12-2016

Substituting Failure Avoidance for Redundancy in Storage Fault Tolerance

Christopher David Brumgard
University of Tennessee, Knoxville, cseller1@utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Christopher David Brumgard entitled "Substituting Failure Avoidance for Redundancy in Storage Fault Tolerance." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Science.

Micah D. Beck, Major Professor

We have read this dissertation and recommend its acceptance:

James Plank, Jian Huang, Kenneth Stephenson

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Substituting Failure Avoidance for Redundancy in Storage Fault Tolerance

A Dissertation Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

Christopher David Brumgard
December 2016

DEDICATION

In loving memory of my father who encouraged and supported my love of science and who often sat up late into the night to help me with high school homework. And, for my patient and loving mother, whose nurturing nature and gentle nudging encouraged and cajoled me into finally getting this dissertation finished.

I do not know how I could have done any of this without the two of you.

I love you.

ACKNOWLEDGEMENTS

I would like to thank the following individuals and organizations for them supplying bandwidth usage information:

Internet2
NLR, National Lambda Rail
CENIC, The Corporation for Education Network Initiatives in California
FutureGrid
SoX, The Southern Crossroads, in particular, Cas D'Angelo
USLHCNet and Caltech Hep department,
LEARN, Lonestar Education and Research Network
Pacific Northwest GigaPop, Dale Tanigawa
Florida Lambda Rail, in particular, Keith Monroe, David Pokorney, Chris Griffin, and Veronica Sarjeant
Anthony D. Pisano (Caltech) and the Caltech IMSS - Voice & Data Networks Group

I would like to thank the following institutions for the use of their computing resources:

NICS, The National Institute for Computational Sciences
The University of Tennessee and their use of the Newton Cluster
Vanderbilt University and their use of the Vampire cluster

And of course my professor and advisor, Dr. Micah Beck, for all of the patience and understanding he has shown during my academic career.

ABSTRACT

The primary mechanism for overcoming faults in modern storage systems is to introduce redundancy in the form of replication and error correcting codes. The costs of such redundancy in hardware, system availability and overall complexity can be substantial, depending on the number and pattern of faults that are handled. This dissertation describes and analyzes, via simulation, a system that seeks to use disk failure avoidance to reduce the need for costly redundancy by using adaptive heuristics that anticipate such failures. While a number of predictive factors can be used, this research focuses on the three leading candidates of SMART errors, age and vintage. This approach can predict where near term disk failures are more likely to occur, enabling proactive movement/replication of at-risk data, thus maintaining data integrity and availability. This strategy can reduce costs due to redundant storage without compromising these important requirements.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

CHAPTER ONE
INTRODUCTION AND GENERAL INFORMATION

Introduction

The amount of data in the world is exponentially increasing every year and magnetic hard disk storage has become the preferred medium for preserving and accessing data [1], [2], [3]. For instance, according to the IDC, the amount of digital data has a compound annual growth rate of at least 57%. By 2020, the IDC projects that the amount of data to reach 35 zettabytes from the paltry 0.8 zettabytes that was present in 2009. Complicating this matter even further is that over a third of the data will need to be accessible via the "cloud." While the storage capacity of hard disks has been following Kyder's Law such that disk areal storage density increases 40% annually, the reliability factor and IO bandwidth of disks have not been keeping pace at less than 10% [4]. As the divide between capacity and IO continues to grow, it will become significantly more difficult to preserve and maintain data.

As the amount of data increases traditional techniques are not scaling well and are economically infeasible to work with [2], [3]. In the past, the data community has relied upon constant hardware improvements to keep pace with their data processing requirements, but we are entering a critical intersection of Kyder's and Moore's Laws. As Moore's Law comes to an end, the processing and IO to storage ratios will inevitability diminish. While we may yet have sufficient space to store the explosion of data, it is going to take a herculean effort to maintain and use the data effectively. Important data must constantly be managed and replicated to protect against loss while at the same time being available for processing by users across the globe. Both of these goals require significant processing power and IO bandwidth, and these are resources that cannot afford to be wasted.

This dissertation examines how adaptive data replication and placement heuristics based on hard drive health prediction can improve the resilience of data. By utilizing adaptive replication heuristics based on hard drive SMART errors and other factors, the research demonstrates that greater survivability of data can be achieved than with traditional multisite/multisystem replication techniques. Adaptive replication can also make better economic use of hard drives by anticipating when drive failures are likely to occur and can match perceived reliability of hard drive storage with the criticality of data. Currently, drives are replaced when they reach some certain predefined criteria (age, utilization, errors). Adaptive replication can predict which data needs to be migrated to other disks while using the less reliable disk to store either less

essential or more replicated data, making it possible to use hard drives until they have complete failure.

The initial results for the simple, no repair or recovery, policies demonstrate a clear need for active recovery and management. All of the failure models experience a significant data loss regardless of the initial replication settings. The SMART error failure suffered near total data loss of 99.51% to 99.94%, the disk age failure model lost 90.64% to 96.79% and the manufacturer, model and vintage failure model did better with just 76.73% - 91.55% data loss. With the exception of the 1-replica reactive policy, all of the failure models have far fewer data objects lost with reactive recovery. The 2 and 3-replica reactive policies are over 10 to 1,000 times more reliable for the SMART error age model and over 14 to 1,600 times more reliable for the SMART error count model. They are between 40 to 7,500 times better for the disk age model and 1,060 to nearly 3 million times for the manufacturer, model and vintage model.

Despite the tremendous reliability gains from the simple reactive mechanisms, the research conducted in this dissertation demonstrates that substantially further improvement is possible for the four different failure models by using adaptive, proactive techniques at various replication levels. For instance, in the case of the SMART error failure models, several of the 2-replica proactive configurations are 59 to 200 times more reliable than the 2-replica reactive policy and are up to 2 times more reliable than the 3-replica reactive while preserving 2.41 to 3 petabytes of storage over the 3-replica reactive policy.

The configurations for the 2-replica proactive policies for the disk age model range are between 63 to 118 times more reliable than the corresponding reactive policy. The 3-replica proactive is 1,300 times better than the 3-replica reactive. There are several 1-replica proactive policy configurations that are 58 to 156 times more reliable than the 2-replica reactive policy and those same proactive policies possess mean lifetime replica counts less than 2. There is even one 1-replica configuration that competes in reliability with the 3-replica reactive policy. A 2-replica policy configuration, requiring on average 2.14 replicas, is twice as reliable as the 3-replica reactive policy.

Finally, a couple of the proactive policies for the manufacturer, model and vintage demonstrate remarked improvement over the reactive policies. At the same replication level, the 1-replica, 0.99-policy outperforms the 1-replica reactive policy with the same storage utilization and the 2-replica, 0.9999 and 0.99999-reliability suffer only 0.76 and 0.126 times the loss rate of the 2-replica reactive policy. The 1-replica, 0.999 and 0.9999-reliability policy has better reliability than the 2-replica reactive, losing only 89 to 90% of the data objects as the 2-replica reactive. The 1-replica, 0.99999-reliability proactive configuration is competitive with the 3-replica reactive while requiring only an average of only 2.6 replicas.

These proactive policies tend to reduce the total data flow especially across the wide area versus simulations with the standard reactive policies. Being able to proactively replicate and migrate replicas on the same depot and in

the local area before failure occurs instead of having to pull a replica from another site after failure can result in drastic reductions. While maintaining high reliability, the 2-replica proactive policies for the SMART errors produce wide area traffic as low as 40% of the reactive policies. The manufacturer, model and vintage failure model has similar results for its proactive policies. For instance the 1-replica, 0.99999-reliability policy that is competitive to the 3-replica reactive generates just 4% of the wide area and 60% of local area traffic. The 2-replica 0.9999 and 0.99999-reliability policies generate 90% of the wide area and 80% and 60% of the local traffic, respectively, in comparison to 2-replica reactive policy. Sometimes, such as in the case of several of CFDR proactive policies, the proactive policies can create large quantities of bus-only traffic due to frequent migration as the disk reliabilities change over time, but the local and wide area data flow quantities remain either unchanged or reduced.

The dissertation begins with a review of the philosophy of Logistical Networking and the current state of its various components services before moving on to the literature review in Chapter Two. The literature review chapter provides justification for why hard drive and storage media failures matter in the grand scheme as well as a comprehensive review of the techniques currently employed addressing this problem in the wide area. The literature review ends with what is currently known about failure trends for hard drives that will serve as the basis of the failure models used for this study. Chapter 3 covers the materials and methods used to assess the feasibility and benefits of proactive, disk aware data migration policies over more traditional approaches. It examines the process of creating the LoDN simulator and the components required for emulation as well as a brief description of each of these individual components. The chapter then continues with a discussion of the disk failure models created for simulation based on the research from chapter 2 and the policies devised for them. The chapter also covers how the requisite information for network topologies, routing and competing data traffic was obtained. It ends with a discussion as to how the results are analyzed and compared with one another. Chapter 4 then describes the results found from the simulations and provides an analysis and a critique of their significance. Chapter 5 provides a recap of the research performed and the importance of the results before concluding with a set of recommendations for future work in this area. The dissertation ends with a set of appendices with comprehensive result tables.

Logistical Networking

Big data is a big problem! Lots of researchers across the globe need to work on data too big to be often moved, so data needs to be made available for use in both distributed environments and local clusters. However, combining big data with many distributed users and resources creates severe problems: adequate and accessible storage is rarely where it needs to be and going across networks from various institutions can pose severe limitations. When the data

gets big relative to network bandwidth and application timetable, its physicality can become problematic. Therefore, there is the need to stage data intelligently, at desired sites, according to policy and automatically on behalf of the users and resources. “Data logistics,” a term coined by Dr. Moore at the University of Tennessee, addresses these concerns by examining the time-related positioning of data resources and seeks to arrange resources so that data is where it needs to be when it needs to be.

Logistical Networking focuses on the problem of creating storage infrastructure scalable and flexible enough to meet the needs of data logistics. Logistical Networking is designed to scale-up in multiple dimensions: number of sites, sizes of datasets and number of users. It strives to be as open and as generic as possible with different layers of the software stack providing different services. Each Logistical Networking layer is as independent and separate as possible from one another with no “looking up” from the lower layers to the higher layers and all of the layers being directly accessible to the end user. Lack of generality and removing flexibility from users contributes to the lack of scalability in conventional storage infrastructures. For instance, while the higher layers virtualize storage resources from the users by allowing them to select storage based on their needs, the users are still free to reach down the stack and directly access the storage themselves.

Logistical Networking is designed to address the major data logistics issues in all data-intensive environments. It offers the ability to do various forms of caching, pre-staging and both push and pull models of data replication and distribution. It offers time-limited working storage at its base with higher layer services to maintain and distribute the data for as long as the users need. As the lowest layers of Logistical Networking operate at the block level instead of the file or dataset level, it makes complex multi-server and disk striping easy to manage and maintain with the individual blocks directly controllable. It also means that data transfers are by their very nature multi-streaming with redundancy.

The layers of Logistical Networking are described in the following sections with a general layout shown in Figure 1 adapted from [5].

The IBP Protocol

The Internet Backplane Protocol (IBP) is the fundamental core of logistical networking forming “the narrow waist” of the stack. In order to give application as much freedom as possible, it is designed to be a very generic and lightweight block level transfer and storage service. Middleware storage servers, called depots, are deployed as part of the network infrastructure. These depots allow endpoints to allocate temporary storage space. For each allocation, the allocating client is given access role keys to the allocation called capabilities that grant the client certain abilities over the allocation. These capabilities contain both the address information to identify the allocation and the rights to perform actions on the allocation. There are three types of capabilities for each allocation: a read capability to read the data on the allocation, a write capability to

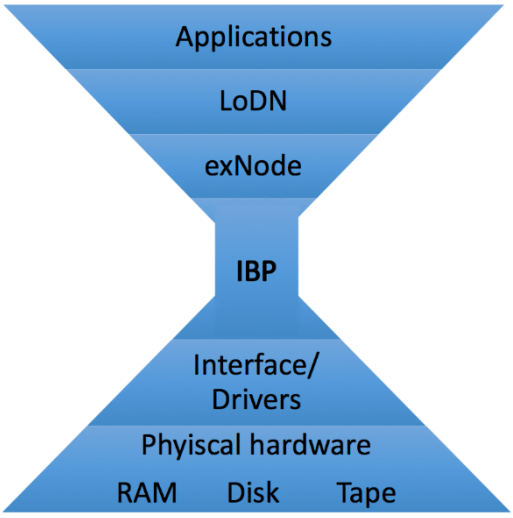


Figure 1. Logistical Networking Storage Stack.

write data on the allocation and a management capability to manipulate and retrieve the properties of the allocation.

Clients or other depots at the behest of a higher architectural layer can transfer data to and from depots. Data transfer is done per allocation/block basis and is “best effort” only. Additionally, the allocations are themselves a form of “best effort” storage where they are time limited meaning that at any time data can expire can be “lost” on a depot. Forms of reliability and service quality can be achieved via higher layers through techniques like replication and data encoding.

In many ways, logistical networking’s IBP is similar to network IP. Both serve as the common interface that architectural layers above and beneath can understand and adhere. These protocols are best-effort only services for data transfer relying on additional layers for stronger guarantees. Also, IP is a datagram with fragmentation protocol similar to how IBP deals with data blocks instead of higher concepts of files and data sets. Higher layers can use IBP and depots to store and forward data across networks in a unicast or multicast mode just like IP. In fact, IBP depots are very similar to IP routers in that allocations are treated as being put on a queue where data can be dropped if it has not been moved to the next destination in sufficient time.

The key difference between IP and IBP is a matter of the time scale involved. IP routers focus on the forwarding operation of data as opposed to storing it with data retention lifetime being in the milliseconds on the queue. IBP depots measure data lifetime in weeks and months by comparison with less emphasis on immediate forwarding. This behavior can enable clients to route and transfer data more reliability and avoid frequent end-to-end retransmissions of packets found in the current IP network.

ExNode

Since the IBP protocol has no notion of a file at the block level or any formalism for stating a relationship between different allocations, a higher service is required to represent files and complex datasets. Even on local systems, a file is an abstraction of data blocks. A file system is capable of presenting files to the user using a data structure called the inode. An inode (or in some cases a tier of inodes) can be used to provide a mapping between the physical blocks on disks and the logical file representation. Building on the inode concept, LoCI developed the exNode service and API. The exNode provides the information necessary to reconstruct a file from a set of allocations even where the file has been split up, stripped and replicated multiple times in the wide area on remote depots.

An exNode is a metadata container for mapping byte-level allocations to the logical address space of a file. Every exNode consists of a series of mappings with each mapping containing the IBP capabilities for that allocation, the physical offset and length within the allocation and the logical offset and length within the file. Each mapping can contain end-to-end conditioning for that allocation including encryption and checksumming. Furthermore, the exNode service provides the ability to extend every mapping and exNode with additional information about the allocations and the file it represents. This extensibility has allowed the exNode to meet a wide range of user and application needs including various data protection schemes like Reed-Soloman encoding of files.

Furthermore, the exNode has several advantages over inodes and related techniques that are important in dealing with wide area data. Unlike the inode, the exNode directly exposes the structure of the file and allocations to the user. Traditionally file systems have hidden this information from the user, but by exposing this information, it allows the user to optimize better for his needs. For instance, the exNode reveals the locality of the data needed by the user. He uses this information so that data is retrieved from relatively nearby, high availability or high bandwidth depots. By looking at this information, the end user can use different download and upload algorithms that best meet his needs. Since the exNode contains the IBP allocation information, it allows users the ability to bypass the file level abstraction and work directly with the blocks themselves.

LoDN

With the ability to represent files and datasets, an important problem arose with the management of the ExNodes. Simply storing them on a local file system made it difficult to for the user to keep track of them, share them with collaborators and create understandable relationships between exNodes. Additionally, manually renewing, replacing and controlling a collection of exNodes is a very tedious task and can be daunting to do reliability and efficiently enough to meet the needs of users. This lead to this author's primary work for the last several years developing and researching a Logistical Distribution/Data

Network for users and applications to work not only with the data but the metadata associated with IBP.

LoDN, the Logistical Distribution/Data Network, is in part a policy and exNode metadata repository that offers a hierarchical directory service to the user. A LoDN user can manage a directory tree and reference exNodes that have been imported. Through user's view of LoDN, each exNode appears to be the file itself removing the need for the user to understand and work directly with the metadata much like how the interface of a POSIX file systems obscures the underlying details of storage. Through one of the several LoDN interfaces and API's, a user has a standard POSIX-like control of directory service. Directories and "files" can be created, renamed, moved and removed at will.

LoDN also has an authentication and authorization system for user verification and access permissions. Directories can be set to being user accessible only or traversable and readable by others. Users are prevented from creating or accessing metadata to and from other users' areas. ExNode files at any time can be set to being only user accessible (private) or publishable to the world. When published to the world, LoDN can select whether to make the data read-only or writeable by others via manipulation on the exNode. The exNode information given out is stripped of the IBP capabilities for writing and managing the data. Without these capabilities users are prevented from changing, overwriting or removing the data, but can still read the data being disseminated by the source owner. These abilities make LoDN an excellent tool for project collaboration.

Data Dispatcher

An essential part of LoDN are the data dispatcher services which are responsible for distributing and maintaining multiple file replicas at sites as specified by the user's policy. The data dispatcher service is in charge of content distribution, data recovery and providing reliability through redundancy. It runs both periodically, on a time interval tied to the duration of storage allocations, and on demand for instance, when a new exNode is imported to LoDN or a policy has been updated. The data replication and migration component is a continual background process that does not interfere with data access by users and attempts to optimize the transfer of data between depots. The data dispatcher renews the time-limited storage allocations and also checks the availability of allocations, making repairs if needed. Repairs are not triggered by transient non-availability only when multiple faults or long-term unavailability has been detected.

The process of data replication and movement is controlled and driven by user-specified policy. The data dispatcher operates on exNodes based on the "nearest" hierarchical policy: per exNode, directory and account. The data objects are replicated to user defined "sites" with only replicas at those locations maintained. Users can upload a file locally and then use the data dispatcher to

migrate the data to sites across the globe, near collaborators effortlessly based on policy.

REDDnet

The largest deployment of Logistical Networking is REDDnet. It is an NSF funded infrastructure project designed to provide a large distributed storage facility for data-intensive collaboration with the mission of providing working storage to help manage the logistics of moving and staging large amounts of data in the wide area network. It currently consists of 9 sites: California Institute of Technology, European Organization for Nuclear Research, Stephen F. Austin State University, Texas Advanced Computing Center, the University of California at Santa Barbara, University of Florida, University of Michigan, University of Tennessee and Vanderbilt University. All of the sites have either 1 or 10 gigabyte per second links to the national network backbones including Internet2 and the now defunct NLR. Figure 2 adapted from [6] provides a graphical representation of REDDnet as it existed in 2009.

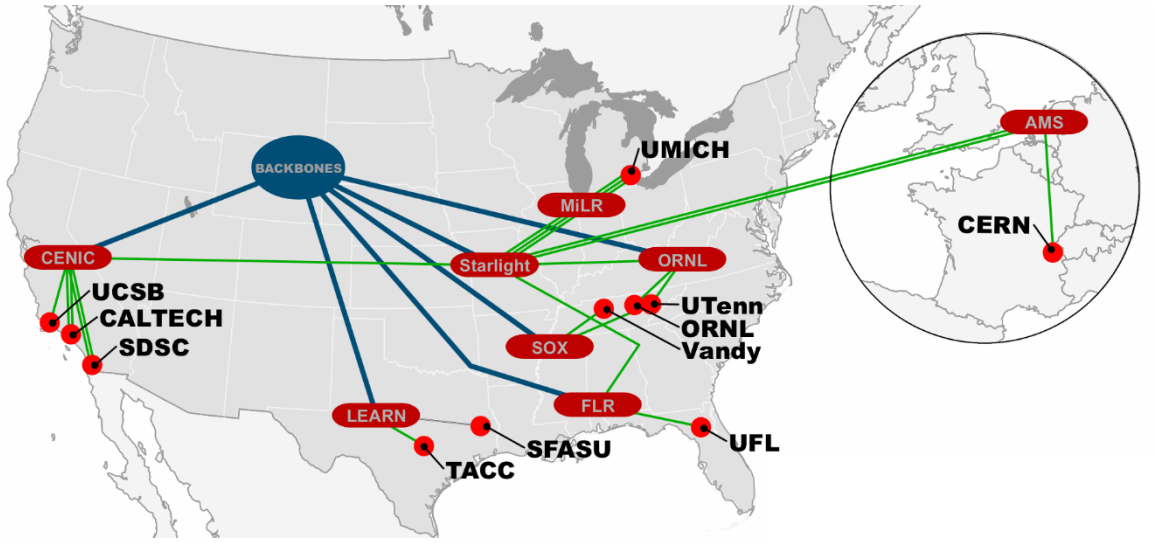


Figure 2. REDDnet Deployment Map.

CHAPTER TWO
LITERATURE REVIEW

Before starting the actual research, an extensive review covering the state of knowledge about disk failures and other storage mediums was conducted. It began by determining the importance of hard drive based storage in relation to other available mediums and the extent of the impact that hard drive failures had in the world. After the significance of hard drive failure had been verified, an examination was conducted on what had already been accomplished to address the problem and in particular if any work had been done towards predicting disk reliability. With an eye towards wide area storage systems, it was ascertained that almost all systems still rely solely on reactive repair and having as many replicas as possible to safeguard data. Perceiving a potentially crucial gap in an avenue of research, this work collected all of the available information about drive failure in the literature. To that end, every publication that could be accessed for the last several decades was reviewed with the best sources having been published in the past ten years. Three primary sources were analyzed that spanned a gamut of metrics for disk failure: SMART Errors, Age and Vintage. This chapter provides an overview of the findings of the research performed.

Significance of Hard Drive Failure

The natural question arises “why study hard disk failure for Logistical Networking?” According to the IDC, the amount of digital data is growing exponentially at a compound annual growth rate of 57% while available storage is only growing at 35% a year [2]. In 2007, the amount of information being generated began to exceed the total amount of storage available and continued to outpace available storage. Just four years later, it was anticipated that half of the digital data would not have a permanent home [2]. This analysis by the IDC is illustrated in Figure 3 taken directly from their work. While most of the data, some of which is also replicated, being created does not need to be permanently stored, the growing divide between available storage and data produced does mean we need to use storage as efficiently as possible in Logistical Networking.

Other storage like tape or solid state devices, SSD’s, have not yet been significantly used for Logistical Networking. Since the advent and rise of hard drives, magnetic storage tape has mainly been relegated to archival storage due to issues with speed and latency induced by its lack of random access capabilities. Flash based SSD’s are still new and have not been around long enough for the long-term study of their failure behavior. Until recently with falling costs, they have not been used heavily for large storage systems except as a new cache layer or metadata store. As shown in Figure 4 from their paper, the

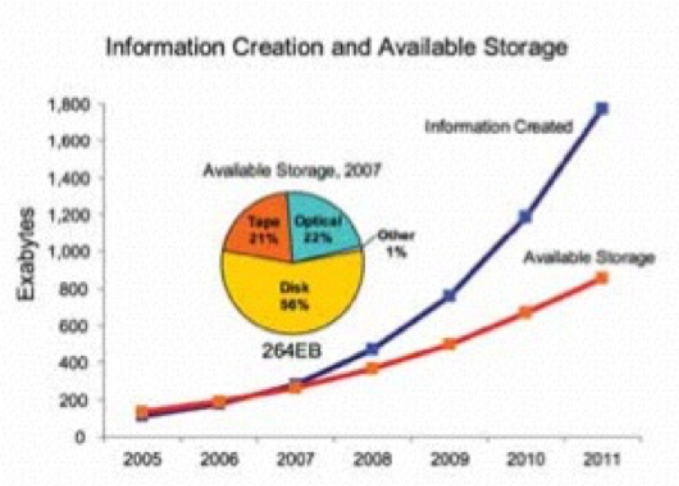


Figure 3. Data Explosion.

