pages 240-251. 1986.

[19] Geist, A., Beguelin, A. Dongarra, J. et al. PVM: Parallel Virtual Machine - A User's Guide and Tutorial for Networked Parallel Computing. MIT Press, Cambridge, Massachusetts, 1998.

[20] Ghormlev, D.P., Petrou, D., Rodrigues, S. H., Vahdat, A. M. and Anderson, T. E.. Glunix: a global layer unix for a network of workstations. Software: Practice and Experience, 28(9):929961, July 1998.

[21] Gifford, D.K. Weighted voting for replicated data. In Proc. of the 7th Symposium in Operating Systems Principles, pages 150 162. ACM, December 1979.

[22] Gifford, D.K. Weighted Voting for Replicated Data. Proceedings of the Seventh Symposium on Operating Systems Principles. Pacific Grove, USA. 150-162. Dec 1997

[23] Hartman, J. H. and Ousterhout, J.K. The zebra striped network file system. ACM Transactions on Computer Systems, August 1995.

[24] Liskov, B., Ghemawat, S., Gruber, R. et al. Replication in the harp file system. In Proc. of the 10th Symp. on Operating System Principles, 1988.

[25] Litzkow M. and Solomon, M. Supporting checkpointing and process migration outside the unix kernel. In USENIX Association 1992 Winter Conference Proceedings, pages 283-290, December 1992.

[26] Mihram, G. Arthur. *Simulation: Statistical Foundations and Methodology* (New York: Academic Press, Inc., 1972).

[27] Mullender, S. P., Rossum, G.V, Tanenbaum, A.S., Renesse, R.V. and Staveren, H V. Amoeba: A distributed operating system for the 1990s. IEEE Computer Magazine, 23(5), 1990.

[28] Olston, C. and Widom, J. Best-effort Cache Synchronization with Source Cooperation. Proceedings of the ACM SIGMOD international conference on Management of data, Madison, USA. 73-84. 2002.

[29] Ousterhout, J.K., Cherenson, A.R., et al. Welch. The sprite network operating system. IEEE Computer Magazine, 21(2), 1988.

[30] Patterson, D.A., Gibson, G. and Katz, R.H. A Case for Redundant Arrays of Inexpensive Disks (RAID). Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, USA. 109-116. June 1988.

[31] Pfister, G.F. In Search of Clusters: The Ongoing Battle in Lowly Parallel Computing. Prentice-Hall PTR. New Jersey, 1998.

[32] Press, William H., Flannery, Brian P., Teukolsky, Saul A., Vetterling, William T. Numerical Recipes in C: The Art of Scientific Computing. Cambridge University Press; 2nd edition. October 30, 1992.

[33] Rosenblurn, M. and Ousterhout, J.K. The design and implementation of a log-structured file system. In Proceedings of the 13th ACM Symposium on Operating Systems Principles, October 1991.

[34] Siegel, A., Birman, K. and Marzullo, K. Deceit: A Flexible distributed file system. Technical Report TR89-1042, Cornell University, Department of Computer Science, 1989.

[35] Skeen, D. El Abbadi, A., and Cristian, F. An efficient fault-tolerant protocol for replicated data management. In Proceedings of the Fourth ACM Symposium, on Principles of Database Systems, pages 215-229, 1985.

[36] Stonebraker, M. et al. Mariposa: A Wide-Area Distributed Database System. VLDB Journal, 5(1), 48-63. Jan 1996

[37] Vogels, W., Dumitriu, D., Birman, K. et al. The design and architecture of the microsoft cluster service: A practical approach to high-availability and scalability. In Proceedings of the 28th symposium on Fault-tolerant Computing, Munich, Germany, June 1998.

Appendix

Email communication with “Numerical Receipts in C” people to obtain their consent to use their code for this thesis research.

Jose,

Thank you for your explanation. If I understand properly, you do not intend to distribute your software to other people, and you will not be printing the
Numerical Recipes routines within the thesis. In this case, it will not be necessary to have any permission from us for redistribution. Although it is normally

necessary to purchase either the software or the book in order to copy any of the Recipes to your computer, you may take this email as our express permission for the use of the Numerical Recipes routine "ran2" and its supporting function in your thesis work without any license fee.

We wish you luck with your research.

Regards,

William T. Vetterling
Numerical Recipes Software
vetterw@nr.com

==========

From: "Jose E. Torres" <jetorres@ece.uprm.edu>
To: Internet Mail::[vetterw@nr.com]; Internet Mail::["William T. Vetterling" <vetterw@nr.com>]

Subject: Re: Licensing question
Date: 4/10/03 4:57 PM

William:

Hello!

What I do with the "ran2" routine (along with the supporting functions) is to use it instead of the ANSI C "ran" function. The research is actually a Thesis. I need a large period random function because the simulator runs over extensive calls to a random function. The numbers produced by the function are used like failure/success values. With these numbers the simulator intends to model a RAID failure rate.

The code for the random function (and its supporting funtions) does not need to be printed in the Thesis document, because the purpose of the research is to create an algorithm to sustain the availability of a distributed storage system over topological changes, using the outcomes obtained during the simulator process.

The simulator code will be printed, and the intended algorithm. I can write a comment line with a reference to all your routine's book and page, to notify/direct the commitie/reader where it came the function in case they want to see how the random numbers were generated. The goal/final algorithm does not depend on the "ran2" function, although will be tested against, to verify if it works.

I want to inform again. I do not own the book, I used it from the University library.

Cordially;

Jose Torres

On Thu, 10 Apr 2003, William T. Vetterling wrote:

> Jose,
>
> Thanks for your email.
>
> Technically, the answer is yes. When you own a copy of the Numerical
> Recipes book, you have an "immediate license" (defined in the front of
> the
> book) that allows you to type the programs into a single computer and
> use them. However, this license does not allow you to distribute the
> programs to anyone else. Therefore you would need a different ftype
> of license in
> order to get permission for reproducing the software in your report, or
> for
> distributing the software as source code or executable code to other people.
>
> However, we often make exceptions to these rules for cases in which
> the use of the software is academic, and the distribution is very
> limited. For example, we routinely allow the use of up to 5 of the
> Numerical
Recipes
> in printed form in technical reports such as your paper. Also, we can
> give explicit permissions for the distribution of the code with
> limitations on the type of distribution and number of copies.
>
> Perhaps if you described to us exactly how you wish to use the

> software, we can suggest a way that would not involve any license
fee.
>
> Regards,
>
> William T. Vetterling
> Numerical Recipes Software
> vetterw@nr.com
>
> ==
>
> From: "Jose E. Torres" <jetorres@ece.uprm.edu>
> To: Internet Mail::[vetterw@nr.com]; Internet Mail::[nr@nr.com]
>
> Subject: Licensing question
> Date: 4/9/03 3:05 PM
>
> Hello!
>
> I am a computer science student doing research. In my research
> project I need a random generator to support a simulator program. I
> found on
my
> university library a copy of your book and write down the random
> routines on chapter 7 to use them on my simulator.
>
> I have been using ran2() for some time, and now I found this site
> saying that there is a license cost to use a machine readable
> software.
>
> I know that I have to write/notify on my report the use of your
> routines, but do I have to buy a license too?
>
> Cordially;
>
>
> Jose Torres
>
>
> PS: Please answer promptly.
>