IDC continues to expect that hard drive based storage will continue to make up over half of the available storage when compared to the alternatives of optical, tape and SSD storage [2].

Others have also reached the similar conclusion that disk drives will continue to be predominate for the foreseeable future. According to Kryder's law, named after Dr. Mark Kryder of Seagate, the areal density of hard drives is expected to increase at a rate of 40% a year [7]. His projections show that by 2020, a two platter 2.5 in hard drive will be capable of holding 40 terabytes at a cost 3 dollars. No other storage technology will be able to supplant hard drives in storage per dollar in the near term [8].

Dr. Kryder reviewed a number of different storage mediums that are under development and their viability for replacing hard drives by 2020. He found that the most significant challenge to hard drives is NAND based SSDs. While SSDs have been making a push into the storage market in the last few years, they are still ten times more expensive per gigabyte and are approaching the limits of lithographic manufacturing technology with innate physical limits beyond the 22nm process for them. SSD's should hit this limit in the near future stalling further capacity growth while magnetic storage is still decades away from its fundamental limit of 100 terabits per square inch. While SSD's are useful for mobile devices and as an additional layer in the storage hierarchy, hard drives will continue to be the workhorse of storage including for Logistical Networking and other storage systems. Other more futuristic technologies (Ferroelectric RAM, Magnetic RAM, Spin Transfer Torque RAM, Phase Change RAM, Carbon Nanotube RAM, Probe Memory, Holographic Memory, Copper Bridge RAM, Resistive RAM, Racetrack RAM, Single Electron Memory, Molecular Memory and Polymer Memory) are very unlikely will be able to supplant hard drives due to technical challenges, cost per unit of storage, reliability and access speed.

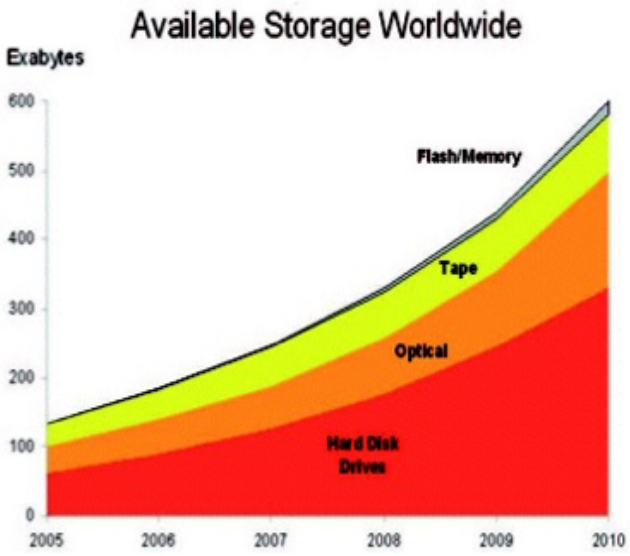


Figure 4. Available Storage Worldwide.

Another poignant question is “why not study other system components?” In fact, a recent paper [9] raises that very criticism of only studying disk failure in storage subsystems stating that other components are also responsible for storage system failures and disks are not always even the dominant factor. A gentle rebut, however, would be that only disk failures can cause permanent data loss; other system failures only affect data availability transiently. Even the authors of the paper, themselves, concluded that 20-55% of overall storage subsystem failure is due to disk failures indicating that while not the sole cause, disk failure is very significant. [10] argue for the importance of understanding disk drive failures. Disk drives are one of the few components that possess mechanically moving parts, and as such, they are among the least reliable of components accounting for up to 50% of all hardware replacements according to the Computer Failure Data Repository. A more recent study [11] concluded that 78% of all server hardware replacements were hard drives. Hard drives are and will continue to be a critical element for storage servers and techniques that exploit their failure characteristics will be of paramount need.

Review of Current Redundancy Methods

Currently, resiliency techniques in Logistical Networking have been implemented using either simple replication, erasure coding or distributed RAID with single and double redundancy. All of these techniques share the same fundamental problem. The growth in device storage capacities are outpacing their reliability and IO transfer rates; in fact, some storage mechanisms are

becoming inherently less reliable as their capacity increases [12]. In terms of hard drives, according to Kyder's Law [7] capacity is expected to grow at 40% percent per year while IO transfer rates are typically growing less than 10% a year. Based on data from [4], there is a calculated growth rate of only 7.5% per year for transfer rates from 2001 to 2009. As a result of these growing gaps between capacity, reliability, and IO, when a storage device fails, it takes increasing longer to rebuild from the failure. The data being vulnerable for increasingly extended periods of time, in turn, escalates the likelihood of a subsequent failure occurring in the interim that can cause permanent data loss. The resiliency techniques have responded by increasing the level of redundancy with more replicas and coding blocks.

In Logistical Networking, essential data is already being replicated 3 to 4 times or coded for at least the same level of redundancy. As the gap continues to widen, it is likely that Logistical Networking will continue escalating to even higher levels of redundancy. For simple replication schemes, this is problematic due to storage, bandwidth and management requirements of maintaining more replicas. Distributed RAID and other forms of erasure encoding can ease the storage requirements, but they are very processor and IO intensive when computing and recovering blocks. For instance, RAID 5 will easily quadruple network load when computing the parity blocks and when forced to execute a recovery operation. The drastic rise in processor and IO can slow down and sometimes even prevent access to other data. Furthermore, disk failure exhibits both strong correlation and bursty patterns. Schemes like RAID that assume independent failures do not tolerate correlated, bursty failures [9]. Finally, in Logistical Networking both of these techniques only allow users to specify static levels of redundancy that does not dynamically adapt to the overall changing storage environment.

Wide Area Storage Systems

Other wide area infrastructures were reviewed to see how they handle redundancy and storage failure. Most of them are similar to LoDN and Logistical Networking relying on either basic naïve replication or erasure coding [13], [14], [15], [16], [17]. A brief review of the modes of operation and techniques for several of the most notable infrastructures are described below.

The following wide area storage system use simple naïve replication.

LOCKSS

LOCKSS [18], Lots of Copies Keep Stuff Safe, is a long-term digital preservation system for journals and other academic works that is based on the notion that "Lots of Copies Keep Stuff Safe." It is designed to model the methodology used by librarians for physical documents by creating as many copies as possible and distributing them across the world. It creates as many replicas as there are participating peers with one replica one peer. The peers in

LOCKSS periodically poll the other trusted peers to verify that their local replica is present and valid. When data loss is detected, LOCKSS replaces it at the peer by pulling from another peer across the wide area. The infrequent polling interval means that a significant amount of time may expire before the loss of a replica is detected. Therefore, LOCKSS is very dependent on having highly replicated datasets to avoid potential loss.

PAST

PAST [19] is a peer-to-peer archival and content distribution that replicates a static, fixed number of primary replicas with the ability to create temporary replicas based on user access demands. It attempts to roughly balance the number of files on every node by randomly dispersing the files and all replicas based on the proximity of a unique id assigned to each file and the id of the storage nodes using the Pastry protocol [20]. Replica placement is mostly static only changing when nodes fail, come online or the storage capacity of the node is near exhausted. As nodes come online or reach exhaustion, PAST performs rebalancing of files and nodes. When a node fails, the replicas are replaced by putting a new set on the next closest node as determined by its identifier that does not already have a replica.

CFS

CFS [21], Cooperative File System, is another peer-to-peer file system that is read-only. It replicates data based on a fixed replication factor and is reactive in nature to data loss. When replica loss is detected, it replaces the replica from across the wide area on a new host.

The following infrastructures use some form of distributed raid and erasure encoding.

OceanStore

OceanStore [22] is perhaps the most similar to LoDN in form and function. It utilizes the Tapestry [23] peer-to-peer overlay network with location awareness for managing their nodes and replicas. Its replica management is directed at staging temporary replicas based on service requests and relieving the load on central nodes, not at handling or preparing for data loss. OceanStore maintains an inner ring that stores a complete copy of a data object plus a set of erasure encoding blocks for recovery. Any other complete replicas are “soft” temporary replicas that serve to speed up access by the client and are often on the client machine itself. The erasure coding is based on Reed-Solomon and Tornado codes with the level of erasure encoding being determined on a per object basis. A process routinely and passively sweeps through objects finding damaged objects to repair and rebuild. OceanStore attempts to rank separate administrative domains by reliability and attempts to avoid locations that have high correlated failure probabilities.

POTSHARD

POTSHARD [24], Protection Over Time, Securely Harboring And Reliably Distributing Stuff, aims at providing long-term, secure archival storage. It uses erasure encoding and "secret sharing" to protect data objects from data loss and for security. Using secret sharing, it creates "shards" of data objects that distributed among a set of independent and distributed archives. The form of erasure encoding is either Reed-Soloman or a simple parity function.

PASIS

PASIS [25] was an early distributed storage architecture that uses a p-m-n erasure encoding scheme like POTSHARD. It passively monitors for data loss and will perform recovery when needed.

Tahoe-LAFS

Tahoe-LAFS [26] is a more recent cloud storage infrastructure that is security and privacy-focused. It provides reactive repair of data based on erasure encoding.

Besides just the form of redundancy used, several of the wide area storage systems have some more advanced resiliency features.

Ceph and CRUSH

Ceph [27] is a distributed POSIX-like file system overlay of the CRUSH, Controlled Replication Under Scalable Hashing, algorithm [28]. CRUSH is pseudo-random data distribution algorithm that distributes replicas across heterogeneous, structured storage cluster. Using a deterministic function, it maps a data object or group to a list of devices on which to store object replicas. Redundancy is supported via replication, RAID parity schemes, erasure encoding and hybrid approaches. Similar to LoDN's current policy mechanism, CRUSH defines placement rules for each replication strategy and distribution policy. As such the authors mention that CRUSH could allow the end-user to handle potential sources of correlated device failures with the caveat that information must be encoded into the cluster map used by the policy. Doing so would make it possible for the placement policy to disperse the replicas across different failure domains. This mechanism is not dynamic, though, and the policies are static.

Glacier

Glacier [29] uses extremely, highly replicated redundancy to combat Byzantine level of correlated node failures. Instead of predicting potentially correlated failures or using reliability estimates, Glacier takes the view that it is impossible to understand every potential interaction of hardware and software that could lead to a large-scale correlated failure event. It uses any available storage in the system to create more replicas and coding blocks. When a data

object arrives, Glacier will replicate a small set of full replicas (2-3) and maintain them for fast access. After a prescribed time limit, Glacier generates redundancy via Cauchy Reed-Soloman erasure encoding and distributes the blocks across a collection of nodes. There are periodic sweeps of the data objects, replacing lost redundancy blocks and generating further redundancy using low levels of bandwidth.

Tempo

Tempo [30] is a distributed hash table that seeks to reduce bandwidth bursts after failure by continuously replicating data objects in the background using continuous low level of bandwidth. Instead of reacting to failure that can generate substantial network loads and bursts when performing repairs and replacements, Tempo produces more replicas in the background over time before failures occur. Tempo is somewhat unique, in that there is not even monitoring of storage failures. It just sweeps over data objects continuing to increase the replica count of a data object until some maximum value is reached.

TotalRecall

TotalRecall [31] is a policy based, flexible distribution system. It varies the type of redundancy typically favoring replication for small files and erasure coding for large files. It also will vary the speed with which it will repair lost data. Similar to what this dissertation's reach aims to do, TotalRecall attempts to lower bandwidth and storage usage by monitoring the short and long-term behavior of nodes adjusting the redundancy level of files when certain thresholds are reached. TotalRecall relies on the overall behavior of the storage nodes to judge if a node departure is transient or permanent. The only experiments done would appear to be done in simulation with a "perfect" node predictor to determine the effects on bandwidth.

Protector

Protector [32] is another peer-to-peer distributed storage system. It has the standard reactive repair mechanisms but seeks to improve the replication efficiency by using a probabilistic algorithm to determine if a node failure is transient or permanent. The algorithm uses conditional probability, based on the current downtime of a node, to determine if the node is permanently down. For transient failures, Protector has the option of not replicating more copies. This ability serves to reduce the number of extra replicas needed overcome transient failures and avoid over replication.

FarSite

FarSite [33] is a serverless file system designed to run on distributed, untrusted computers with the intention of providing both data reliability and security. The replica placement of data objects is initially randomized; however, over time based on availability FarSite will attempt to equalize data availability

among data objects by shuffling locations of higher availability files with those of lower availability.

Phoenix

The Phoenix [34], [35] recovery system is a distributed, cooperative backup system between users. It attempts to lessen the amount of replication needed for “Internet Catastrophes” by placing replicas using a technique called “Informed” replication. Based on node diversity; where the diversity is based on various attributes of participating client nodes such as their OS, web server, Internet browser; a data object is replicated across a cross section of these attributes to reduce the risk of any particular attack vector successfully removing all data availability. In essence, like some other techniques, it is aimed at preventing correlated failures.

In summary most wide area storage systems are merely reactive in nature to failure using erasure coding or replication to achieve a level of redundancy, hopefully, high enough for repair and recovery to take place before enough failures can accumulate to cause permanent data loss. There are a few other approaches that build on this. Glacier and Tempo take the opposite approach as to what this dissertation wants to achieve. They continually perform replication increasing the number of copies per data object to avoid data loss. Several like OceanStore and Ceph have the capacity to allow end users to combat correlated failure events by avoiding placing replicas on similar hardware. This ability is not their main goal nor have there been results found to demonstrate their effectiveness. TotalRecall and Protector try to reduce the amount of redundancy needed by attempting to predict the behavior of nodes to see if more reactive repair is necessary, but they do not seem designed to proactively replicate data in anticipation or likelihood of near-term failure. Phoenix with its “Informed” replication is similar to what this research aims to achieve, but it only uses high level, static attributes of systems to avoid correlated failures. It does not to migrate data based on a dynamically changing environment. Based on the extensive review done, it appears that there are no wide area storage systems that attempt to improve reliability while adjusting individual data object redundancy by dynamically migrating and replicating based on changing disk health metrics.

Disk Failure Prediction

For the last two decades, considerable effort has been spent researching and attempting to predict hard drive failures. Much of this attention has been focused on using algorithms that take into account various SMART error subsets. Since 1995, most of the hard drive manufacturers have been implementing SMART, Self-Monitoring, Analysis and Reporting Technology, errors into nearly all hard drives as firmware in hard drive controllers. The SMART error interface within the controller monitors drives for certain events and errors keeping track of

them with a set of counters [36]. These counters are accessible through various software utility packages. In the original approach, once certain SMART error counters exceeded a specified maximum threshold, a failure warning would be triggered to alert the administrators to replace the drive. The objective was to be able to provide up to 24 hours of advanced warning before drive failure. Unfortunately, this approach has been shown to possess a high FAR, false positive rate, and a detection rate of only 3 to 10% [37], [38]. Additionally, many of the drives that do fail have had no SMART errors beforehand [39].

Since then there have been numerous attempts at refining the predictive use of SMART errors using a broad range of machine learning algorithms trained against disk populations from a few hundred to over 200,000 disks. As early as 2001, there were approaches with Naïve Bayes classifiers and expectation-maximization trained submodels [40]. The classifier method was able to obtain a 56% detection rate with a lead of 24 hours at a FAR of 0.82% on a disk population of 1,936 disks. The submodels did not fare as well, achieving a detection rate of just 30% with a FAR of 0.2% for the same warning interval. The next attempt was made by [37] and [38] in 2002 and 2003 using Rank Sum Tests. Training against two disk populations of 369 and 3,744 disks, a 40-60% detection rate was obtained with a lower false alarm rate than was seen in the original approach. Utilizing Maximum Likelihood Rules, Agrawal, et al. [41] was able to construct a ruleset of 100 rules that could predict 66% of failures with a 3% FAR using a disk population of 28,887 disks from NetApp. Several variations of SVM, Support Vector Machines, have been tried starting in 2010 with [42]. Fitting a spectrum-kernel to an SVM, the authors analyzed and classified sequences of disk events from the syslog files for a cluster. With a lead time of 24 hours, their SVM could predict disk failure 80% of the time. In 2013, Zhu et al. [43] was also able to get an 80% prediction rate using another SVM trained on 23,395 hard drives. They were able to average 330 to 360 hours of advanced warning preceding the disk failure at a FAR of just 0.3%. The same authors also claimed to get a 100% detection rate with a 2.26% FAR and an average of 356 hours warning using a Backpropagation Neural Network. Using Hidden Markov Models and Hidden Semi-Markov Models, a 46 to 52% detection rate with a 0% FAR was achieved on disk population of 277 disks by [44] in 2010. A 95.49% prediction rate at 24 hours before failure was achieved using Classification and Regression Trees on a population of 25,792 disks with just a 0.09% FAR [45]. The largest disk population analyzed for prediction was 220,022 disks by Yang et al. in 2015 [46]. They implemented a predictor called Hdoctor, based on an L1 Linear Logistic Regression Model, that after being trained on the dataset could predict 97.82% of the failures with 0.3% FAR. Table 1 provides a summary of the results for the disk failure prediction schemes.

In 2013, an extensive survey of 21 different machine learning algorithms was conducted by [47] on a dataset consisting of 369 drives. The algorithms used covered a broad set of learning algorithm categories: Probabilistic Models using Naïve Bayes classifier, Multinomial Naïve Bayes classifier and Bayesian

Table 1. Summary of disk failure prediction schemes.

Approach	Detection rate	FAR	Look ahead (hours)	Pop size	Year
Naïve Bayes classifier	56%	0.82%	24	1,936	2001
Naïve Bayes submodels trained using expectation-maximization	30%	0.2%	24	1,936	2001
Rank Sum Tests	40-60%	0.5%		369 and 3,744	2002, 2003
Maximum Likelihood Rules	66%	3%		28,877	2009
Support Vector Machines	80%		24	1,000's	2010
HMM and HSMM	46-52%	0%		277	2010
Support Vector Machines	80%	0.3%	330-360	23,395	2013
Backpropagation neural network	100%	2.26%	356	23,395	2013
Classification and Regression Trees	95.49%	0.09%	24	25,792	2014
Hdoctor - L1 linear logistic regression model	97.82%	0.3%		220,022	2015

network; Decision Trees using C4.5, REPTree and Random Forest; Rule-based using ZeroR, OneR, Decision Table, RIPPER and PART; Hyperplane Separation using Support Vector Machine, Sequential Minimal Optimization and Stochastic Gradient Descent; Function Approximation using Simple Logistic Regression, Logistic regression, Multilayer Perceptron and Voted Perceptron and Instance-based Learning using Nearest Neighbor Classifier, K-Star and Locally Weighted Learning. For the methods employed, prediction rates varied from 0 to 97.4% with a 0 to 7.8% FAR. They ended by concluding that there was no clear winning approach and that choosing which algorithm to employ would be application specific based on the strength of prediction, training time and prediction time.

In addition to SMART errors, there has been work done on other factors believed to affect hard drive reliability. For instance, a theoretical framework based on the clearance between the magnetic heads and the disk platters as well as environmental factors such temperature and humidity was created by [48] in 2007. There has also been work done to predict high-risk disk groups by manufacturers before drives are shipped to customers using Neural Networks with Rank-Level Fusion [49] and Decision Tree Learning [50].

While this work into failure prediction could potential be very useful, the research presented in this dissertation takes a different approach. Many of these discussed prediction schemes are extremely costly computationally and are binary classifiers meaning that they can only produce will-fail/no-fail outcomes without providing further potentially useful information. Instead of predicting failure and replacing disks ahead of supposed failure, this work focuses on ranking the health of disks and data objects. This research will proactively adjust the replication of data objects to maintain high reliability accordingly. These disks will be capable of continuing to be used by data objects that have higher reliabilities and for other purposes. This technique has the advantage of having

0% FAR since disks are only replaced post failure and ensuring that disks are not wasted. The disks that survive their periods of higher risk can continue to be utilized effectively. As far as could be found, none of the prediction schemes are yet used in any storage systems. Most of the prediction performance metrics are based on replaying the dataset or a subset thereof after training. Perhaps the closest predictor to this dissertation's research was in [45]. In that work, the authors created a health degree model based on Regression Trees in addition to a binary classifier. The model can produce a health assessment for the near-term ordering of disk replacements and reconstructions and fine tune disk failure detection thresholds. This approach still calls for early disk replacement and only controls the ordering of replacements instead of more long-term health projections.

Review of Known Disk Failure Trends

For disk failure trends, the literature over the past decade and a half was reviewed to attempt to understand what important factors have been uncovered. Despite the importance of hard disk drives, there has been surprisingly very little large scale work done analyzing their failure trends. For as long as hard drives have been used, the storage community does not have as good an understanding of their reliability characteristics as one might expect. Data about hard drive reliability characteristics mostly comes from two sources: disk manufacturers and end users.

Disk manufacturers' numbers are not very useful for prediction as much of the analysis is based on accelerated aging and stress experiments, limited short-term database results and small test studies as outlined in [51]. They primarily use MTBF, mean time between failure, and MTTF, mean time to failure [52]. MTBF and MTTF are flawed in that they are the reciprocal of the failure rate and thus assumes that failure rate is constant over time producing an exponential distribution of failure. Numerous studies, [10] ,[39], [52], [53], [54], have shown that failure rate for hard drives is not constant over time and that the exponential distribution is a poor fit for modeling their failure. Even if true an exponential distribution based on age would be useless since failure is constant over time. Other hard drive data is difficult to obtain from vendors and manufacturers as they do not want to disclose proprietary information that could be used against them by competitors [10].

Additionally, failures are seen differently by manufacturers and users. The failure rate perceived by end users is much higher than what manufacturers observe, sometimes by as much as 30 times [10], [52]. 15-60% of the drives returned to the manufacturer for testing reveal no problems according to the manufacturer's testing regime [52]. Similar results were found by another manufacturer in which 43% of the returned hard drives likewise exhibited no errors [10]. Much of this could be due to differences in workloads and performance expectations between end-users and manufacturers. Also, hard

drives rarely function in a fail-stop method and have much more complicated failure behaviors. As the end users' perception is what Logistical Networking will experience, it is this perspective that will be considered for this research.

It has only been in recent years that meaningful data and trends have begun to arise from large populations sets. Large-scale user studies are rare because of the number of hard drives that would be required combined with the time and ability to analyze the data. Users often worry about disclosing data that could be used due to non-disclosure agreements with vendors [10]. Despite this, there have been three studies published by NetApp, Google and the CFDR, The Computer Failure Data Repository. The failure trends used in this research was drawn from the results obtained from these datasets.

Age

The first disk metric with a believed correlation to failure was drive age. It has long been suspected to be a primary culprit in disk failure modeling with several different distributions envisioned to describe the failure behavior. Already discussed were the MTTF and exponential distribution which was shown to be rather fruitless for this research. Two other possibilities are the bathtub curve distribution, commonly used in age models for various components, and empirical measurements from sources like the Computer Failure Data Repository.

Bathtub Curve

A commonly believed age based failure distribution for hard drives is the bathtub model, so named because the curve resembles the cross section of a bathtub as depicted in Figure 5. The bathtub model consists of three distinct failure rate phases: infant mortality, useful life and wear out. The first phase, infant mortality, is characterized by a decreasing failure rate and is usually the result of imperfections during manufacture. Useful life begins when the failure

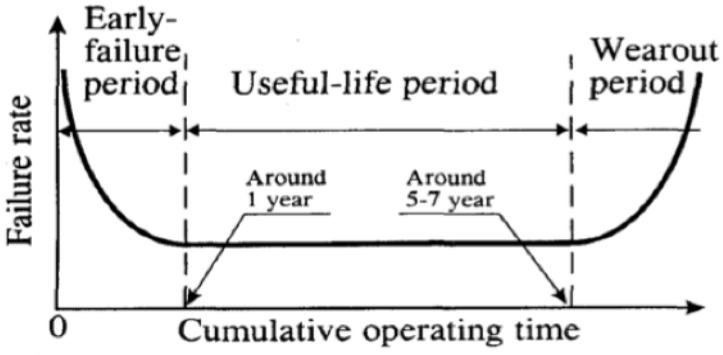


Figure 5. Traditional Bathtub Curve.

rate steadies out and becomes constant. This stage is typically the main lifetime of the hard drive. In the final phase, components undergo wear out, hence its name, and the failure rate rises. The infant mortality phase is typically assumed to end after the first year, followed by the useful life until the wear out begins sometime between the fifth to seventh year [51]. As discussed in the subsequent sections, the accuracy of the bathtub curve, while widely believed, has become disputed in recent years by a number of hard drive studies.

CFDR

Because raw, empirical data for reliability studies of hardware components is so difficult to obtain, Bianca Schroeder and Garth Gibson of Carnegie Mellon University and USENIX established the Computer Failure Data Repository, CFDR, to aide in public research on system reliability [10]. The CFDR features failure logs and data from Los Alamos National Laboratory for nine years of operation. Schroeder and Gibson used this information in [10] to analyze failure trends in hard drives.

As shown in Figure 6, the failure rates are significantly higher than what would be expected from manufacturers' MTTF specifications. In the early years, failure was six times greater than excepted and 7 to 10 times higher in years 4 and 5. There was no indication of a strong infant mortality problem or the standard bathtub failure curve. Instead, the replacement rate of drives tended to increase over time indicating that wear-out begins sooner than expected and is more dominate of an influence.

According to their work, disk replacement seems to have a degree of correlation and that as time without a disk replacement increases, the lower the hazard rate for the other drives. The time between replacements is best modeled on by Weibull curve with a shape distribution of 0.7 to 0.8 in Figure 7.

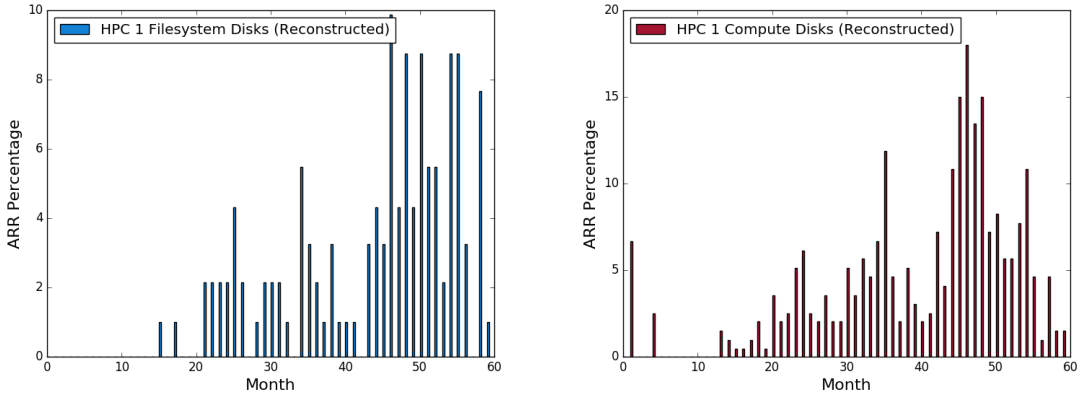


Figure 6. Average replacement rates for the CFDR HPC 1 nodes. On the left are the reconstructed values for the filesystem nodes and on right are the values for the compute nodes.

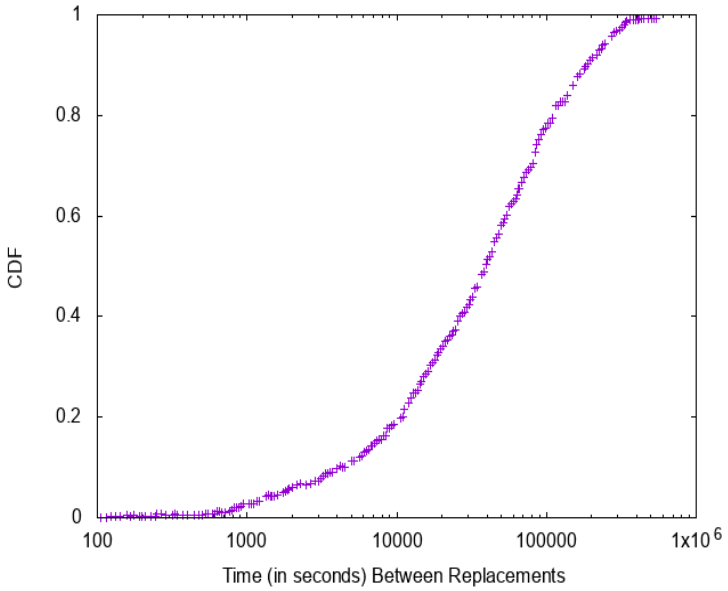


Figure 7. CFDR Distribution of Time Between Replacements and Hazards (Reconstructed)

There was also some interesting information in [39] about the annualized failure rate of various disk ages but the authors mostly discounted it as being the result of the use of different manufacturers and models. However, the reconstructed data for their values is in Figure 8.

Manufacturer, Model and Vintage

Another long-held belief was the impact of a disk's manufacturer, model and vintage on the disk's reliability. Dr. Jon Elerath and Sandeep Shah of NetApp have studied the role of these factors in disk reliability for years and published numerous papers on the subject [52], [53], [54], [55], [56]. Network Appliance, NetApp, is one of the largest computer storage and data management companies in the world. It uses hard drives from five of the world's leading manufacturers of disk drives and maintains long term records on these drives. Elerath and Shah have used this database to analyze failure trends from this disk population and concluded that the disk manufacturer, family, model and vintage have a significant impact on disk reliability.

A disk family consists of all of the disks in a generation of disk technology that varies only by the number of platters and read/write heads. Within a disk family, each model will be identical in the number of platters, read/write heads and capacity. A vintage is a particular period of disk production for a model. For each family and model, the later vintages tend to be more reliable as production techniques mature as shown in Figure 9. Within a family, the number of read/write heads and platters has a positive correlation with failure rate. As seen

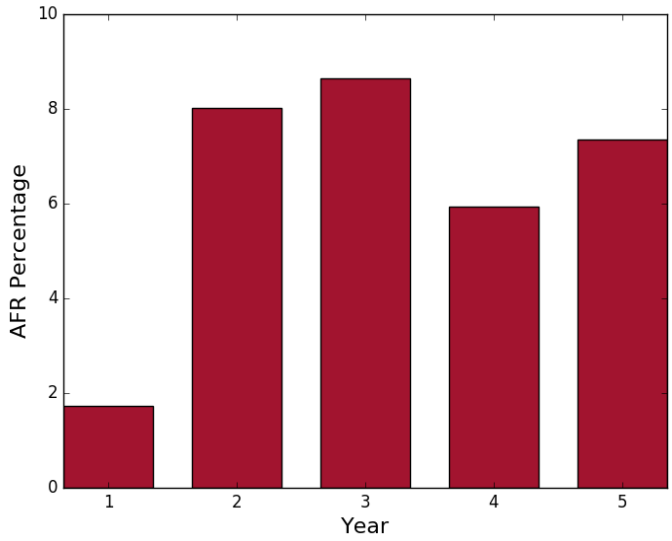


Figure 8. Google AFR by Age. (Reconstructed).

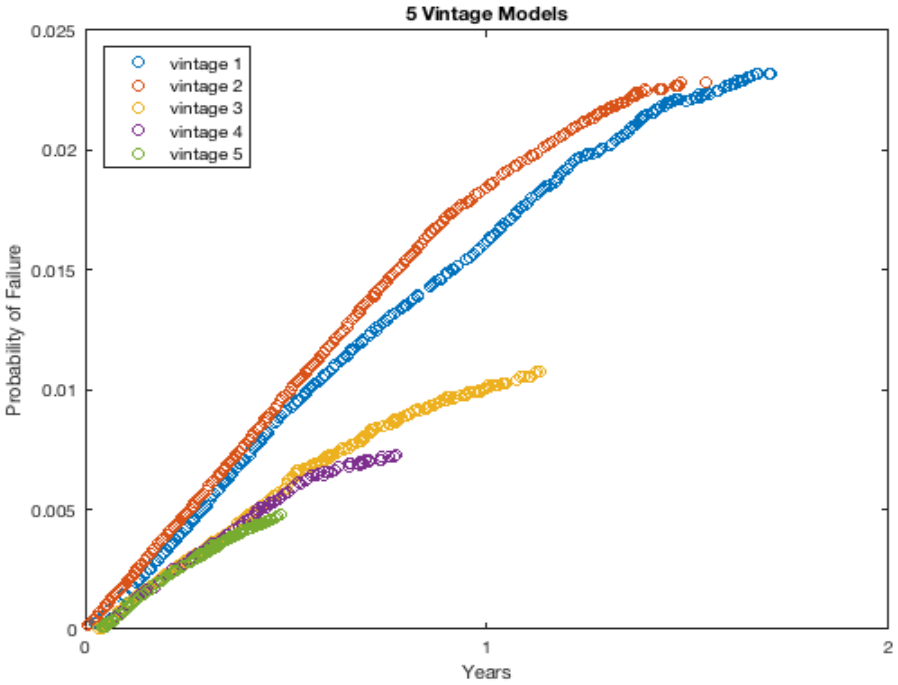


Figure 9. Elerath Vintage Failure Rate vs. Vintage. (Reconstructed)

in Figure 10, adapted from [55], for one disk family where disks represented by the blue and green lines have twice and four times as many heads, respectively, as the disks indicated red line, the relationship between the number of heads and failure is evident. While other families examined did not show as a distinct trend, Elerath and Shah believe that consideration of disk capacity within a family should be considered for gauging reliability. [39] also backs up the notion of the impact of manufacturer, model and vintage but does not provide the relevant information.

SMART Errors

In 2007, the largest disk failure study was published by Pinheiro et al. in [39] based on Google's experience with their disk populations. This study was a groundbreaking study for its scope and analysis of SMART error factors. Since its publication, it has been referred to many times in subsequent works. The authors scrutinized five years' worth of logs for a disk population consisting of over one hundred thousand hard drives with both serial and parallel ATA drives. The population was a heterogeneous collection of drives from several different manufacturers and models ranging in capacity from 80 to 400 GB. This study was also significant because the authors explicitly based the failure mode on the end user's perspective, i.e., when a disk behaves poorly enough for an end user's use case, then it must be replaced.

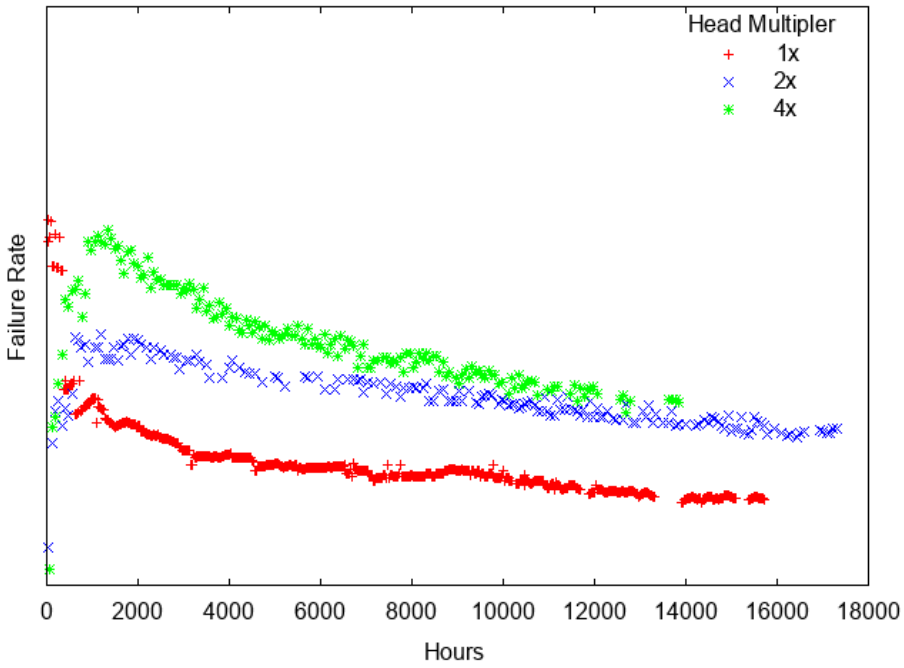


Figure 10. Hazard Rate vs. Number of R/W Heads. (Reconstructed)

From their experience, they overruled the conventional wisdom that temperature and usage rate were significant contributors, and thus predictors, to disk failure. They found little correlation between failure rates and temperature and usage level. In Figure 11 taken from their work, it is only in the first and last year that utilization would appear to be indicative of near-term failure and even then it may have more to do with the manufacturers and models then the utilization rates.

While the authors discounted many potential disk metric predictors, they found that 44% of the failed drives had particular kinds of SMART errors within the last 8 to 9 months of their life. Filtering through the dozens of different kinds of SMART errors, four types were found to be extremely relevant to disk failure: scan errors, reallocation counts, offline reallocation and probational counts. The next several subsections have an overview of their findings.

Scan Errors

Scan errors occur in the background when the disks detect errors in the sectors as the drive heads pass over and read them when the drive is typically in an idle state. The scan error value stored on the drive is the count of the number of sectors on the drive that have experienced this phenomenon. Scan errors were witnessed in 2% of the overall disk population during the timeframe, and those disks that did suffer from them had an average ten times greater AFR, annualized failure rate, than non-affected disks. 30% of the disks with at least one scan error failed sometime in the subsequent 8-month timeframe. Dividing the drives by age groups revealed that while young drives were more likely to fail

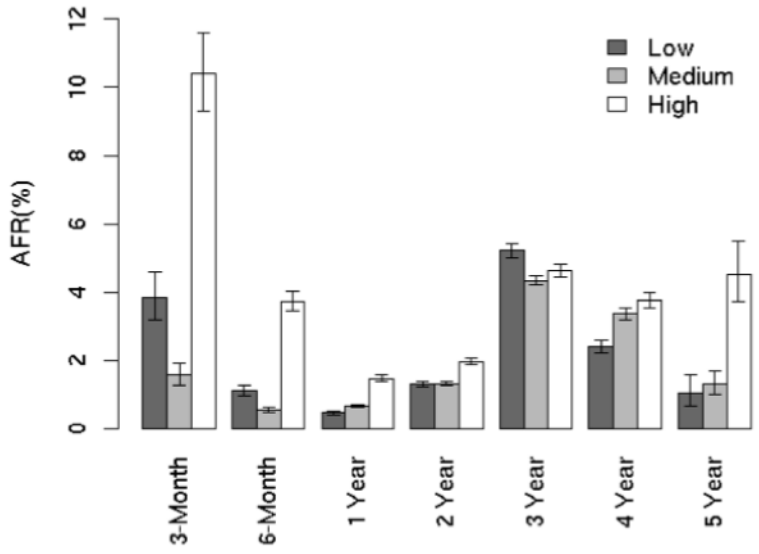


Figure 11. Utilization Rate vs. Failure.

within the first month of an event, the older the drive was, the higher the cumulative failure rate. Additionally, those drives with more than one scan error were more like to fail than those with just one.

Figure 12 and Figure 13 provide a breakdown of the data reconstructed from the paper and used by this research. The process of reconstructing the data is discussed in the methodology chapter later. Figure 12 has the AFR of disks with scan errors versus disks partitioned by the age of disks when the error occurred. The graph on the left of Figure 13 has the failure CDF of these disks grouped into four age brackets of 0-8, 8-15,15-25 and 25-44 months. The graph on the right of the figure gives the failure CDF for disks that suffered either only one or multiple scan errors.

Reallocation Errors

The next set of SMART errors under consideration were reallocation errors. Typically, these SMART errors occur when there is an error during a disk sector write event, and the drive can remap the sector to a spare sector if it considers the original sector damaged. The drive maintains a counter for the number of times this has occurred. Over nine months, 9% of the disk population experienced at least one of these errors with those same disks having 3 to 6 times greater AFR than the non-affected disks. On average 15% of these disks failed within the next eight months from the onset of the first reallocation.

Figure 14 provides the reconstructed yearly AFR for the disk population with and without reallocation errors over the nine-month observation window. Those disks with reallocations fail significantly more than those without any

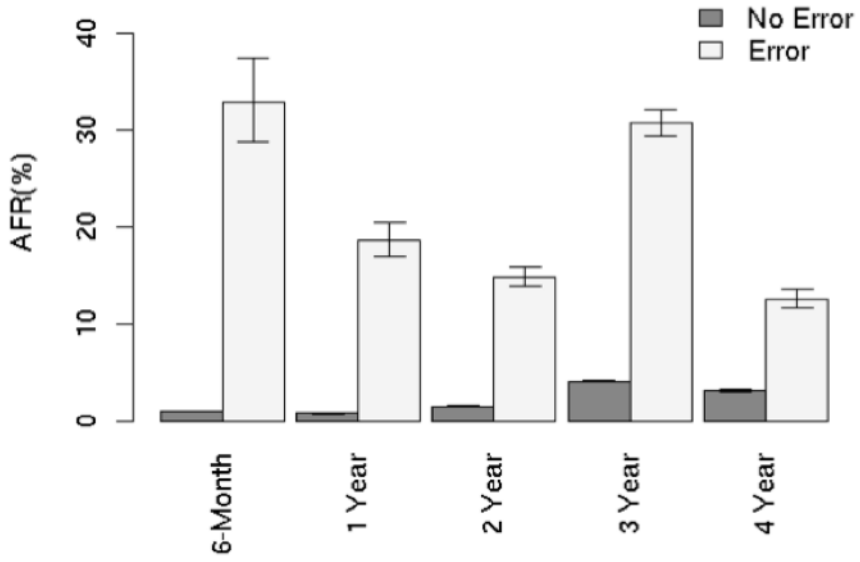


Figure 12. Scan Error AFR.

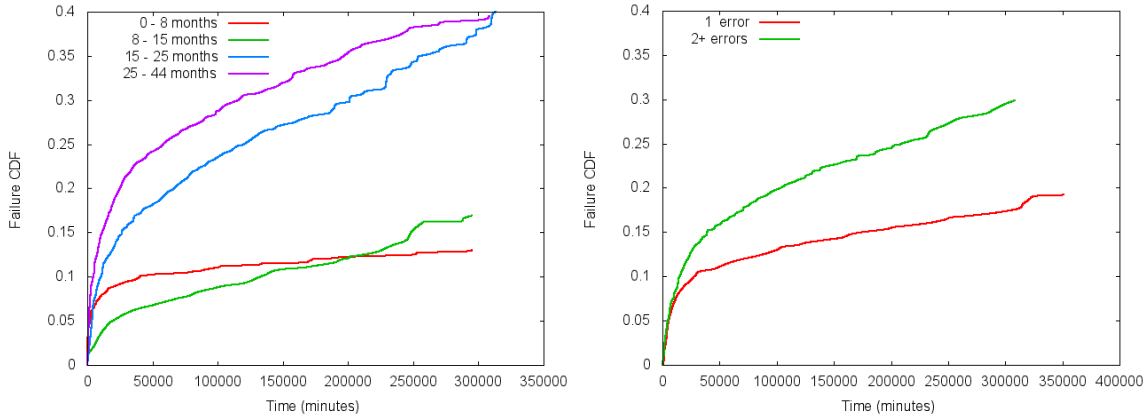


Figure 13. Scan Errors Failure CDF Graphs. The graph on the left provides the CDF of disk failure with respect to time dividing the disk population by age of first scan error occurrence. The graph on the right provides the CDF of disk failure with respect to time by the number of scan errors seen on the disk before failure. (Reconstructed)

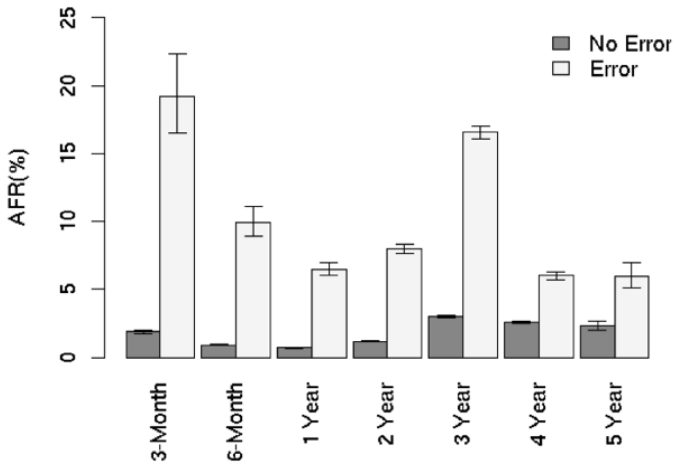


Figure 14. Reallocation Error AFR.

errors. The discrepancy in failure rate between disks with and without errors is higher, the younger the disk. The left graph in Figure 15 has the failure CDF for the afflicted disks separated into four distinct age groups: 0-5, 5-10, 10-20 and 20-60 month. With the exception of the 5 to 10-month disk group, as the age of disks increased so did the failure rate for them. While the failure CDFs for the 0-5 and 5-10 month disks quickly flattened, the two older groups continued to rise over the eight months from the onset of the first error. The graph on the right provides the CDF's of disk failures for those that experienced 3 or fewer reallocation errors and those disks with more.

Offline Reallocation Errors

A significant subgroup of the general reallocation errors is offline reallocation errors. Unlike the previous SMART error, these errors occur in the background when a disk is performing routine sector scrubbing maintenance. When possible the disk will remap the problematic sector to one of the available spares and increment the counters for both the offline and regular reallocation errors. Offline reallocation errors were observed in 4% of the population during the nine-month period.

The reconstructed data for yearly AFR and the failure CDF are shown in Figure 16 and Figure 17. The first figure compares the AFR's for the unaffected disks with affected drives. The second figure has the CDF failure curves partitioned by various disk ages: 0-9, 9-17, 17-25 and 25-56 months; as well as, the CDF curves for disks with only one error versus those with multiple errors. The failure rates associated with offline reallocation errors are significantly higher than the general reallocation errors. The older the disks, the more fatal the error

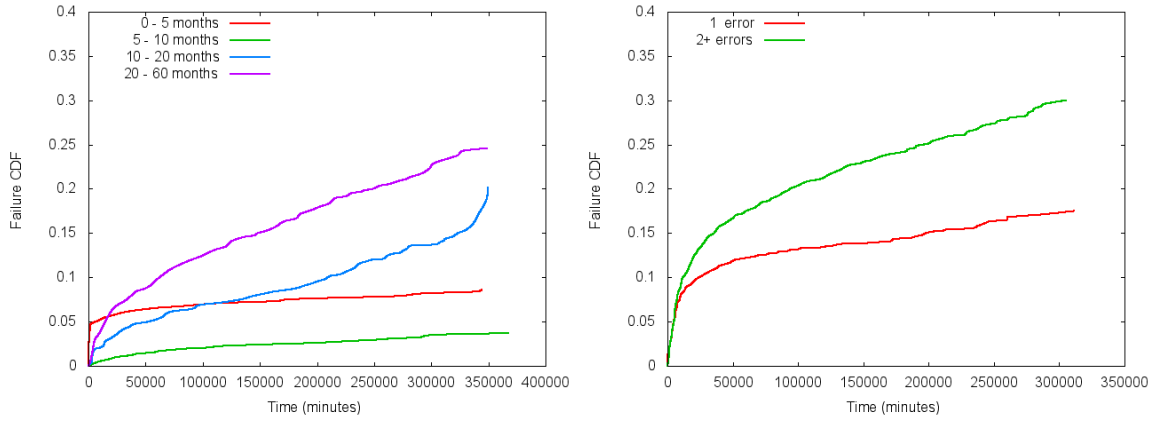


Figure 15. Reallocation Errors Failure CDF Graphs. The graph on the left provides the CDF of disk failure with respect to time dividing the disk population by age of first reallocation error occurrence. The graph on the right provides the CDF of disk failure with respect to time by the number of reallocation errors seen on the disk before failure.

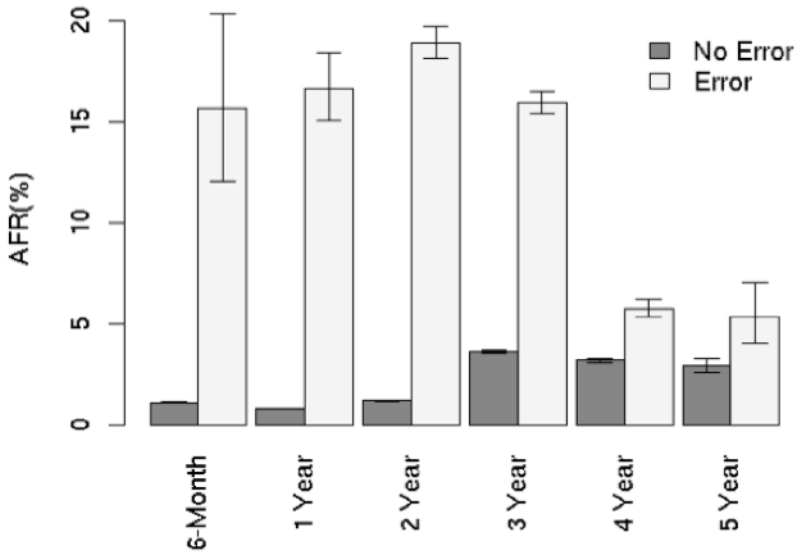


Figure 16. Offline Reallocation AFR.

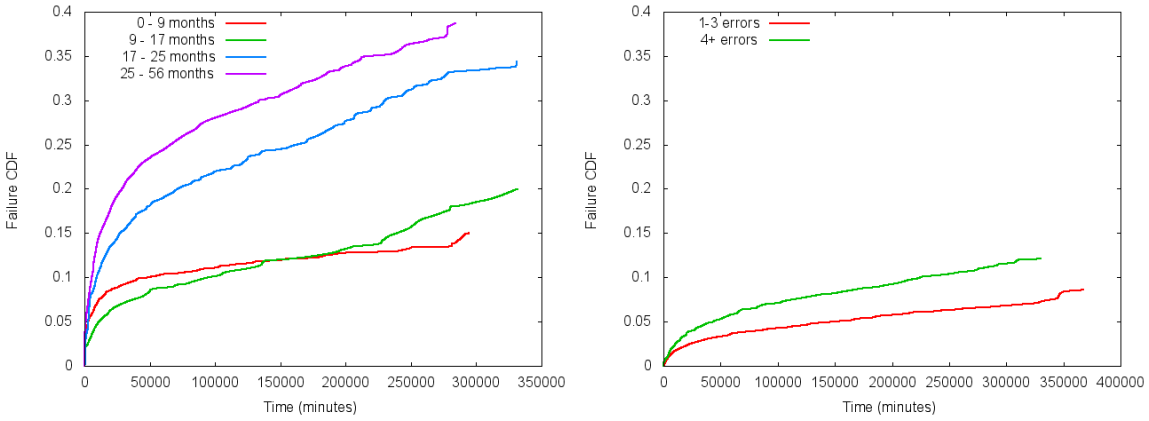


Figure 17. Offline Reallocation Errors Failure CDF Graphs. The graph on the left provides the CDF of disk failure with respect to time dividing the disk population by age of first offline reallocation error occurrence. The graph on the right provides the CDF of disk failure with respect to time by the number of offline reallocation errors seen on the disk before failure. (Reconstructed)

appears to be, and not surprisingly disks with more than one error fail more than disks with just one.

Probational Count

The last SMART error reviewed was probational count that occurs when a disk is unsure of the reliability of a particular sector but does not reallocate it. Instead, the disk puts the sector on probation to see if the issue resolves itself or persists, requiring replacement. The authors observed 2% of the disks within the population suffer at least one these error events during the nine-month timeframe.

The data reconstructed for probation count is presented below in Figure 18 and Figure 19. Figure 18 compares the yearly AFR's disks with probational counts against those without. For all ages, those disks with probational counts were statistically more likely to fail. Figure 19 has the failure CDF's for the probational counts grouped into four age groups: 0-11, 11-17, 17-23 and 23-48 months and by error counts of 1-2 and more than 2. With the exception of the first age group that has a higher failure rate than the 11-17-month group, the age groups have higher failure rates the older they are. There also is a substantial difference in failure rates for those disks that have 2 or fewer errors and those with more. The disks with more than 2 have much higher failure rates.

Relying on SMART errors alone, however, has several drawbacks. Given the lack of occurrence of predictive SMART errors on 56% of the disks preceding failure, relying solely on them would lead to large scale data loss. Also, many drives that experienced SMART errors did not end up failing within that time frame, so this would generate a significant number of false positives.

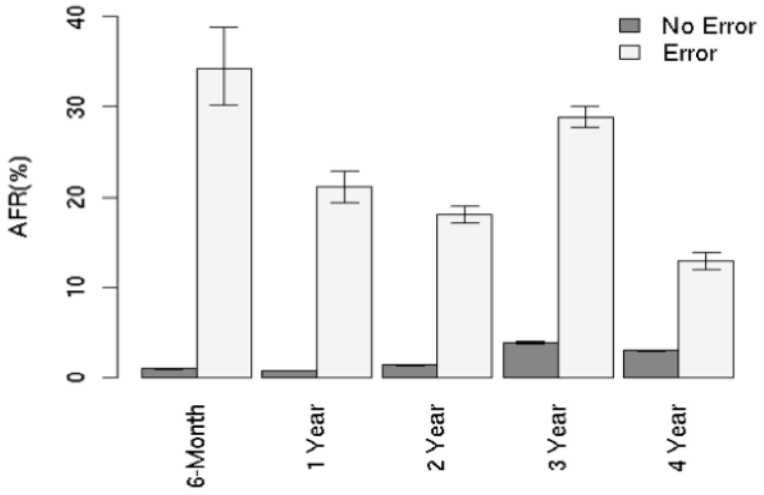


Figure 18. Google Probational Count AFR.

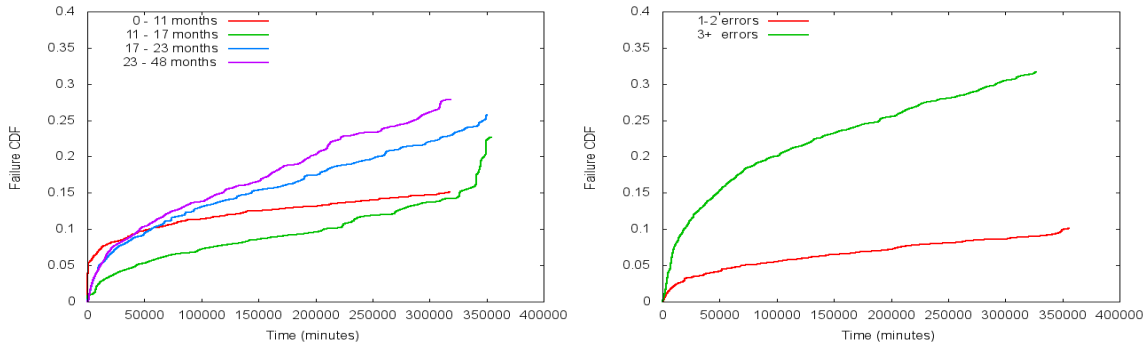


Figure 19. Probational Errors Failure CDF Graphs. The graph on the left provides the CDF of disk failure with respect to time dividing the disk population by age of first probational error occurrence. The graph on the right provides the CDF of disk failure with respect to time by the number of probational errors seen on the disk before failure. (Reconstructed)

Miscellaneous

Interestingly the authors also found a weaker correlation between utilization levels and failures than other works with only the first year and fifth year being impacted by higher utilization levels. Temperature, except in extreme cases, was ruled out as well having only a negligible impact.

Data Reliability

There are several proposed metrics for estimating the time it would take a storage system to have an unrecoverable data loss event due to storage failure. MTTDL, Mean Time To Data Loss, for the last two decades has been the standard reliability metric in most all academic work owing in part to its simplicity and easy formulation. It is based on Markov Models where each state is the number of working hard drives and transitioning from each state is governed by constant failure and repair rates. From a Markov Model, it is very easy to construct a closed expression for reliability analysis.

Several works [57], [58] have recently begun to find fault with the metric, though. Because MTTDL uses a Markov Model, it assumes that failure is an exponential distribution which from the failure trends is a rather dubious assumption. MTTDL only specifies a mean time of data loss but cannot provide information as to the degree of that data loss. While two storage systems might have the same MTTDL, the amount of data lost could be drastically different. Additionally, MTTDL lacks the ability to cover a given duration of system operation. Instead, it states the MTTDL from the start of operation to infinity. It is very unrealistic to assume that a storage system can continue infinitely and most people study a mission lifetime of less than a hundred years.

Greenan et al. introduced a new measurement scheme that is far superior for use in this dissertation called the Normalized Magnitude of Data Loss,

$NOMDL_t$. It states the expected amount of data loss for a given period of operation as a ratio with the amount of storage used. $NOMDL_t$ is a more informative metric for understanding data loss as it makes comparisons between different scenarios far easier and understandable. It has the advantage of being computable via a series of Monte Carlo simulations. As a simulation approach was already planned to be used in this dissertation, it a very natural fit for this work.

CHAPTER THREE
MATERIALS AND METHODS

LoDN Simulator

Introduction

There are multiple possible ways to proceed in investigating the feasibility of a proactive disk-aware approach. An empirical study would be too cost prohibitive with the amount of hardware and staff required to run operations and hosting at a multiple geographically disperse sites. Furthermore, it would take years to run just a single configuration and policy study. The next option would be analytical models via the use of Markov chains similar to [59]; however, with the possible range of different event types it would be too complex to model without too many simplifying assumptions. Markov Models are “memory-less” in that future events are solely dependent on the current state of the model. Past states have no influence in determining future event transitions. In the case of modeling a set of disks, any failure or repair transitions wipe out concurrent repair operations and reset the likelihood of failure. For a large system of disks, this unrealistic behavior can distort results that one would witness in the real world. Markov models also have an implicit assumption that most events will follow an exponential distribution which as discussed is not always the case. This leaves the possibility of using a discrete, event-based simulation to emulate the individual components of Logistical Networking. Simulation has much more latitude in options and sophistication than analytical models and is far less time consuming and expensive than the real thing. Simulation makes it possible to test the storage failure behaviors described previously and experiment with different policy mechanisms.

It is impossible to do an exhaustive analysis of all possible combination of factors. The configuration of network topologies alone is limitless. Therefore, the simulations will be done in Monte Carlo style with several static options like topology, data seeding and storage sizes while other variables such as failure behavior and data dispatcher policies are selectively targeted.

After reviewing possible pre-existing simulator frameworks, it was decided that none of them would be suitable for this study. The available simulators mostly focused on either the storage or networking elements and were too limiting for setting up the Logistical Networking stack. Instead, a discrete, event-based custom simulator, LoDNSimulator, was designed after researching materials on the implementation of simulations in [60], [61], [62]. The LoDNSimulator is driven by an event queue with events arriving from different components of the simulation including storage drives, depots, buses, network

links and LoDN. It takes as input a configuration file that specifies information about the depots, the local and wide area networks, storage and their failure behavior, data arrival patterns and the source clients. An overview of its components is provided in the following subsections.

Depots

The depot component simulates the IBP depots infrastructure on a global scale. Each depot acts as separate autonomous units that are interconnected via a set of topological network links. The IBP depots consist of several separate simulated components including the storage elements, buses that link the storage elements together and a network interface for external communication. The depots are organized and partitioned into multiple, geographically dispersed sites. At each site, the depots are connected via a local area network using a crossbar-like switch with a single external link to the global network. The depots implement a subset of the IBP protocol including allocation, store, deletion and an enhanced depot and resource query for the different disk metrics. Depots are configurable as to when they can join the Logistical Networking infrastructure, although this option was not utilized for this research.

Data Sources (Clients)

Simple clients were simulated to generate data objects at arrival times specified by the configuration file. The data sources can be independently configured in regards to their topology placement and data arrival patterns. The clients request allocations from LoDN using their location and data object size as part of the selection criteria and then perform an upload to the allocation provided. Once completed, the client informs LoDN of success and hands the data object off at that point. If a disk failure occurs during the data upload, the client will request a new allocation from LoDN and retry. For the simulations used in the research, there was a single client per site connected directly to the site's local area network.

Data Objects

Every data object is a separate, simulated entity with unique identifiers. The data objects are configurable for their size and arrival time. Every data object maintains a list of all of its replicas including the ones lost to disk failure. When a data object no longer has any valid replicas, it, in turn, is considered lost. In addition to other metrics, information about the time of loss and replicas are then dumped into a data store for analysis. For the simulations used, there were 3 million, 1-gigabyte data objects that arrived once every minute for 5.7 years.

Storage Elements

The storage elements are individually configurable and are assigned classes that represent their manufacturer, family and model providing the details for their capacity, read and write speed and failure behavior. The storage

elements maintain their current status including what data objects currently reside on them, the used and free space and the information necessary to calculate their failure time based on the failure model assigned to them. The storage devices can be configured to arrive in the system either statically or dynamically. Statically refers to their initial distribution in the system, and dynamically refers their replacement behavior, i.e. after a drive fails how long it takes to be replaced. A storage element can represent an array of different real life storage devices including SSD's and optical disks, but for this work are analogous to hard drives.

The storage elements are unique in the simulator in that they are the only components that can suffer errors and fail. While their failure behavior is configurable, there are two key assumptions in place: no bit rot of data and fail-stop. If a disk is operable and has a replica, then that replica is accessible without error until either the disk fails or the replica is deleted. When a disk fails, it fails in totality; there is no chance of data recovery. After a failure occurs the components of the simulation will be unaware of the failure until the depot is probed by the data dispatcher for up-to-date information on the storage media. Only at that time will LoDN and the data dispatcher be able to take action to handle the failure event.

Buses and Network Links

Like the other components, each bus and network link is an independent element within the simulator that serves to create data flow paths between clients, depots and disks. Every path between two depots is a series of one or more network links, and each link can have multiple sources and destination links. The input configuration file specifies the topology of the network and the characteristics of each link including the bandwidth. Unlike links which are unidirectional, buses are bidirectional allowing reading and writing from multiple drives. Routing for the network paths is handled by a global, static lookup table specified in the configuration file. Currently, there is only one route per source and destination pair as links are assumed to never fail.

The data going across a network link is encapsulated within data packets. These data packets are not like conventional IP packets lacking size restrictions, traffic shaping and more advanced options. Instead, they behave more like a segment of a particular data flow. They have the destination address and IBP allocation on the depot to which the data is to be delivered. The packets can be fragmented into smaller packets to meet the fair share bandwidth and queue limits of the next link in the path. A packet can range in size from 1 byte to the full size of the data object. At one second intervals, buses and links with data to transmit forward data to the next link within the limits of the bandwidth and queue restrictions of the next component. As no TCP-like layer handles retransmission and congestion control, packets are never dropped. When queues fill up, they can cause backlogs to build up along paths. This is not considered to be impactful on the results of the simulator as each link has a 1-second latency and

the data packets can be regarded as a large payload of individual IP packets and the latency would provide enough time for things like retransmission.

Sites

Named sites are not technically a component of the simulator, but a logical abstraction used for grouping actual components together and simplifying the configuration file. A site has an automatic local area backbone link and external links to the wide area with settings specifiable in the configuration file. When depots and clients are assigned to a site, they are automatically patched into the local area network for the site with the internal routing updated. This reduces the static routing table in the configuration with routes only having to be specified between sites. For these simulations, all the sites have a 10 GB/s internal backbone and a 10 GB/s external link. Although some of the external links connect directly to a smaller capacity link, e.g. 1 GB/s.

LoDN

The final and most complex component of the simulator is LoDN and its data dispatcher. The simulated LoDN mirrors its real-life counterpart being nearly as complex with only the error handling paths being reduced. It maintains a separate database from the rest of the simulation that serves as metadata store for LoDN's current working knowledge of the simulation. Within this database, it stores the IBP capabilities for the current allocations and exNodes of the data objects, as well as the relevant information for the available storage media and depots. This database is created to be extensible for the data dispatcher and the various policies.

The knowledge that LoDN possesses is based on periodic probing of the depots and the result of the operations it executes. When a depot arrives in the simulation, the depot registers itself with LoDN. LoDN will periodic probe the depot for state changes especially in regards to its hard drives learning about new and remapped disks, failures and other drive media events like SMART errors. The probing interval is configurable with one day used for this simulations. Therefore, it might take LoDN a full day to learn about a disk failure and take action on performing any recovery or repair operations. This behavior is fairly realistic with the real world LoDN taking time to recognize failures and distinguishing permanent from transient errors.

LoDN is capable of executing three different operation types on behalf of the client and data management policy: allocate, delete and copy. Each of these operations is relatively flexible and take callbacks from the requestor that handles the results of the operation. These callbacks allow for sophisticated behaviors to be achieved, by allowing requestors, in particular, the policy to chain multiple operations together based on the result of the current operation. The sequence of operation execution is managed by LoDN's scheduler that uses per assignable operation priorities to determine job order. The scheduler is configured to issue up to 100 separate operations per second.

Within LoDN it is the data dispatcher that handles the creation of operations for replicating and maintaining data objects according to a specified policy. The data dispatcher has a set of call outs for implementing a policy framework that allows for multiple behavioral modes of the dispatcher. The three modes used for LoDN's data dispatcher in this dissertation are:

1. Traditional and simple that only handles the initial upload and distribution of the data object.
2. Reactive with data replication being induced by disk failure.
3. Enhanced proactive data replication with intelligent, storage-aware heuristics developed for study in this dissertation.

The exact setup of the policy framework and the policies themselves are discussed later in this chapter.

Failure Models

Based on the disk failure research, several different failure models have been developed for study. As indicated in [39], SMART errors are a prominent cause and indicator of impending disk failure. From the information provided from the paper, two different failure models were created that use the information and statistics provided. The first model generates SMART errors and disk failures according to the relationship between disk age and the SMART error types. The second model uses the relationship demonstrated between the frequency of the different SMART error types and disk failure. The findings from [63] and [10] and the CFDR were used to create a disk age based failure model using a function based on the monthly ARR frequency described in the paper instead of a traditional bathtub curve. Finally, while there was not much concrete information about the effects on manufacturer, model and vintage; a model, nevertheless, was created from Elerath's research with the vintages in [56] due to the long perceived importance of this factor on disk reliability. While these models are far from an exhaustive list of possible factors, they do represent the most pertain research on the subject from the last several years. Additionally, they cover several different aspects of anticipating disk failure from individual indicators on a disk, i.e. SMART errors, to large disk population dynamics with age and vintage.

The following subsections examine the implementation of the models from the information gathered and how the models were analyzed and validated against the original datasets.

SMART Error Failure Models

From [39], two models were created using the paper's data and analysis of SMART errors: the first one based on the lethality of SMART errors by disk age and the second one by the lethality of SMART errors by event count. Unfortunately, the raw data for their analysis was not available for review nor the results used to generate the graphs and statistics, and personal correspondence with the authors was fruitless. The generated failure models had to rely solely on

their analysis and graphs, and several assumptions had to be formulated. As these assumptions obviously affect the accuracy of the failure models, the necessity and justification for making them will be discussed throughout this section.

It is tempting to use the statistics from the five-year AFR graphs for the creation of an age based failure model, but as stated within the paper, the different manufacturers and models for each year may be a larger contributor to the failure rate. The age of each disk was highly correlated to particular sets of manufacturer and model. While this may be true, in the absence of better data these two models used these numbers for calculating the total failure rate for each year of a disks' life. Using a graph digitalizing software package, FindGraph [64], to reconstruct data points, the following failure rates in Table 2 were extracted from Figure 2 of [39]. No age-based breakdown of SMART error versus non-SMART error induced failures was given, only a raw percentage of 56% for the entire disk population that had failed without any SMART error

Table 2. Google AFR by Age (Reconstructed)

Age (In years)	Annualized Failure Rate
1	0.01727
2	0.08013
3	0.08652
4	0.05935
5	0.07347

occurrences in the 8-month period prior to their failure. Therefore, it is assumed for each year of a disk's life that 56% of the failures will not have a preceding SMART error within eight months.

The four most prominent errors were SCAN Errors, Probational Count, Reallocation Count Errors and Offline Reallocation Count. Table 3 gives the distribution of the SMART errors witnessed within the population.

For each of these SMART error types, two sets of survival probability analysis were performed. The first set examined the survival probability over eight months since the first error occurred on the disk with the disks grouped into various age brackets, revealing the lethality of SMART errors on different disk ages. The second set examined the lethality of SMART errors by the frequency counts of the same type. For the various SMART error types, their counts were bracketed into groups and graphed by survival probability over time for eight months. This information was presented in Figures 8, 11, 12 and 13 and the data points for disk survivability over time CDF's were extracted using FindGraph.

Table 3. Google SMART Annual Rate by Type

SMART Error Type	5 Year Distribution Percentage
Scan Error	2%
Probational Count	2%
Reallocation Count	9%
Offline Reallocation Count	4% *(Some ambiguity exists due to Offline Reallocation Count being a subset of Reallocation Count, so it is not clear if this 4% is part of the 9% or an independent.)

There were several difficulties encountered when applying this information to create error models. There was no data regarding the combination of smart error types, a lack of refinement in the age of the disks and the timeframe of SMART error count groupings.

SMART Error Age Failure Model.

For the SMART error age model, each SMART error type has a separate failure probability generator based on the data from the age survivability graphs. Each generator uses a failure probability engine selected according to the age of the disk profile at the time of the error. The failure probability engines are produced based on the survival CDF, R(t), data points for the different disk age groups. The data points are refined to ensure they are monotonically decreasing. The monotonic property is guaranteed since the data points represent a CDF ensuring that $F^{-1}\big(F(t)\big) = t$ in the lookup tables.

The data points are converted to their failure CDF, $F(t)$, values using the property that $F(t) = 1 - R(t)$. Then the inverse distribution, F^{-1}, is found for each set of data points via spline interpolation using the $F(t)$ data points for x values and the t values as the y values. Since $F^{-1}\big(F(t)\big) = t$, the F^{-1} spline is used to generate a set of data points, t, from 0 to the maximum value of the failure CDF at 8 months. These resultant data points are used in lookup tables within the generators for failure probability generation.

The failure probability generator works by using a uniform random probability engine that generates numbers [0,1). If the random value is above the maximum value for the failure CDF, F, then the generator produces no error. Otherwise, the random value is used as the key into lookup table to obtain the time, t, at which failure will occur within 8 months of the SMART error. Using this method, allows the survivability CDF, $R(t)$, to be transformed into a failure probability engine that has correct failure rate, f, according to [60].

This failure model produces 100,000 disk profiles at a time, the same number as stated in the paper. These profiles are then used to dictate the behavior of that many corresponding disks. When the simulator has exhausted the current set of profiles, the model generates a new, independent set of

profiles. For each year of the 5-year span, the model calculates the number of disks that are supposed to failure based on the AFR for that year in Table 2. 56% of those will fail with no SMART errors within eight months of the failure time and 44% will. Of the drive profiles that are still operational by the start of that year, 2%, 2%, 9% and 4% will be uniform randomly chosen to have SCAN, Probational Count, Reallocation Count and Offline Reallocation Count errors respectively. The SMART error candidate disks selected for each error group are not mutually exclusive since no data was provided about their exclusivity. Therefore, some disks may suffer multiple error types within a year. These disks are then uniformly randomly assigned their corresponding SMART errors with the error selected to occur at some point within that year. When these disks are allotted an error, the disk profile is assigned a failure probability generator. The generator based on the disk's age probabilistically determines if and when the disk will fail from that SMART error. If the disk is designated to fail, the disk profile records the failure time.

This process of SMART error assignment continues until the number disks needed to fail that year are meet to ensure that 44% SMART error related failure statistic. Every one of these profiles is guaranteed to have at least one SMART error failure meeting the SMART error distribution requirement, thus meeting the SMART error occurrence distribution. Then 56% of the disk profiles that are to fail that particular year without a SMART error are chosen randomly. They are initially assigned a uniformly random time to fail that year. Since the possibility exists that a SMART error may have occurred in the previous year for that profile and the 8-month timeframe from that error may extend into the current year, each of these disk profiles is checked to ensure their failure times are outside of that range. If such a condition is detected, then the disk profile is reassigned a new failure time outside of the overlap. This prevents the possibility of having too many disks that appear to die from SMART errors.

SMART Error Count Failure Model

This model is constructed very similar to the previous model utilizing the same statistics for the SMART error distribution and failure patterns. The SMART error candidate and disk failure selection mechanisms work in the same manner. The difference arises mostly from using a different set of SMART error generators. The new SMART error type generators are based on the disk survival probabilities by error count. When a candidate profile is assigned a SMART error, the generator randomly assigns a time for the error to occur and examines the disk profile for any errors of the same that occur on the disk within in a several month window. The number of such errors is used as the profile's error count to select the current survivability distribution. Using this distribution, the generator probabilistically determines if and when the disk profile is going to fail for that year. The profiles are assigned an increasing number of SMART errors and re-evaluated by the generators until the failure quota for SMART errors is reached.

Validation

Once implementation of the models was finalized, the next step was to conduct a validation and verification analysis, similar to one the performed in the original paper. To this end, 1,000 simulations with each simulation consisting of 100,000 disk profiles were executed to capture the failure the behavior of the hard drives. The SMART error distribution and failure population patterns were found to be in general agreement with the perceived interpretation of the findings. This section details the results from these simulation runs and provides a comparison to the original data.

As a first step, the annualized failure rates of the disks were recorded for their 5-year lifetimes and compared against with the expected rates. As illustrated in Figure 20 and Figure 22, the yearly AFRs are virtually identical between the two models and the expected rates; in fact, the standard deviation error bars are not even visible within in the graphs.

The annualized disk failure rates caused by the four SMART error types for the two models was examined next. Figure 21 and Figure 23 provide the disk failure rate by year and type of SMART error for the models. While every simulation was not perfectly identical to the expected rates provided, in almost all cases, the expected rates laid within one standard deviation of the populations' statistical mean. Some notable exceptions occurred for the SMART error count model, where in years 1 and 2, the disk failure rate for each of the SMART errors was slightly greater than expected. This discrepancy was subsequently offset by the disk failure rate being too low in year 5 for the SMART error count model. The reason for the variance in the count model was the problematic nature of getting the yearly AFR's of the general population to match the expected values while maintaining the intended frequency of the SMART errors and their lethality. While this dissertation acknowledges that a difference between the expected and the simulated count model exists, it is assumed that difference should be too small to invalidate the findings in chapters 4 and 5.

Once the general and SMART error AFR's of the models were verified, the next check was to confirm the validity of the individual SMART error lethalities. Not only must the prescribed number of disks fail annually, but the SMART error induced disk failures must be distributed correctly according to the SMART error type, count and disk age. It is not enough for a disk to fail after a set of SMART errors, but its failure distribution must closely match those in the Google paper.

The disk failure profiles of the 1,000 sample simulations were scrutinized by segregating the cause of the failure, the age of the disk when the SMART error event occurred and if and when the disk failed as a result. The empirical cumulative failure distributions for each combination were compared against their expected distributions. Figure 24 on the following pages provides a breakdown of this analysis by plotting each the combinations, in red, against their expected values, in blue, over time. Included in the graphs, as dashed red lines, are the 95% confidence bounds for the empirical CDFs. With a few exceptions, the

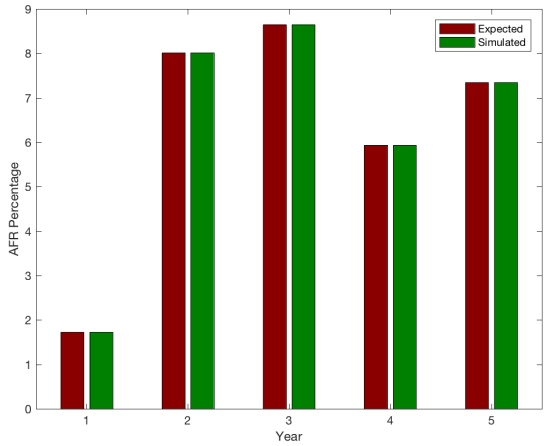


Figure 20. Comparison of the AFR between the reconstructed Google data and the failure model.

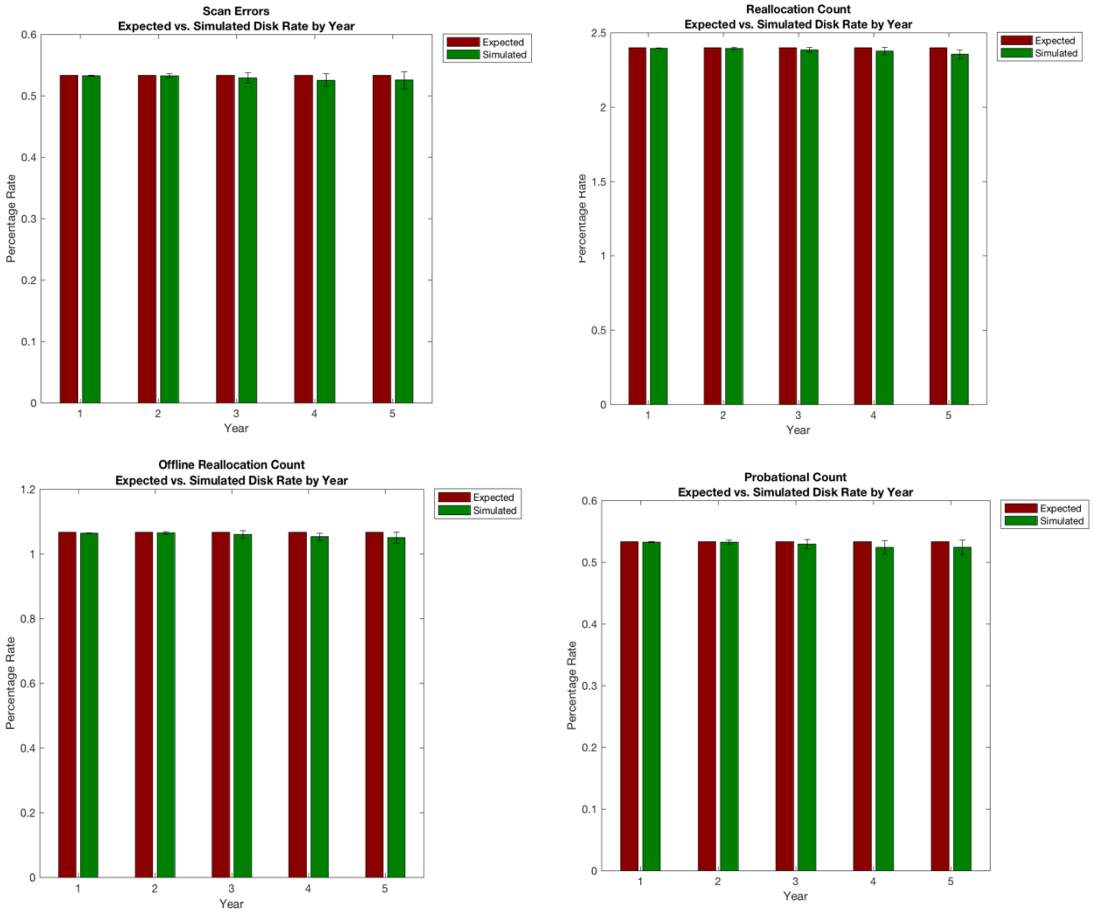


Figure 21. Comparison between the Google expected and the Model simulated AFR's. There are one standard deviation error bars for each of the four SMART error type.

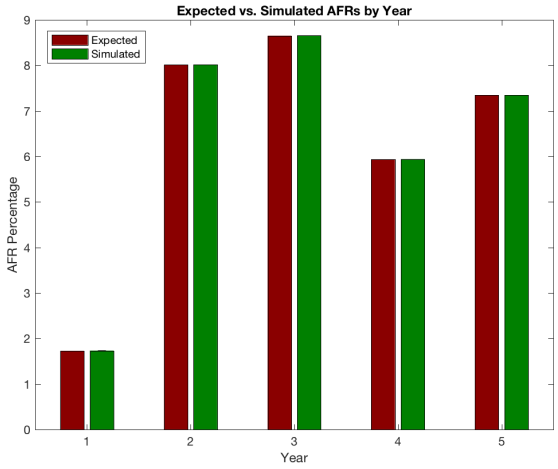


Figure 22. Comparison between Google's expected and Count Model simulated AFR's.

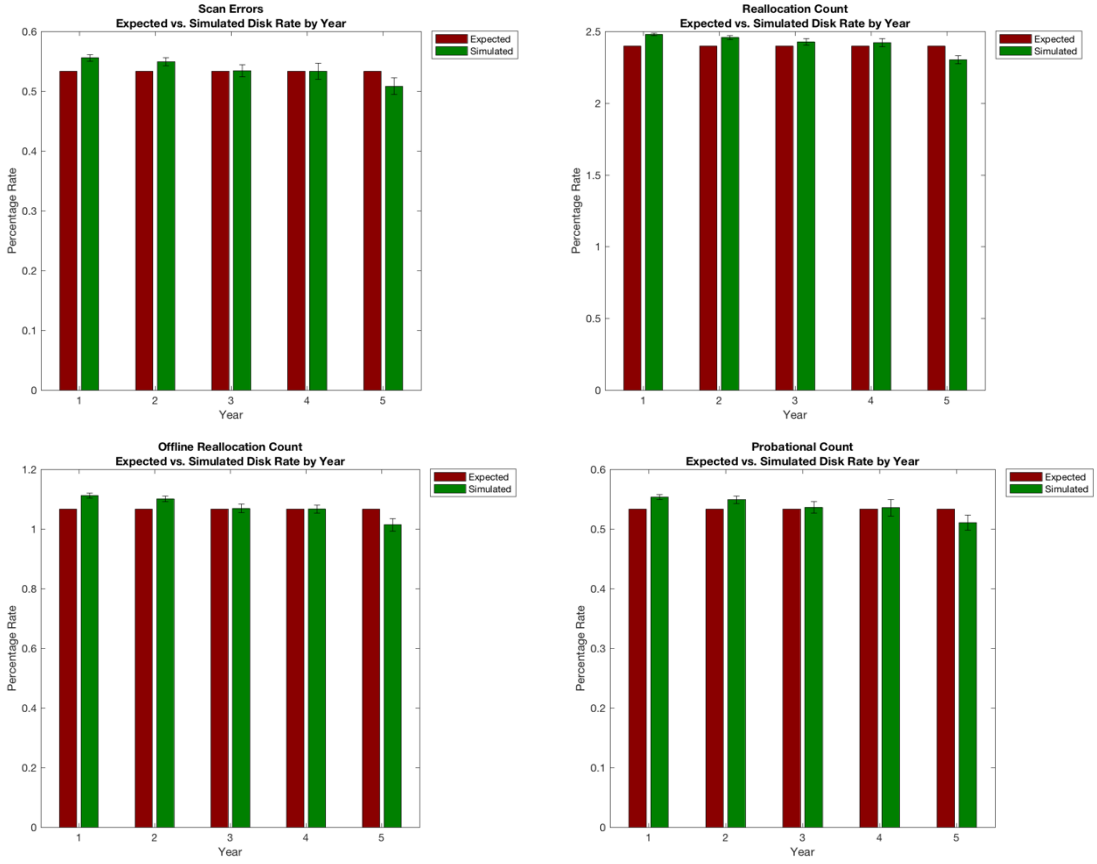
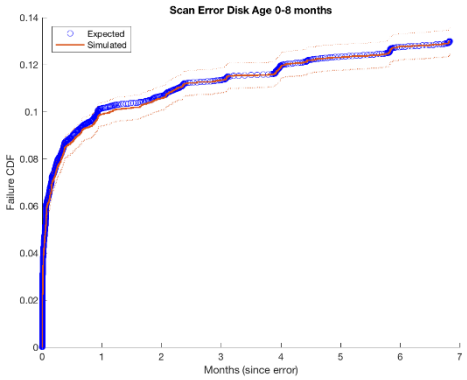
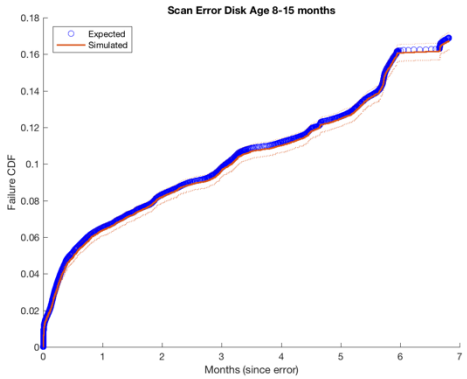
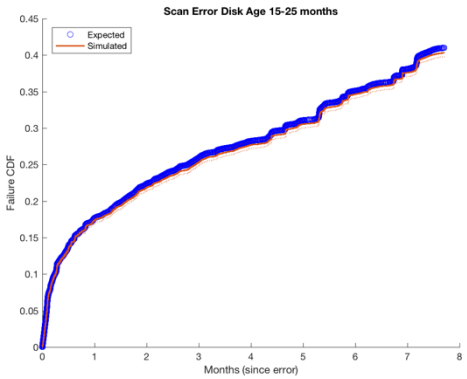


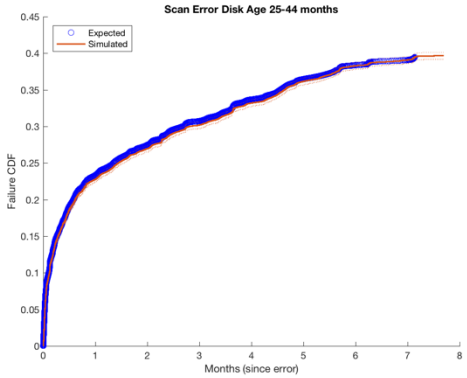
Figure 23. Comparison between the AFR's of Google's expected data and the Count model's simulated results for each of the four SMART error types. The simulated bars include an error bar for one standard deviation.

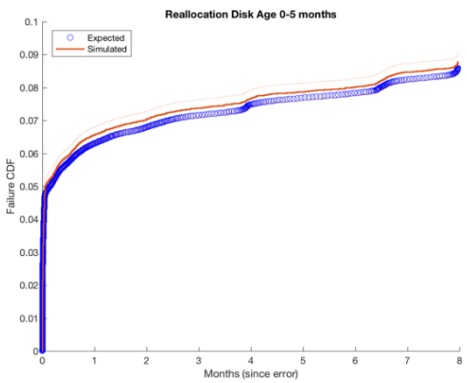
Figure 24. Lethality of SMART errors graphed by type and disk age.

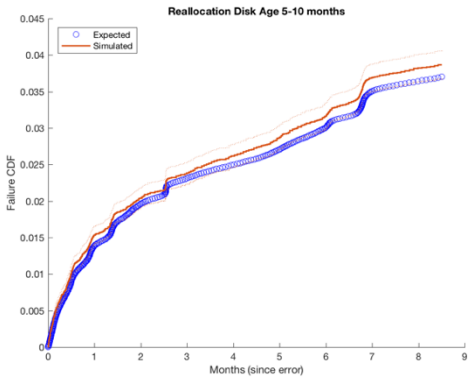


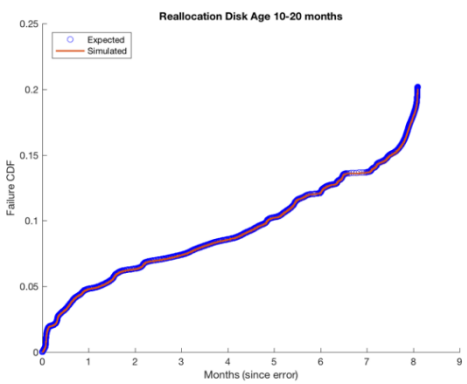












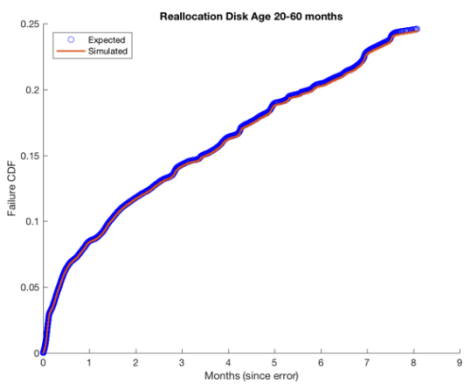
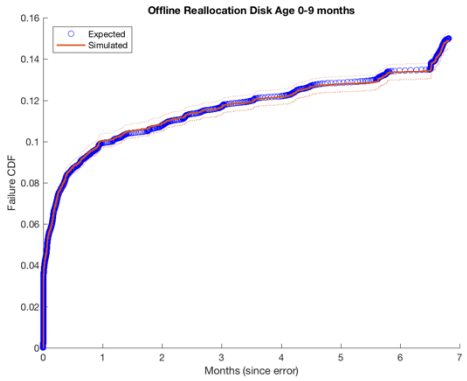
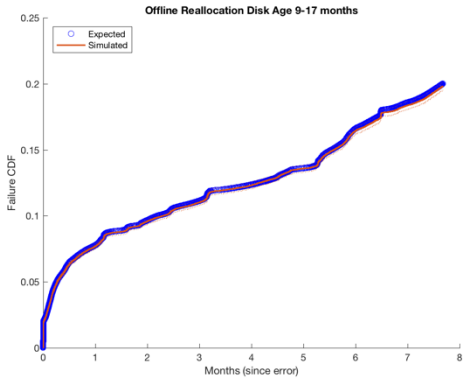
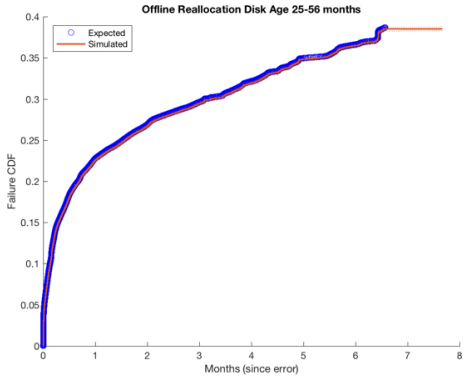
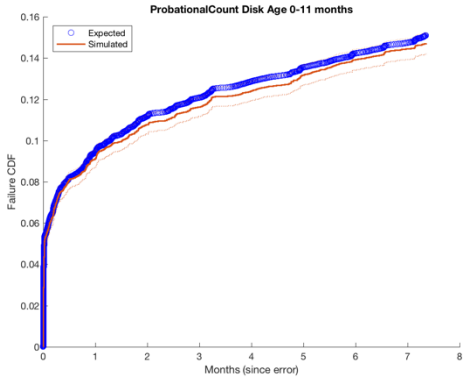


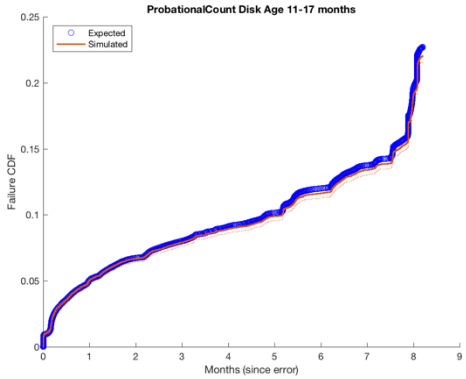
Figure 24. Continued.

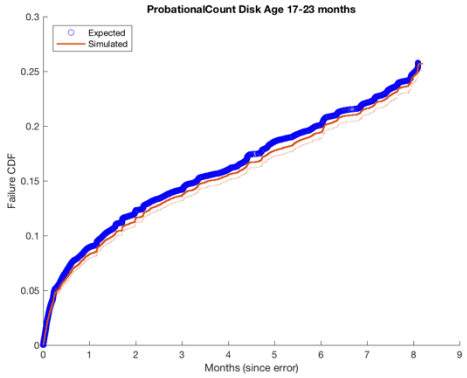












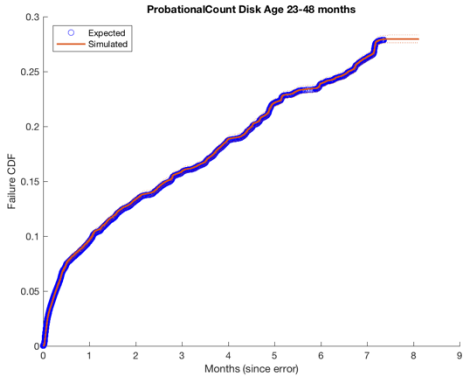


Figure 24. Continued.

generated failure distributions match the expected data provided. Those distributions that do differentiate still have the original data fall within their confidence bounds meaning that subsequent simulations could also generate those distribution patterns.

With the validation complete against the original datasets, the general CDF's of the models can be visualized to see out how SMART errors and disk failures will be experienced within simulations. Figure 25 and Figure 26 provide a breakdown of the different CDF's for the age and count-based models. The left graph in the first figure gives the cumulative failure for the general disk population under the SMART error age model and the failure distributions for SMART and non-SMART error related. As can be seen from the figure, the total failure distribution is what would be seen based on the AFR's from Table 2. The percentage of SMART to non-SMART error failure fluctuates over time but stays around the 44/56% split. The right graph in the same figure plots the individual failure distributions of each of the SMART error disk failures. This graph provides the cumulative failures caused by each SMART error type over a 5-year period taking into account both their lethality and frequency of occurrence. For instance, by year five the reallocation error type will be responsible for killing 7% of the disks.

Figure 26 provides a similar rundown for the SMART error count failure model. The graph on the left has the CDF for all of the disks and the individual CDF's for the disks that failed with or without an associated SMART error. The overall failure ratio is 44% to 56% of SMART to non-SMART errors. The graph on the right has the CDF for each SMART error type, providing their total lethalities within the disk population.

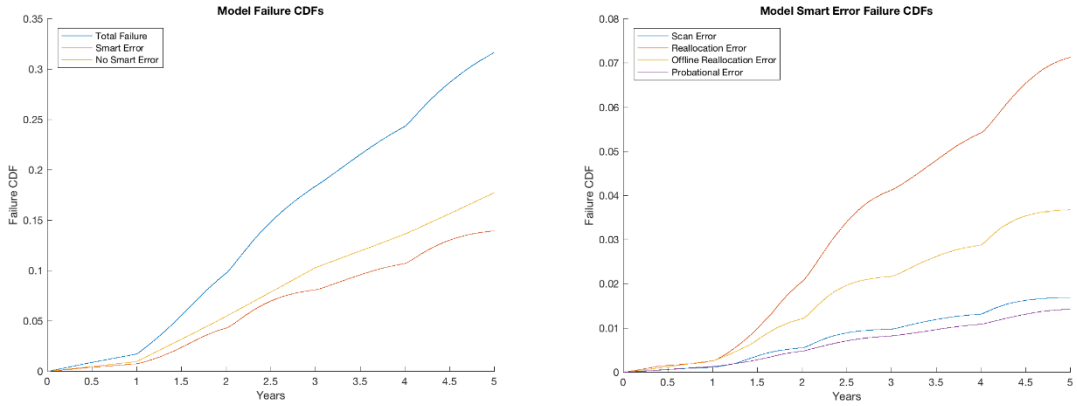


Figure 25. SMART Error age failure model CDFs. Failure CDFs split into total cumulative failure, cumulative failures due to smart errors and cumulative failures lacking a corresponding SMART error are on the left. Cumulative failures due to the individual SMART error types are on the right.

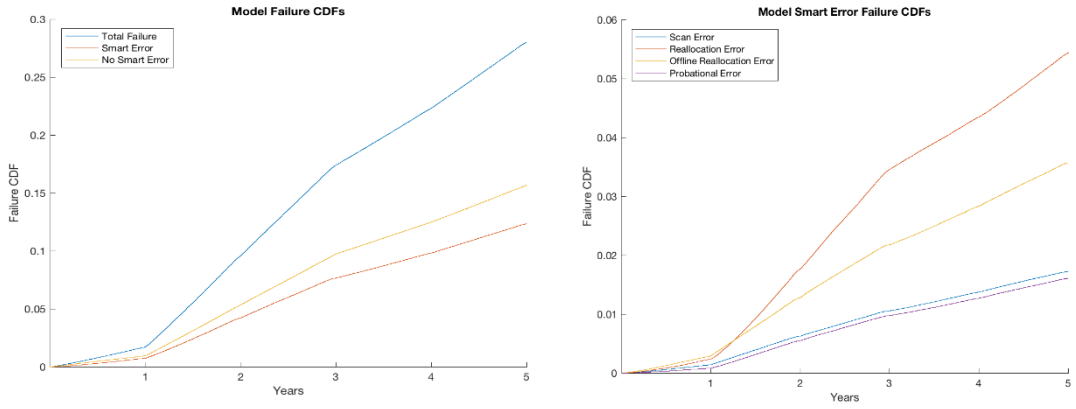


Figure 26. SMART error count failure model CDFs. The count model's total cumulative failure including the cumulative failure due to SMART errors and without and the cumulative failures due to the individual SMART error types.

Age (CFDR) Model

From the analysis performed by Drs. Schroeder and Gibson on the Computer Failure Data Repository, an age-based model was developed. [10] analyzed the hardware replacement logs, in particular, disk replacement, of seven different sites with a total of 100,000 disks from 4 separate vendors. Most of the studied populations were of too short a duration to be used for the creation of a long-term failure model with some sites only being covered for a few months. As such the decision was made to focus solely on the HPC1 dataset that covered a five-year period with several thousand disks. The crucial analysis performed for this model was a set of disk ARR's, annual replacement rate, where the ARR served as a proxy for disk age. As with the previous models, attempts to obtain access to the raw data via either the Computer Failure Data Repository or personal contact with the authors proved fruitless. Most the data used for the creation of the model was from the results sets and reconstructed from Figure 4 of [10] using FindGraph on the HPC1 nodes. The figure provided the ARR for each month of operation for the disk population for both the compute and file system nodes.

This age model assumes a 5-year maximum lifetime for the disks using the monthly replacement rate, as a stand in for the failure rate of the disks of the corresponding age. To widen the sample pool and improve accuracy, the model's failure rates use the weighted average of the compute and file system datasets. By incorporating both datasets, the total population of disks being examined is tripled in size. Since the total of number disks over time in each population is not available, the weight applied to the individual ARR's is the ending disk count for each population. This data was given in Table 1 of the paper where the ending disk counts for compute and file system nodes were

2,318 and 1,088, respectively. As a result, the compute node population has over twice the influence on the combined ARR. The disks in the compute nodes have a significantly greater chance of failure than disks for the file system nodes, so this gets incorporated into the model.

The implementation of the model works by generating a population of 5,000 disk profiles at a time. For each month, it computes the number of disks that need to fail for that month. The failure number is computed by taking the remaining number of disks and multiplying by the ARR for that month. The number of disks to fail in that period is subtracted from the remaining disks for the subsequent months. For each disk selected to fail, the exact failure time within the particular month is uniformly randomly selected. Once all 60 months are tabulated, the list of profiles is shuffled randomly for later selection as disks are created in the simulation. The generation of disk profiles is continually repeated as the list of disk profiles becomes exhausted.

This model makes the following set of assumptions. The first is that failure means FAILURE. In the real world, disks do not always completely stop immediately; instead, performance and behavioral characteristics degrade to the point that the end user deems the disk needs to be replaced because it has "failed." At this point, the disk is still readable and most or all of the data may still be accessible causing only partial failure. Given the difficulty in modeling partial failures and the lack of information for such events versus complete failures, the model shall be based on the assumption that all documented failures are complete with no chance of data recovery and not just elective replacements.

Secondly, while the authors of the paper did an excellent job describing their methodology, it is not clear if replacement disks were included in the HPC1 population. If these new, younger disks were included then that could shrew the data used, i.e. a disk that is a replacement disk that is only one-year old could fail and need to be replaced in year four of the population. Only the ability to access the HPC1 dataset at the CFDR or a response from the authors could resolve this lingering question. Therefore, this model will assume that the age listed for the population is also the age of the disks.

Validation

Validating the CFDR age model was far simpler than the SMART error models since disk age was the only relevant factor. 1,000 separate model simulations were performed with the disk failure times recorded. The failure times were binned into monthly intervals and had their frequency analyzed. The histogram in Figure 27 compares the combined ARR from the reconstructed data and the results from the simulations. There is no discernable discrepancy between the original data and what failures the model produces.

As stated, this model relies on using ARR as a proxy for the estimated failure rate thus while this model can generate failures that produce matching ARR histograms, it may differ from the underlying values used for the original.

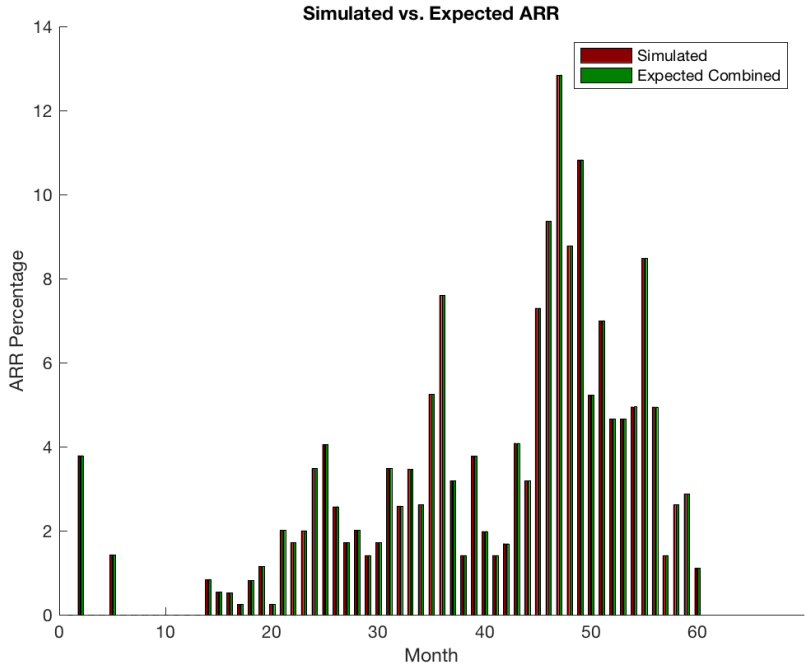


Figure 27. CFDR Simulated vs. Real ARR. A comparison of the expected annualized replacement rate for the combined HPC1 datasets vs. the simulation results from the failure model.

To ensure that the failure trend is well understood by the reader, Figure 28, provides the complete the CDF (left) and PDF (right) functions that the model obeys.

Manufacturer, Model and Vintage (Elerath) Model

Of the research done in the last two decades of disk reliability related to manufacturer, model and vintage, no one has published as much work on the subject as much as Elerath. His experience with NetApp allowed him to perform analysis on large populations and subpopulations of hard disks and subsequently analyzing and refining his work for several years. He was able to corroborate the long held belief of the impact of manufacturer, model and vintage on the long term reliability of hard drives. Due to his insights, and the lack of better data elsewhere, his results were chosen for the basis of a hypothetical disk manufacturer, model and vintage failure model for study.

While his work and findings have been of great importance, it is also incomplete for the purposes of creating a fully-fledged failure model. The datasets in [56] extend just a little beyond 10,000 hours. As with the previous models, no raw datasets are available for direct study. Additionally, he purposefully and admittedly withheld data from his published work and figures for

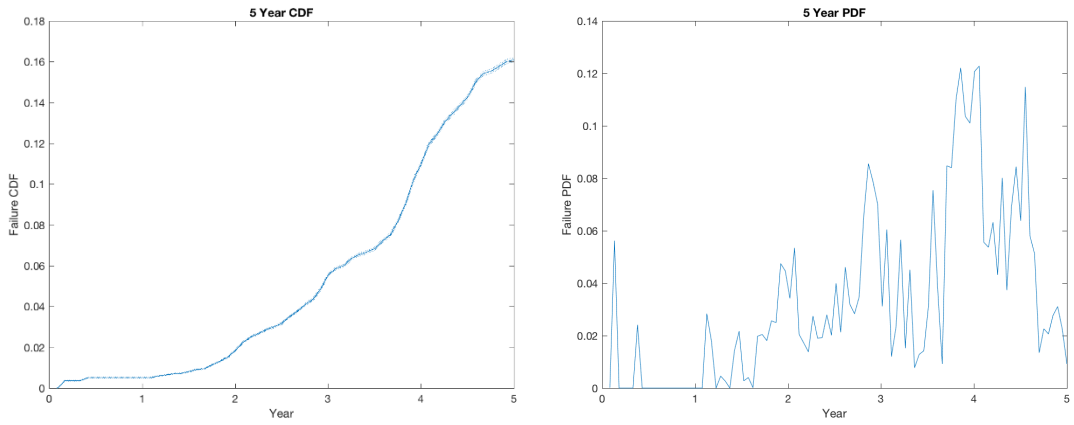


Figure 28. Simulated CFDR model failure CDF and PDF. On the left, the cumulative failure function and on the right the probability failure function for the 5 year lifetime of the disks.

fear of compromising the anonymity of hard manufacturers. While this is quite understandable, it has made producing a more accurate and complex reliability model impossible. As a result, the model relies solely on his final analysis and publications with some generous assumptions.

Like the SMART error and age-based models, digitalizing software will be required to reconstruct data from published graphs. Elerath et al. already performed much of the reliability analysis needed for the model. The hard drive population was partitioned into six distinct vintages based on the manufacturing date of the hard drives. For each of these vintages, he produced individual Weibull CDF distributions of best fit for the population data.

The Weibull distribution [60], [62] is one of the most commonly used distributions for fatigue and survival analysis. The Weibull distribution is extremely flexible and changes shape based on the value of shape parameter. When the shape parameter is less than 1, the failure rate decreases over time; greater than 1, the failure rate increases; and equal to 1, the failure rate is constant. At a value of 1, the distribution reduces to the exponential distribution. The scale parameter handles the "spread" of the distribution, the greater the value, the more spread out the distribution is. Because of its unique properties, it has been used to generate a pseudo-bathtub distribution by combining multiple different Weibull distributions simultaneously such as in [65]. The PDF and CDF for the Weibull distribution are given below Equation 1 in and Equation 2 where t is the time to failure, β is the shape parameter and η is the scale parameter.

The reconstructed data is provided in the following table (Table 4) as the parameters of the Weibull CDF curves.

This model makes an assumption that while the Weibull distributions are imperfect, they are good enough. The data analyzed by Elerath et al., only covers a time frame extending one-year in duration, falling short of the typical

Equation 1. Weibull PDF.

$$f(t,\eta,\beta) = \frac{\beta}{\eta}\left(\frac{t}{\eta}\right)^{k-1} e^{-(t/\eta)^{\beta}} \quad t \geq 0$$

Equation 2. Weibull CDF.

$$F(t,\eta,\beta) = 1 - e^{-(t/\eta)^{\beta}} \; t \geq 0$$

usage lifetime of a hard drive. Therefore, the model will assume a maximum five-year lifetime per disk before automatic replacement by a fresh disk, bringing the rejuvenation cycle in line with the other reliability models under study. Extending the lifetime necessitates the belief that that the Weibull distributions in Figure 29 can be used to predict the disk vintage reliability for the full timeframe. To this end, a MATLAB [66] program was created to extend the given distribution and generate the CDF and PDF distributions for the six vintages using the data produced by Elerath in the previous table. The 5-year CDF is given in Figure 30 and Figure 31.

The six different Weibull curves of the vintages form complex and interwoven reliabilities over five years. Vintage 1 initially has a much higher failure rate than most of the other vintages but is rapidly overtaken by the other five. Vintage 5 also has an initially high failure with a sharp increase before being overtaken first by the failure rate of Vintage 3 and then Vintage 6 around the half year mark. Vintage 6 starts low but climbs rapidly overtaking all of the rest after the first year. The overlapping and changing failure rates should prove interesting to see how a proactive policy can adjust to the frequent changes and maintain the data objects.

For the disk population, the standard depot and disk layout was modified slightly from the other models. As before there were sixteen depots for the nine sites with each depot having two buses with 16 disks per bus, meaning that there

Table 4. Elerath Weibull CDF values.

Vintage	Shape, β, (k)	Scale, characteristic life (η)
Vintage 1	0.9839	6.7411E+5
Vintage 2	1.2963	1.3262E+5
Vintage 3	1.3577	8.4842E+4
Vintage 4	1.0987	4.5444E+5
Vintage 5	1.2162	1.2566E+5
Vintage 6	1.4873	7.5012E+4

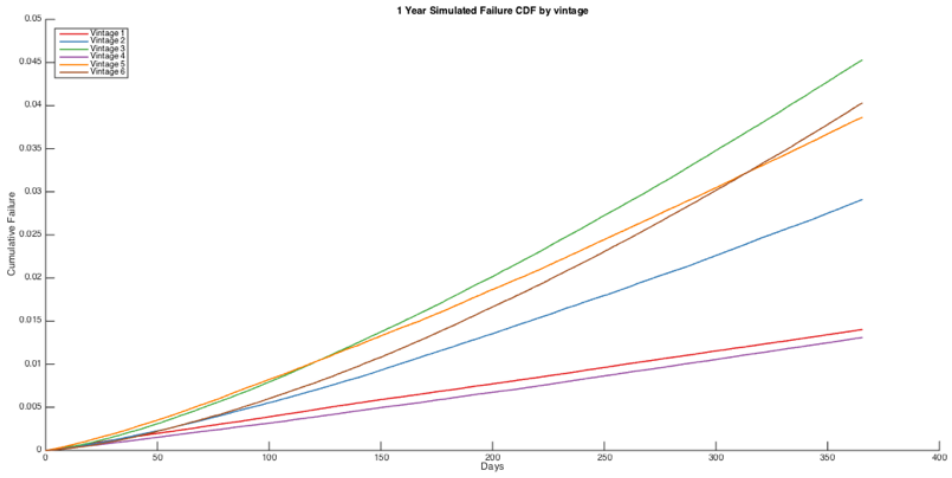


Figure 29. 1 year CDF of the Vintage model by vintage.

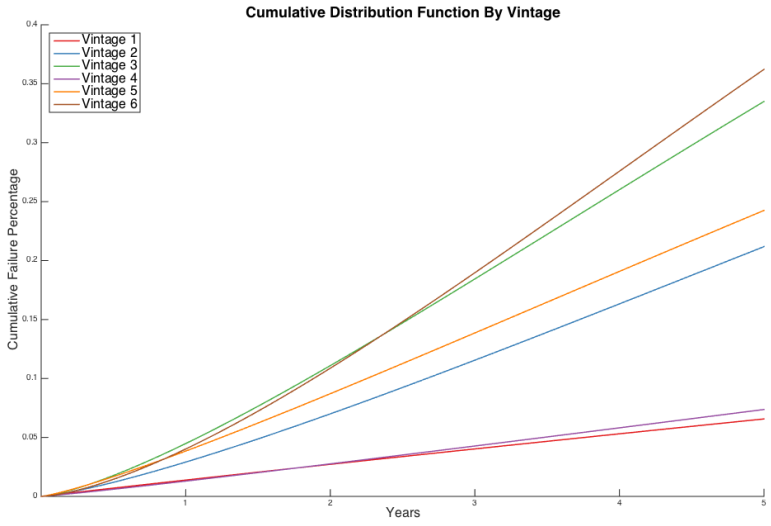


Figure 30. 5 year projected CDF of the Vintage model by vintage.

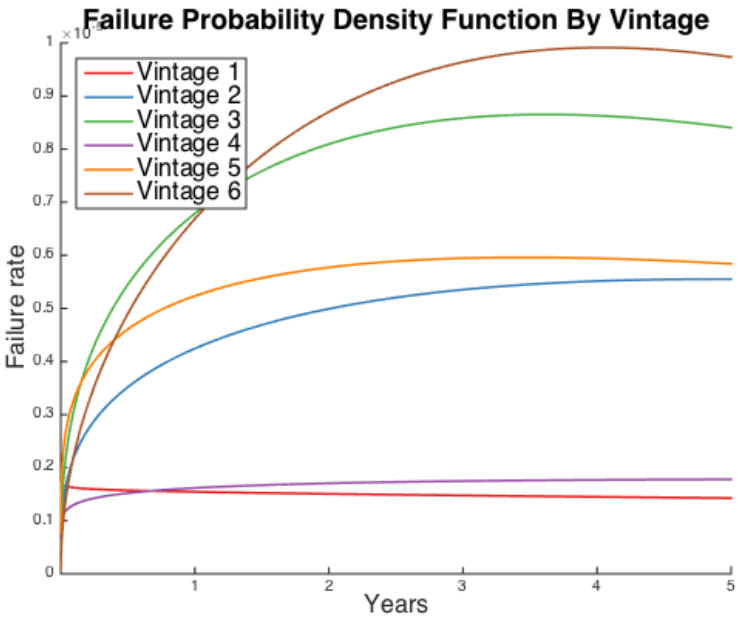


Figure 31. 5 year PDF of the CFDR model by vintage.

is a maximum of 4608 disks at a time during a simulation. Six new disk types were created corresponding to the six vintage types. These disks have the same 2 TB capacity and read and write speed as the previous disks. However, each disk type belongs to a separate disk model class that identifies its vintage. Since no information was present about the makeup of the disk vintages within overall populations, the configuration divides the disk population evenly among the six disk vintages. Each vintage will have 768 disks at a time within the system. Since a depot contains 32 disks and each bus supports 16 disks, it is not possible to evenly distribute the six vintages amount the depots and buses. Therefore, each depot bus will have a base of two of each vintage and an additional four disks of four out of the six vintages. When a disk fails or reaches its end of life period of 5 years, it is replaced with a disk of the same vintage and configuration. Thus the simulation will maintain an equal distribution of the six vintages.

For each vintage, the model creates disk profiles that determine when a particular disk for that vintage will fail. When a new disk is created, the Weibull vintage failure model uses a probability generator for that vintage to generate a failure time. If the failure time is within the 5-year lifetime of the disk, then it creates a failure event for the disk at that time. The probability failure generator for each vintage is generated from a lookup table of the inverse of the Weibull CDF, where the input key is a uniform probability from 0 to 1 and the output value is the time of failure. These lookup tables allow the failure model to transform a uniform distribution to the approximate failure distribution of the vintage.

Validation

In order to verify the failure model, the model was run on a million disks of each vintage. The cumulative failure percentage was calculated from the disks' failure times and compared to the failure Weibull parameters provided by Elerath. The following figure, Figure 33, provides the result of the comparison for each vintage. The Weibull failure CDF, shown by a black dashed line, is projected over the empirical CDF computed from the simulated disks' failure points with the corresponding color of the vintage.

As illustrated in the figures, the model's empirical CDF's almost exactly matches the Weibull CDF distributions with only a small amount variation due to random probability and sampling size. Figure 32 has the CDF for the entire simulated population over the course of 5 years.

Topology

REDDnet

There were several reasons for choosing the REDDnet topology. There are essentially an infinite number of topologies and in order to efficiently perform the study as many variables as possible needed to be locked down. REDDnet is the largest deployment of Logistical Networking, spanning the continental United States and CERN in Europe. It provides a realistic use case in terms of complexity with large organizations using it for data distribution. The network

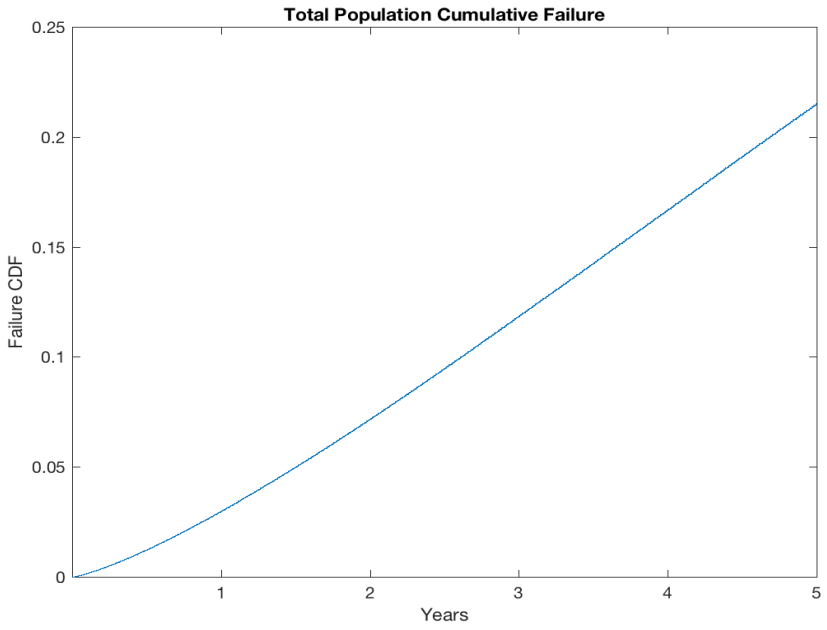


Figure 32. Vintage model total population CDF bounds.

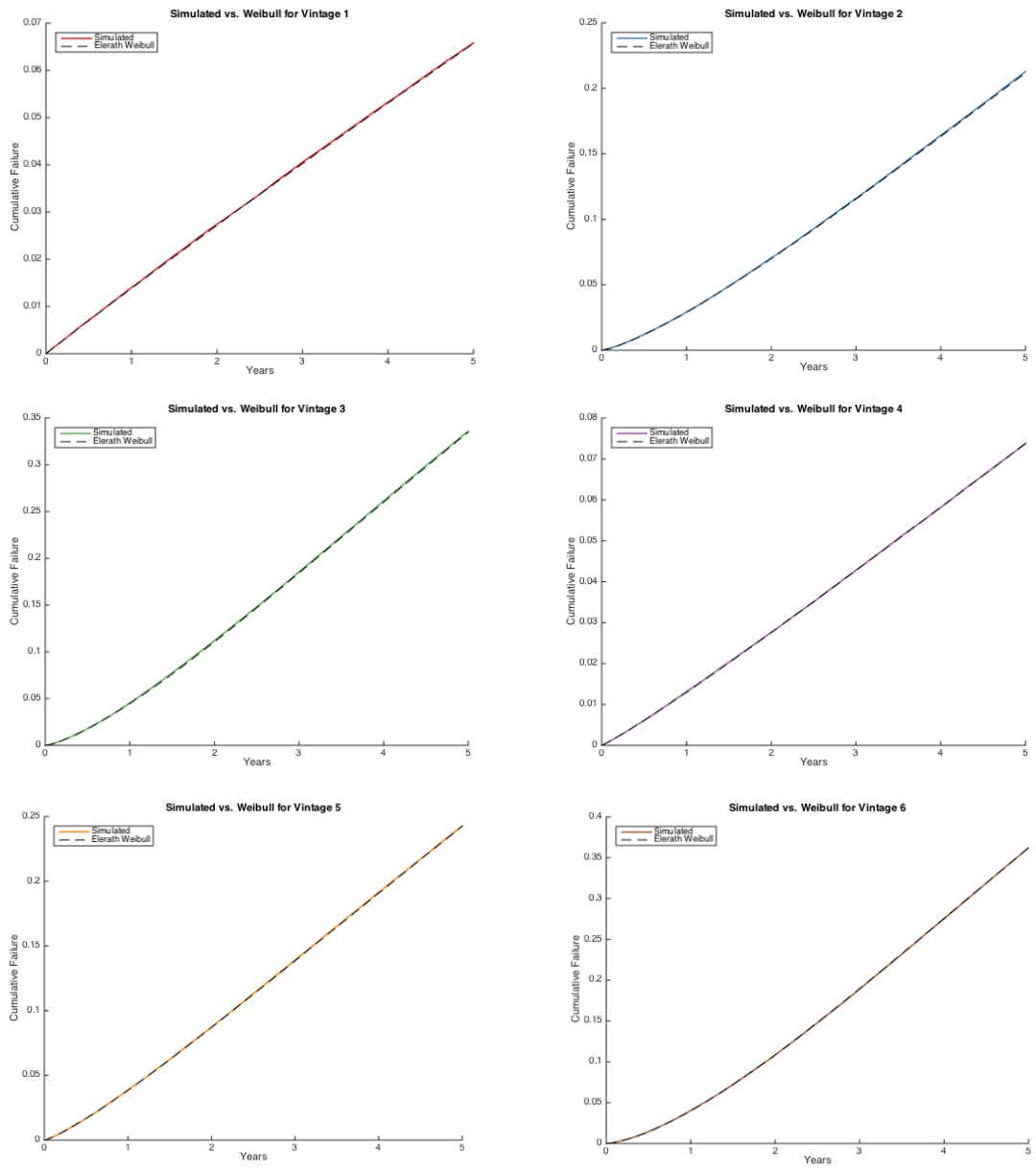


Figure 33. Vintage model simulated failures vs. Weibull CDF.

backbone of the deployment is interconnected by the fastest wide area research networks including Internet2 and National Lambda Rail, NLR. Furthermore, personal connections to many of the site and network administrators aided in the data collection processed required for the simulation.

Routes

Modeling the topology of REDDnet requires establishing accurate routing information across the various networks that interconnect the sites. To this end, a script based on the traceroute utility was created and run for a period of time between all of the depot sites to all of other depot sites forming an n-by-n routing matrix. This matrix was stored in an SQL database using two tables: a routing table and a segment table. In addition to the name of each network segment, the segment table also includes the IP and autonomous system information. This information is presented in Table 5.

Once the information was collected it was refined and tuned. First, all hops and routes pertaining to SDSC were removed since SDSC has ended their participation in the REDDnet project, leaving the topology with nine other sites. The unused hops and routes were excised. The intra-site hops and routes are merged into a single per site backplane that has 10 Gb/s bandwidth capacity and is fully accessible to the depots. Doing so, greatly simplifying the routing topology by removing over half of the hops collected and most of the missing hop information. Most site administrators are, for good reason, protective with information regarding their internal networks making acquiring traffic data beyond their site gateways difficult due to ICMP and UDP traceroute blocking. By consolidating the individual site segments, the processor and memory costs for simulating the full network are also reduced.

Links

Aside from implementing and running all of the simulations, the most laborious part of the process was collecting the pertinent traffic data for the network segments from the heterogeneous network organizations involved. Some of the organizations, like Internet2 and NLR, make traffic information publicly available over the Web to anyone or by a series of email correspondence while others are more protective of this information and require signed contracts before relinquishing the relevant data. All of the traffic data provided is critically important to this research as it presents an honest depiction of IO traffic that the simulator's Logistical Networking data flow would have to compete against in real-time. While storage failure and errors are the principle causes of data loss, IO can limit adequate preventive and recovery techniques to storage failure to stop data loss. As such, the author of this dissertation is extremely grateful for all of the assistance and patience showed by the staff and network administrators that aided in obtaining the information.

Table 5. Network Segments.

Network Segment	IP	Data	Organization	Gb/s
Caltech		Yes	Caltech	10
hpr-lax-hpr--i2-newnet.cenic.net	137.164.26.133	Yes	CENIC	10
hpr-i2-newnet--lax-hpr.cenic.net	137.164.26.134	Yes	CENIC	10
hpr-lax-hpr--caltech-ul.cenic.net	137.164.26.161	Yes	CENIC	10
hpr-caltech-ul--lax-hpr.cenic.net	137.164.26.162	Yes	CENIC	10
hpr-lax-hpr2--nlr-pn.cenic.net	137.164.26.25	Yes	CENIC	10
hpr-nlr-pn--lax-hpr2.cenic.net	137.164.26.26	Yes	CENIC	10
lax-hpr2--ucsb-10ge.cenic.net	137.164.26.5	Yes	CENIC	10
ucsb--lax-hpr2-10ge.cenic.net	137.164.26.6	Yes	CENIC	10
jax-flrcore-7609-1-te23-1800.net.flrnet.org	198.32.155.193	Yes	FLR	10
tlh-flrcore-asr9010-1-te0103-1908.net.flrnet.org	108.59.26.242	Yes	FLR	10
con-ufl-gnv-renet-v1908.net.flrnet.org	108.59.26.243	Yes	FLR	10
tlh-flrcore-asr9010-1-te0102-513.net.flrnet.org	108.59.26.4	Yes	FLR	10
jax-flrcore-asr9010-1-te0103-1300.net.flrnet.org	108.59.26.8	Yes	FLR	10
prov-i2-atla-renet-v1300.net.flrnet.org	108.59.26.9	Yes	FLR	10
tlh-flrcore-asr9010-1-be1-1.net.flrnet.org	108.59.31.24	Yes	FLR	10
jax-flrcore-asr9010-1-be2-1.net.flrnet.org	108.59.31.25	Yes	FLR	10
prov-nlr-hous-renet-v513.net.flrnet.org	108.59.26.5	Yes	FLR	10
sox-to-internet2-chicago.sox.net	143.215.193.14	Yes	Sox	10
xe-0-0-3.1306.rtsw.star.net.futuregrid.org	149.165.144.53	Yes	Indiana University	10
ae-1.10.rtr.hous.net.internet2.edu	64.57.28.112	Yes	Internet2	10
ae-0.10.rtr.clev.net.internet2.edu	64.57.28.159	Yes	Internet2	10
ae-0.10.rtr.wash.net.internet2.edu	64.57.28.163	Yes	Internet2	10
ae-0.10.rtr.kans.net.internet2.edu	64.57.28.36	Yes	Internet2	10
ae-0.10.rtr.chic.net.internet2.edu	64.57.28.37	Yes	Internet2	10
ae-0.10.rtr.atla.net.internet2.edu	64.57.28.4	Yes	Internet2	10
ae-0.10.rtr.hous.net.internet2.edu	64.57.28.57	Yes	Internet2	10
ae-8.10.rtr.atla.net.internet2.edu	64.57.28.6	Yes	Internet2	10
xe-0-2-0.104.rtr.chic.net.internet2.edu	64.57.28.69	Yes	Internet2	10
ae-3.10.rtr.losa.net.internet2.edu	64.57.28.96	Yes	Internet2	10
ae-2.10.rtr.hous.net.internet2.edu	64.57.28.97	Yes	Internet2	10
tx-bb-i2-hstn.tx-learn.net	74.200.187.26	Yes	LEARN	10
setg-i2-hstn.tx-learn.net	74.200.187.30	Yes	LEARN	10
rt1-hardy-hstn-xe-0-1-0-3018.tx-learn.net	74.200.187.6	Yes	LEARN	10
hstn-hstn-nlr-layer3.tx-learn.net	74.200.188.33	Yes	LEARN	10
74.200.189.30	74.200.189.30	Yes	LEARN	10
74.200.189.34	74.200.189.34	Yes	LEARN	10
hstn-hstn-nlr-ge-0-0-0-0-layer3.tx-learn.net	74.200.188.34	Yes	LEARN	10
kans-i2-xe-0-1-3-3008.tx-learn.net	74.200.187.14	No	LEARN	10
hous-i2-xe-1-3-0-3018.tx-learn.net	74.200.187.5	No	LEARN	
74.200.188.49	74.200.188.49	No	LEARN	
74.200.188.50	74.200.188.50	No	LEARN	
rt1-hardy-hstn-xe-2-0-0-3001.tx-learn.net	74.200.187.29	Yes	LEARN	10
rt1-akard-dlls-xe-1-3-0-3008.tx-learn.net	74.200.187.13	Yes	LEARN	10
tx-bb-i2-dlls.tx-learn.net	74.200.187.50	Yes	LEARN	10

Table 5. Continued.

Network Segment	IP	Data	Organization	Gb/s
vlan-53.atla.layer2.nlr.net	216.24.186.54	Yes	NLR	10
vlan-53.jack.layer2.nlr.net	216.24.186.55	Yes	NLR	10
vlan-43.elpa.layer2.nlr.net	216.24.186.72	Yes	NLR	10
vlan-43.losa.layer2.nlr.net	216.24.186.73	Yes	NLR	10
vlan-47.elpa.layer2.nlr.net	216.24.186.74	Yes	NLR	10
vlan-47.hous.layer2.nlr.net	216.24.186.75	Yes	NLR	10
vlan-51.hous.layer2.nlr.net	216.24.186.78	Yes	NLR	10
vlan-51.jack.layer2.nlr.net	216.24.186.79	Yes	NLR	10
vlan-42.losa.layer2.nlr.net	216.24.186.82	Yes	NLR	10
vlan-40.sunn.layer2.nlr.net	216.24.186.93	Yes	NLR	10
216.24.184.14	216.24.184.14	No	NLR	10
TACC-FutureGrid.gw.xsede.org	198.17.196.143	Yes	NCSA	20
setg-i2-gw.tx-bb.net	192.88.12.142	Yes	noc.gigapop.gen.tx.us	10
caltechhep-1-is-jmb-776.lsanca.pacificwave.net	207.231.241.134	Yes	Pacific Wave	10
nlr-1-lo-jmb-706.sttlwa.pacificwave.net	207.231.240.14	Yes	Pacific Wave	10
198.32.236.50	198.32.236.50	Yes	Rice University	10
198.32.236.65	198.32.236.65	Yes	Rice University	10
198.32.236.49	198.32.236.49	No		
198.32.236.67	198.32.236.67	No		
SFASU		Yes	SFASU	10
slr-to-nlr.sox.net	143.215.193.6	Yes	SOX	10
dallas-stl-atl.sox.net	143.215.194.130	Yes	SOX	10
nashville-ornl-atl.sox.net	143.215.194.138	Yes	SOX	10
nashville-sl-dallas.sox.net	143.215.194.154	Yes	SOX	10
dallas-sl-nashville.sox.net	143.215.194.153	Yes	SOX	10
nlr-to-slr.sox.net	143.215.193.5	Yes	SOX	10
atl-ornl-nashville.sox.net	143.215.194.137	Yes	SOX	10
TACC		No	TACC	
UCSB		Yes	UCSB	10
UFL		Yes	UFL	10
cit-vl12-e600.ultralight.org	192.84.86.225	Yes	UltraLight	10
cit-vl12-la7606.ultralight.org	192.84.86.226	Yes	UltraLight	10
cit-gre101-tocit-la7606.43.32.198.in-addr.arpa	198.32.43.217	Yes	Ultralight	10
cit-gre101-tocern-r04gva.43.32.198.in-addr.arpa	198.32.43.218	Yes	Ultralight	10
UMich		Yes	Umich	10
aust-utnoc-core-ge-0-0-0-744.tx-bb.net	192.88.12.5	Yes	University of Texas System	40

Table 5. Continued.

Network Segment	IP	Data	Organization	Gb/s
aust-utnoc-core-ge-5-0-0-706.tx-bb.net	192.88.12.50	Yes	University of Texas System	40
aust-utnoc-core-ge-6-0-0-521.tx-bb.net	192.88.12.229	Yes	University of Texas System	40
hstn-lvl3-core-ge-5-0-0-706.tx-bb.net	192.88.12.49	Yes	University of Texas System	10
192.88.12.230	192.88.12.230	No		10
CHI-GVA2.uslhcnet.org	192.65.196.226	Yes	USLHCNet	10
e600chi-vl55.uslhcnet.org	192.65.196.245	Yes	USLHCNet	10
sox-atlanta1.ns.utk.edu	160.36.128.157	Yes	UTK	10
sox-atlanta2.ns.utk.edu	160.36.128.158	Yes	UTK	10
UTK		Yes	UTK	10
Vandy		Yes	Vanderbilt	10

The following sections give a brief and as openly as possible description of the process via which the traffic data was obtained. Figure 34 provides the period over which the data was collected.

Internet2

Internet2 and NLR were collectively the two most significant networks used by REDDnet. Almost all inter-site routes relied upon Internet2 and NLR segments for at least a portion their paths. For Internet2, the traffic data was publicly available via the GlobalNoc software at [67]. After deciphering the naming conventions and decoding the HTTP protocol for accessing the data directly, a script was wrote that would automatically download the data sets for each network segment bypassing the need for a browser. As each network segment is a VLAN over a physical link obtaining the maximum bandwidth was difficult, but Internet2 publishes their router and VLAN configuration at [68] which greatly simplified this process and resolved a few naming convention issues that arose.

NLR

Like Internet2, the National LambdaRail, NLR, published their network information publicly at [69] using RRD tools and Cricket software for data collection and graphically display. Unfortunately, access to the raw numeric data was not possible and had to be reconstructed using the FindGraph utility to extract approximate the data points from the graphs. The work was more tedious and less precise; however, the process of obtaining the graphs themselves was capable of being automated via a script. It should be mentioned that NLR ceased operations in 2014.

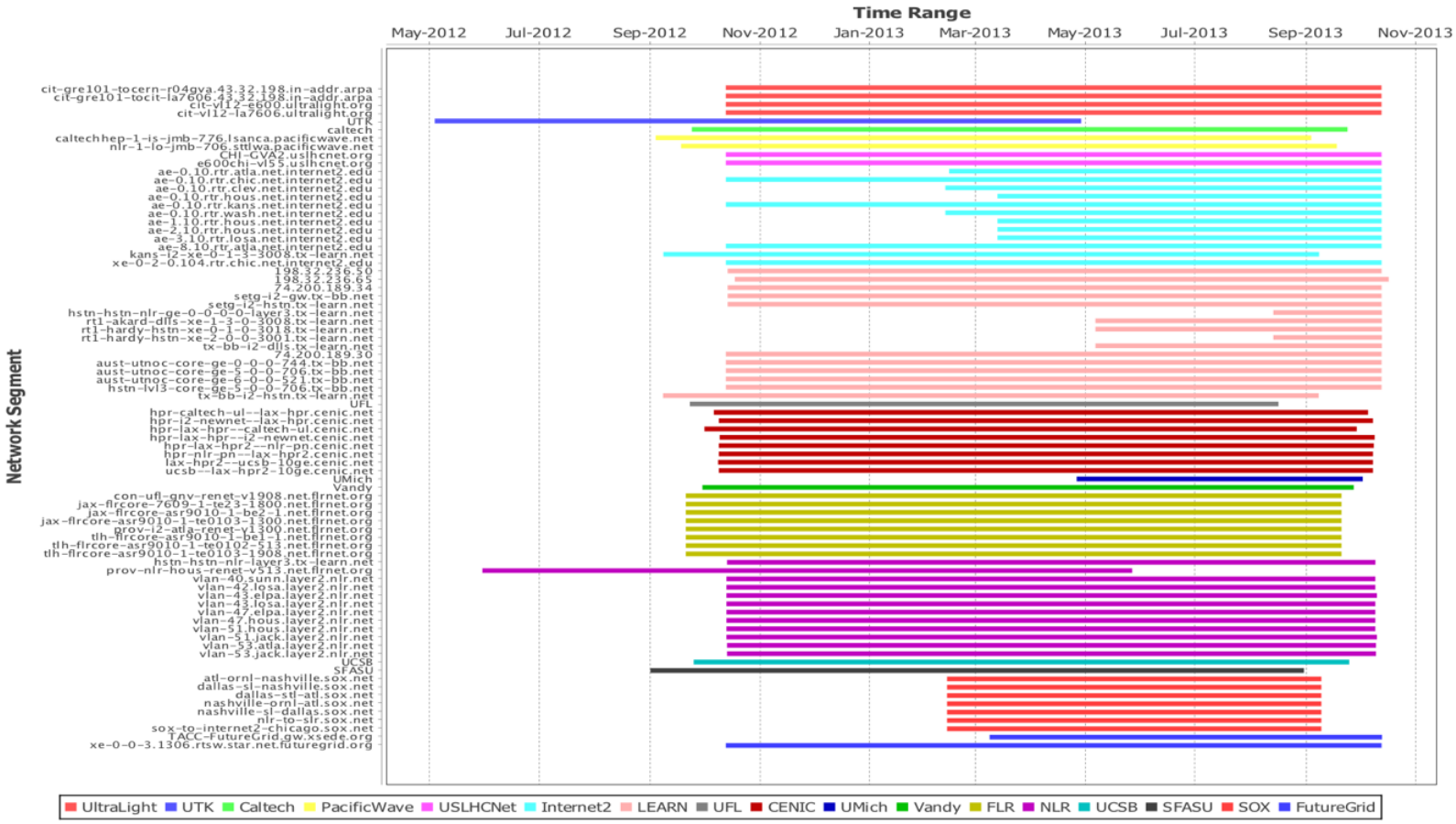


Figure 34. Link data collection time frame.

CENIC

CENIC, the Corporation for Education Network Initiatives in California, was an important network for access to sites in California. The process for obtaining the data from CENIC was similar to that of NLR since CENIC was chosen to be the NOC administrators for NLR before its shutdown. The bandwidth data was collected by CENIC using the RRD tools and published with Cricket software at [70]. The same NLR script to collect the network segment graphs was reused, and FindGraph extracted the data points. In cases, where several of their network segment names were not listed, the Intermapper software at [71] was used along with some detective work to determine the corresponding records.

UltraLight and USLHCNET

The UltraLight and USLHCNET, European Organization for Nuclear Research American LHC Network, networks were utilized solely for access to and from the CERN depots. The data was obtained via MonALISA and published for access at [72]. By writing a small script, the information was retrieved in CSV format for a period of one year.

FutureGrid

FutureGrid was an experimental test bed for the development of computer science applications that ended in 2014. It assisted in connecting the various geographically diverse clusters of the XSEDE project together. For the purpose of this simulation, it owned two of the network segments that connected UFL to TACC. Traffic information was publicly provided by the GlobalNoc software at [73], enabling the same general procedure as for Internet2.

Pacific Wave (Pacific Northwest GigaPop)

The Pacific Wave network, part of the Pacific Northwest GigaPop, described at [74], provided links in the routes between UTK, Vanderbilt and Caltech to UMich and CERN. While much of the configuration information of the Pacific Wave network was available on their website, the traffic information had to be obtained via a series of email correspondence with the NOC admins. After which they provided several raw RRD data files to sift through for the required information.

SOX and SLR

The SOX, Southern Crossroads, and SLR, Southern Light Rail, network serve to interconnect UFL, UTK and Vanderbilt University as well as connect these three institutions to the larger networks like Internet2 and NLR [75]. Much of the incoming and outgoing traffic from these sites traversed several SOX network segments. The required information for the configuration and traffic for SOX was obtained as a set of XHTML tables via email correspondence with the NOC admins. These tables were relatively simple to parse and manipulate for data extraction.

LEARN and the Texas Gigapop

LEARN, Lonestar Education and Research Network, and the Texas Gigapop connect several Texas institutions to the national research networks in particular Internet2. Unlike many of the other organizations, data collection involved utilizing a number of resources including [76], [77] and [78]. These websites provided the ability to collect the data graphically and in CSV format via Cacti in the most recent hourly, daily, weekly, monthly and yearly time frames. After downloading all of the available time frames for each network segment, the information was merged to provide the best possible traffic information. Not all network interfaces matched up with the ones in the traceroute obtained hostnames so LEARN's routerproxy at [79] was essential in obtaining the correct interface identities for data acquisition.

FLR

FLR, the Florida LambdaRail, interconnects UFL across the state of Florida and to the national research networks. Some the required configuration information was at [80] while the traffic data was provided by FLR NOC admins as standard RRD data files. Due to a research policy agreement between FLR and myself, no information about their networks besides the network segments involved will be disclosed.

UCSB

RRD traffic and error data for the UCSB's border router was generously provided by the network administrators, in particular, Kevin Schmidt. Further information was also available at [81].

Caltech

Bandwidth traffic data between Caltech and the CENIC network was provided by Anthony Pisano and Caltech IMSS - Voice & Data Networks Group as a CSV file for the relevant VLANs.

UFL

The University of Florida's border gateway traffic for one year was obtained from [82] as a graph for the primary link to the FLR network. The information on the traffic graph was digitalized using the FindGraph utility into a CSV file for processing.

Vanderbilt

Having worked previously at the HPC center at Vanderbilt University, I was able to use that relationship to acquire traffic graphs for the period of one year for the site's connection to the SOX network. As with the other traffic graphs, the FindGraph utility was used to digitalize the information into CSV format.

UMich

Traffic data for the University of Michigan was granted from the NOC via CSV download from Cacti with the assistance of Robert Ball and Shawn McKee.

SFASU

For Stephen F. Austin State University, their traffic data was provided as a graph from Michael Coffee, manager of Network Services for the campus. The FindGraph utility was used to digitalize the information into a CSV format.

UTK

For the University of Tennessee, a set border gateway PDF graphs were acquired via email correspondence with the network administrators. Then digitalized using FindGraph to obtain the plot points.

Bandwidth

Spline interpolation was carried out on the sample points obtained for traffic flow for each of the network segments. Because of the wide disparage of bandwidth time resolution, ranging from a 30 seconds to a day, and the different recorded timeframe periods, it was necessary to refine the selection interval. Only the last year of data was selected for processing, from Oct 9th, 2012 to Oct 9th, 2013. A standardized interval of time of five minutes was also chosen for the consecutive time points with spline interpolation used to produce the intervals. This standardization between network hops reduced the amount network configuration information and network simulation processing.

Performing spline interpolation and using lookup tables for competing traffic undoubtedly reduced the traffic accuracy for some of the network segments; however, given that much of the data was obtained via sampling and pulled from digitalizing graphs there was already potential sources of error. It is also important to note that a high degree of accuracy of network behavior is not required for the results of these simulations. All the runs of the simulator will be using same network behavior, thus each policy will be penalized or benefited in the same way. No network patterns look the same twice, thus in the real world, these policies would have to contend with vastly different behaviors over the course of 100 years. The goal of incorporating traffic data is to simply create a more plausible network environment in which the policies can run.

The traffic data was incorporated by taking the maximum bandwidth for each network segment and subtracting this number to produce the available bandwidth capacity at a hop for each time interval. This bandwidth capacity is stored in a lookup table per network segment by time. When a network transfer event happens, the network segment consults its available bandwidth table with the current time modulo the period of 1 year and uses that value for its transfer bandwidth.

Data Dispatcher Policies

Policy framework

The data dispatcher policy framework provides six optional event based callback routines that a policy can choose to implement. Each policy implementation registers these callbacks for the events it wishes to handle. When invoked each registered callback is supplied a set of arguments specific to understanding the event that triggered it. In addition to these arguments, each callback is given access to the database backend for LoDN so that it can perform information queries when necessary. In response to being invoked, each callback method is allowed to return a collection of data dispatcher operations to be carried out: allocate, replicate and delete. Each of these operations is executed by their policy assigned priority. An operation upon completion can perform its own callback back to the policy for evaluation. The policy can then, in turn, generate more operations to be executed from the initial operation. Through nesting of operations and callbacks, the policy can support complex operation chaining. The ordering of the handling of certain simultaneous events is important to lessen the chances of endangering data or using too many resources. Policy implementations are guaranteed the ordering of certain events occurring at the same time. All disk loss events will be handled before any allocation loss events, lessening any implementation complexities that policies might have for tracking how allocations were lost. To aid in preventing policies from over replicating when an allocation is lost or being migrated, an allocation import event will also occur before any allocation loss events

As stated there are six event callbacks: new disk event, disk failed event, disk diagnostics event, allocation request event, new allocation event and allocation loss event. The first three events handle disk related occurrences, while the latter three events are for single allocation events. Each of these callbacks is described in the next several subsections and within the subsequent policy implementation sections.

New Disk Event

The policy callback for this event is invoked when a new storage disk is registered with LoDN. This event occurs when LoDN has completed a depot status query that is configured to occur on a daily basis. This callback will be triggered within 24 hours of a new disk entering the simulation. The callback is provided with the identity of the disk and the depot, as well as, the age, size, free space, family and model for the disk. In the case, where the disk is replacing a set of previous disks that have reached their maximum age, the callback is also given a list of identities for the disks being replaced.

Disk Failed Event

The callback for this event is invoked when a storage failure has been detected by LoDN. Like the new disk event, this event occurs when LoDN has

completed one its depot status queries. This callback will be triggered within 24 hours of a disk failure according the configuration currently used. The callback is given the identity of the disk that that failed and the disk's storage capacity. Even though disk failure causes replica loss, it is not necessary to handle individual allocation lose at this step as there is an allocation lost event for every allocation on the disk that will be called following this event. The main purpose of this callback is so the policy mechanism can update any internal bookkeeping, such as removal from a reliability ranking table, for the disk that exists beyond what LoDN does.

Disk Diagnostics Event

A recurring event for each disk that gets triggered periodically every 30 days for the policy to evaluate the status of a disk from the time the disk is registered with LoDN. The data dispatcher invokes the callback with the identity of the disk, current age, and if SMART errors are enabled, a time series of the errors by type. Typically, this is the routine where a policy updates its reliability metrics for each disk and data object and invokes data dispatcher operations in response to these changes.

Allocation Request Event

This callback is activated when a client requests an allocation in order to perform an initial upload of a data object. The callback method has the latitude and longitude of the source client, the requested allocation size and a collection of callbacks to notify the client of an allocation once an allocate operation has completed. Disk and network metric aware policies will try to find the nearest and most reliable disk as this is a time when data objects are at a high risk of loss.

New Allocation Event

Callback triggered when a filled allocation, i.e. a data object replica, is imported into LoDN from a replication operation by LoDN or an upload by a client. The allocation and related information such as size and the identity of the data object are given to the callback function. A policy mechanism can use this event to check that a data object has enough replicas and reliability to meet the policy requirements. The callback can issue more operations for either performing additional replication or deleting existing allocations.

Allocation Loss Event

When an allocation is lost due either to either a disk failure or an allocation being deleted at the behest of the policy, this callback is invoked. Depending on how the allocation is lost determines the time scale at which the policy will be notified via this callback. For general disk failure, it may take up to 24 hours since LoDN is not informed until it performs its daily depot probe. When explicitly deleted by LoDN or the policy, the policy will be altered nearly instantaneously

after the deletion operation has completed. The callback has the identity and size of the allocation and the identity of the data object that resided on the allocation. As with new allocation events, the policy implementation will typically check that the data object still has enough replicas and reliability.

Simple

The simple policy manages allocating on behalf of the client and the initial distribution and replication of the data objects. The configuration for the policy takes only one parameter that specifies the initial number of replicas to create for each data object. The policy is "simple" in that other than being disk location aware, it has no knowledge about the state of the disk system and does not perform replication to replace lost replicas. Once the initial number of replicas have been distributed, the policy will no longer perform any actions for the data object. The purpose of the policy was to test the policy framework and to provide a baseline of as to how vulnerable a data object is by just relying on disk storage to be failure free.

The policy implements the following four callbacks:

- New Disk Event and Disk Failed Event

  The policy does not generate any operations in response to these events but does use the events to update location bookkeeping for the disks by either inserting or removing disks from site tables it maintains.

- Allocation Request Event

  The policy returns allocations to the clients that are as close as possible, as determined by the Haversine distance formula in Equation 3, to the data source.

Equation 3. Haversine distance equation.

$$2r\sin^{-1}\left(\sqrt{\sin\left(\frac{\varphi_2-\varphi_1}{2}\right)^2+\cos\varphi_1\cos\varphi_2\sin\left(\frac{\lambda_2-\lambda_1}{2}\right)^2}\right)$$

- New Allocation Event

  When a new replica is created, the policy checks the replication count for the data object. If the data object has less than the number of replicas specified, the policy will return an operation to produce a new replica. The preference for the location of the new replica will be for a site where the data object does not currently reside. If there is no available storage at such a site,

then the policy will try to obtain an allocation on a depot that is not occupied by the data object. Failing that, the policy will at last resort to any unoccupied disk. There is no point in creating multiple replicas on the same disk since this will not boost data object reliability.

All other events are ignored.

Reactive

The reactive policy extends the simple policy by "reacting" to allocation loss and the arrival of new disks in the system. Like the simple policy, it takes one parameter which is the number of replicas to maintain for each data object. When a replica is lost, the policy will determine if a new replica is required for the data object to restore the replica count. If so, it will generate replication operations to replace the lost replica. The policy prioritizes operations for LoDN to execute by the number of replicas that exist for the data object. The fewer the replicas, the more important the policy ranks replication operation for the data object. Deletion operations are thus the lowest prioritized operation since they have extra replicas and have the least chance of loss.

The reactive policy modifies the three following methods from the simple policy:

- New Disk Event

  When a new disk is registered with LoDN, the policy searches for any data objects that have an insufficient number of replicas and will attempt to create a new replica for each of those data objects on the disk. In the event of multiple disks arriving during this process, each disk will be utilized, and multiple replicas can be created across the new disks if needed to fulfill the replica count requirement.

- New Allocation Event and Allocation Loss Event

  The reactive policy responses to allocation loss and arrival by checking if the data object already has the required number of replicas. In instances where the data object needs further replication, the policy returns a collection of replication operations. The preference for replica placement is first to a site that has no replicas for the data object. If no such replication can be performed, then preference goes to any depot without a replica and finally resorts to any disk without a replica before giving up. If a data object has too many replicas, then the policy will create operations to remove the additional replicas. To reduce the possibility of accidental over deletion due to concurrent disk failure for other replicas, the policy schedules only one deletion operation per data object at a time.

The following callbacks are unaltered from the simple policy:

- Allocation Request Event

This policy best comprises what is currently done in response to replica loss in Logistical Networking.

Proactive

The following three policies; SMART Error, CFDR Age and Elerath Vintage; extend the functionality of the reactive policy by adding different disk reliability-aware metrics and proactive replication and migration. Instead of maintaining a constant replica count, the policies focus on keeping the collective reliability of a data object above the minimum specified. These policies take two parameters: the minimum number of replicas and the minimum reliability or health of a data object to sustain. Even though the policies have different methods of calculating the reliability of data objects, reflecting the metric they track, they operate similarly. In addition to the callback procedures implemented previously, these policies implement callbacks for disk diagnostics and disk failed. The next several subsections describe the operation of the policies in detail.

SMART Error Proactive

The first proactive policy was for the two SMART error failure models based on SMART error findings in [39]. The policy takes two configuration parameters: a minimum replica count and a minimum health rating for each data object. In addition, the policy strives to distribute and maintain multi-site dispersion equal to the minimum replica count specified and to provide allocation and replicas based on the nearest replica neighbor to minimize bandwidth and latency. All operations generated by the policy are assigned a priority equal to the health of the data object at the time they are issued to the data dispatcher.

Of the three proactive policies, this policy has the simplest disk and data object reliability prediction scheme. The other policy implementations use short-term conditional reliability for rating an individual disk's health and then combine the reliability of the individual disks that a data object resides on into the reliability of the data object. Instead, a simpler health parameter was formulated that hopefully will capture enough of the state of the disk and data object to suffice. There were multiple reasons for doing so. This was the first attempt at creating a proactive policy; the other proactive policies benefitted from lessons learned during the development of this policy. Secondly, this policy was devised to handle both SMART error failure models which have different failure characteristics based on disk age and failure count. Finally, while there was suitable information for disk failure probabilities for disk age and SMART error counts, there was not enough information about how individual SMART error occurrences affected disk reliability under various circumstances. For instance, if a disk already has a SMART error then what does its failure probability look like if

it has another at three months later as opposed to if it experiences another SMART error five months later. Alternatively, what if a disk already has a scan error and then has a reallocation error. How does the timing of these two types of errors affect reliability? As a result, assigning a "hard" reliability metric is far more complicated.

For disks, their health is computed by using Equation 4 where s is the number of SMART errors that have occurred within the last nine months.

Equation 4. Disk Health Formula.

$$disk\ health = \frac{1}{2^s}$$

The health of disks that have never suffered a SMART error, or at least not within the last nine months, will have an estimated health rating of 1.0, meaning the least likely to fail. For all disks that have had errors within the time frame, their health exponentially decreases with the number of SMART errors.

The health of a data objects is an aggregation of the health of the disks that a data object replica resides on. Data object health is computed by taking the disk healths from Equation 4 and using them in Equation 5. This formulation

Equation 5. Data Object Health Formula

$$data\ object\ health = H_n$$

$$H_i = \begin{cases} H_{i-1} + disk\ health_i * \left(1 - \left(H_{i-1} - \lfloor H_{i-1} \rfloor\right)\right), & i > 0 \\ 0, & i = 0 \end{cases}$$

where n is the number of disks and $disk\ health_i \geq disk\ health_{i-1}$

allows for the health of the data object to be driven by the number of the replicas that reside on error-less disks and then for a value between 0 to 1, exclusive, to be appended that accounts for those replicas that reside on disks with recent SMART errors. Applying an iterative weight to the health of the disks, ensures that no set of disks with errors will equal the reliability of one error-free disk, while still giving the data object's health some credit for each disk. For example, a data object with replicas on two disks rated 1.0 and 0.5 (one error) will have a

health of 1.5, and a data object with replicas on three disks of 1.0, 0.5 and 0.5 will have a health of 1.75.

This policy allows one replication or deletion operation per data object concurrently to prevent conflicts with either over replicating or deleting.

This policy implements all six of the policy framework callback procedures:

- New Disk Event

  Upon registration of a new disk with LoDN, this procedure computes the health of the disk using the formula in Equation 4 and then updates the health of those data objects that have replicas residing on the disk already. Unlike the corresponding method for the reactive policy, which was only concerned with the number of replicas for each data object, this method also finds all data objects that do not meet the specified minimum criteria for health. The policy issues replication operations to boost the health with priority given in ascending order of data object health until the storage space is exhausted or health requirements have been satisfied. This policy can handle multiple simultaneous new disks events without conflict or over replication.

- Disk Failed Event

  Rather simple callback for bookkeeping and cleaning data tables. The heavy lifting when a disk fails is performed by individual callouts to the allocation loss event handler for each data object.

- Disk Diagnostics Event

  The disk diagnostics callback is invoked on a disk under two conditions: whenever a new SMART error has been detected and periodically every 30 days of the disk's life. Since the data dispatcher probes the status of the disks daily, every SMART error occurrence will be handled within 24 hours. Every month, a disk's health is recomputed increasing its health metric as SMART error events age and eventually move beyond the nine-month window.

  When a disk's health metric changes, the status of all of the data objects that reside on the disk are recomputed accounting for the update. Each of these data objects is processed in ascending order of their new health rating, treating the lowest healthy data objects first. The policy generates replication operations for those data objects whose health is now too low (new SMART error detected). If a data object's health is now higher than required (fewer SMART errors), has more replicas than necessary and is

distributed across enough sites, then the policy will issue deletion directives for those replicas that are no longer needed.

- Allocation Request Event

  This callback returns allocations to the clients in descending order of health and distance ensuring that the initial replica will be on the most reliable disk possible with the lowest estimated latency to the client according to the Haversine distance equation in Equation 3.

- New Allocation and Allocation Loss Events

  The SMART error policy handles new and lost allocation events similarly. When a replica is lost or created, these procedures recompute the health of the affected data object. If the replica count is too low, then the policy issues replication operations with replica placement preferentially given to a site where a replica does not already exist. Failing the first choice, the policy will continue by attempting to replicate at an already occupied site, but on a non-occupied depot and finally falling back to using a non-occupied disk on an occupied depot before finally giving up.

  The preference for replica placement is nearly the reverse when the data object health is too low. Initial replication is first attempted at an already occupied depot, then at an occupied site and with an unoccupied depot and finally to any unoccupied site. The order preference of occupied depots and sites is by ascending depot and site level aggregate health for the data object. The justification for this sequence is that the data object already has enough copies and site distribution and only needs its health boosted. By initially choosing occupied depots and sites, the policy can reduce wide area bandwidth usage and the time it takes to produce a new replica. Furthermore, replicating at the lowest available health location allows the policy to preemptively maintain the number of copies and site distribution since that is the location most likely lose a replica.

  If the multiple site distribution is not maintained, but both replica count and health are sufficient, then replication is attempted to a non-occupied site only.

  When a data object exceeds the requisite replica count, health and site distribution, the policy will scan the list of replicas and attempt to schedule deletion operations for the least reliable allocations that can be removed will maintaining the three invariant requirements.

CDFR Age Aware Proactive

The CDFR age aware proactive policy was designed for the CFDR age failure model based on the findings in [10] and [63]. This policy has two configuration parameters: a minimum replica count and a minimum reliability setting for each data object. The reliability metric of this policy is based on the age of the disks that the data objects reside on. Like the preceding proactive policy, the policy attempts to distribute and maintain a data object across a number of disparate sites equal to the requested replica count. When possible the policy will seek to minimize the distance traveled by replicas by using the nearest suitable neighbor, whether that be a disk on the same depot, a depot at the same site or just a nearby site. The policy will also attempt to minimize storage utilization by deleting any replicas that are no longer required for the replica count and reliability criteria. All operations created by this policy are assigned a priority matching the reliability of the data object. Doing so allows those data objects at highest risk, the quickest handling by the data dispatcher.

This policy used a more traditional means for computing the reliability of disks and data objects than the health metric from SMART error policy. Disk reliability is based on the values from the HPC 1 data sets from the CDFR. These values are partitioned into 30-day intervals and stored in a lookup table that is indexed by the age of the disk divided by the 30-day time period. The system-wide reliability of a data object is treated as the parallel reliability of the data object replicas and thus the parallel reliability of the disks that the data object resides on as shown in Equation 6.

Equation 6. System-wide data object reliability.

$$R_{data\ object} = 1 - \prod_{i=1}^{N} \left(1 - R_{disk_i}\right)$$

N is the number disks and R_{disk_i} is the reliability of disk i.

The CFDR age policy implements the six callbacks of the policy framework similar to the SMART Error policy. The callback procedures for the policy are described below:

- New Disk Event

  When a new set of disks is recognized by the policy, the disks' reliability is computed from their age as described above. In the case of disk remapping, when a new disk replaces an expired

one, the data objects with replicas on the replacement disk have their reliabilities recomputed.

For all of the data objects with affected replicas that now have a reliability score below the acceptable threshold and any outstanding data objects awaiting more replicas, the policy issues replication operations using the current set of new disks. The policy will continue to create more replicas until either all of the data objects have sufficient reliability, been replicated to all of the new disks or all of the new storage space is exhausted. These replication operations are updatable to include new disks that arrive after their issuance.

- Disk Failed Event

  When the policy becomes aware of disk failure, this callback method updates bookkeeping on the disks and allows the allocation loss handler to deal with the individual replica loss events.

- Disk Diagnostics Event

  Every day those disks that have reached one of their 30-day age increments are processed by this procedure. The disk is analyzed and has its reliability retabulated base upon its current age. All data objects that have replicas residing on one or more of these disks have their reliability recomputed based on the changed underlying disk reliabilities.

  Data objects whose reliability is too low have replication operations issued with priority based on the data object's updated reliability. To boost reliability as soon as possible and maintain the multisite distribution, replication is performed in ascending order of site and depot reliability for the data object. The destination disks chosen are those with high enough reliability and descending proximity. Typically, this results in intra-depot disk copying first, followed intra-site copying and finally inter-site copying. In the event of a new destination allocation being successfully created, but the current source allocation (replica) being lost, the origin allocation is changed to the next closest to the destination.

  Data objects that now have reliability exceeding that of the requirements are processed to see if any of their replica allocations can be released. This involves verifying that such action would still maintain the reliability, replica count and site distribution invariants.

- Allocation Request Event

  This callback generates allocations on behalf of the source clients based on their requests in order of descending disk reliability, ascending proximity and remaining storage capacity.

Doing so gives the initial replica the highest reliability until the data dispatcher has the opportunity to replicate and safeguard the data object.

- New Allocation and Allocation Loss Events

  The two most complex procedures for the policy are for handling the arrival and loss of allocations. When an allocation is created or lost, the reliability and replica count of the associated data object are adjusted according to the data dispatcher knowledge of the environment.

  Regardless of the data object's current reliability, when the number of replicas is below the specified count, the policy invokes a replication operation to increase the replicas. To maintain multisite distribution, priority is first given to attempting inter-site replication before falling back to intra-site and ultimately intra-depot replication. At each stage of replication, the policy will always try to replicate to the disks with the perceived highest reliability available that meet the other selection criteria.

  If the number of replicas is sufficient, then the data object's reliability is checked to see if further replication is required. The general objective is to produce a new replica as close to an existing replica to minimize bandwidth and hasten the process. The replication preference is at the depot and site with the least reliability for the data object. This location is where a replica is mostly likely to be lost, so the policy preemptively replaces it. Additionally, the least reliable replica may now be able to be deleted by the policy after successful replication.

  If the data object has either too many replicas or greater reliability than is necessary, then this policy will attempt to remove a replica while ensuring that doing so would not jeopardize the data object. Deletion can only occur if the number of replicas, reliability and distribution are maintained at or above the specified levels.

Manufacturer, Model and Vintage Proactive (Elerath 6 Vintage)

This policy is designed to improve reliability and resource utilization for disks whose failure mode corresponds to the Manufacturer, Model and Vintage Model discussed previously. It takes two configuration parameters: a minimum replica count and a 2-month conditional reliability for each data object. In addition to improving the reliability of data objects, this policy also attempts to minimize storage and wide area bandwidth utilization. The policy releases allocations when they are no longer necessary for the replica count and achieving specified reliability. Except for when explicitly distributing the data objects across the required sites, the policy attempts to reduce wide and local area bandwidth by adding the proximity of existing replicas into consideration

when deciding where to replicate. The priority of the operations match the reliability of the data object; thus data objects needing more replicas are preferred over those that can have allocations released. Those data objects whose reliability is too low are favored over those that just need more replicas for the minimum count.

The reliability calculator is more refined than in the other proactive policies with the failure model based on a series of continuous Weibull distributions instead of a discrete piecewise distribution. A continuous failure distribution made it possible to use near term, conditional reliability as the driving metric. The reliability of a disk and a replica is the Weibull distribution from Equation 2 integrated into the generic formula for conditional reliability as shown in Equation 7 and Equation 8.

Equation 7. Conditional Reliability.The probability that a disk will survive to time T+ t given that it has already survived to time T.

$$\mathrm{R}(\mathrm{T},\mathrm{t}) = \frac{R(T+t)}{R(T)}$$

Equation 8. Weibull Conditional Reliability. β is the shape parameter and η is the scale parameter.

$$\begin{aligned}\mathrm{R}_{disk}(\mathrm{T},\mathrm{t}) &= \frac{e^{-((T+t)/\eta)^{\beta}}}{e^{-(t/\eta)^{\beta}}} \\ &= e^{-((t+T)/\eta)^{\beta} + (t/\eta)^{\beta}} \\ &= e^{\,t^{\beta}/\eta^{\beta} - (t+T)^{\beta}/\eta^{\beta}} \\ &= e^{\frac{t^{\beta}-(t+T)^{\beta}}{\eta^{\beta}}}\end{aligned}$$

The reliability of data objects is calculated using the parallel component reliability formula from Equation 6, replacing the generic R_{disk} with the Weibull conditional reliability for each disk that a data object resides on. Equation 9 gives the Weibull conditional reliability integrated into Equation 6

With a continuous distribution underlying the conditional reliability for the disks, there is a wide array of values for the incremental duration to be considered. For these simulations, a period of two months was chosen since that period is a small enough look ahead to be meaningful but long enough to allow the policy sufficient time to react. For the policy, there is a gamut of possible combinations for the minimum number of replicas and minimum 2-month conditional reliability of the data objects. As with the other policies, the storage capacity allows the number of replicas configured to be from 1 to 3. For the six vintages, the 2-month conditional reliability of the disks varies from 0.9784 to 0.9982 over the 5-year time frame. Naïvely setting the policy parameters for

Equation 9. Vintage System-wide Data Object Conditional Reliability.Reliability from time *T* to *T+t* where *N* is the number of disks, β_i is the shape parameter, η_i is the scale parameter and T_i' is the time adjusted age of the i-th disk.

$$R_{data\ object}(T,t) = 1 - \prod_{i=1}^{N}\left(1 - R_{disk_i}(T',t)\right)$$
$$= 1 - \prod_{i=1}^{N}\left(1 - e^{\frac{t^{\beta_i} - (t+T_i')^{\beta_i}}{\eta_i^{\beta_i}}}\right)$$

one to three copies and the conditional reliability of the data object from 0.9 to 0.99999999 to cover all possible ranges would generate 24 different permutations to study. Thankfully the number of permutations can be pruned by noting that setting the minimum conditional reliability below the minimum possible for the required number of copies serves no purpose. Likewise, setting the minimum reliability to or above the highest conditional reliability for minimum copies plus one as that reliability level would always require more copies than the minimum specified. This prunes the number of permutations to a more reasonable and manageable 13. All of the 3-replica count configurations can also be cut since the real interest is seeing if the 1 and 2-replica can outperform the 2 and 3-replica reactive policies. The final configuration of the 2-replica policy with a reliability of 0.99999999 can also be struck since analytical there would not be enough storage to provide that level of reliability. Table 6 provides the possible minimum and maximum reliability for the number of replicas and the

Table 6. Valid Vintage policy configuration parameters.

Replicas	**Data Object Reliability**		**Minimum Reliability**
	Min	Max	
1	0.9784	0.9982	0.99, 0.999, 0.9999, 0.99999
2	0.99953344	0.99999676	0.9999, 0.99999, 0.999999, 0.9999999, 0.99999999
3	0.999989922304	0.99999999417	.99999, .999999, .9999999, .99999999

reliability settings used along with that replica setting. Note that reliabilities in red have been excluded for reasons given above.

Like the other proactive policies, all six of the hooks in the policy framework have been implemented and are described below.

- New Disk Event

  When a disk is registered with LoDN, the conditional reliability of the disk is computed based on its current age and vintage for the next two months. If this new disk is a remap of a retired disk, then all of the data objects with replicas on the disk have their reliability re-evaluated. The policy will generate new replication operations for all of the data objects in the system with lower than specified reliability including those already on this disk. The policy continues until either all of the data objects have sufficient reliability or storage space is exhausted. It will also delete replicas for any data objects that are on this disk that no longer need as many replicas due to the remapping. As before this implementation allows for updates to the replication process to accommodate the arrival of new drives.

- Disk Failed Event

  For this policy, this procedure is a simple no-op with the procedure for allocation loss handling the resulting individual replica losses.

- Disk Diagnostics Event

  The associated procedure for this event recomputes the 2-month conditional reliability of the disks every 24 hours as the depots are probed by the data dispatcher. Once the reliability of all the disks and replicas have been tabulated, then an update on the reliability of the affected data objects is carried out. Those data objects that then have a conditional reliability too low have replication operations issued for them in priority of their reliability. When possible the policy will delete replicas for data objects that would still meet the number of copies, reliability and multisite distribution specified. As these data objects have the greatest reliability, the priority of these deletion operations are less than those of replication operations.

- Allocation Request Event

  The handler procedure for the allocation request callback provides allocations to the client choosing disks in the order of reliability, proximity and available storage. By favoring disk reliability and proximity, the policy provides the greatest chance of the single replica data object surviving long enough to be further replicated after being surrendered to the data dispatcher's care.

- New Allocation and Allocation Loss

  When allocations are gained or lost, the policy responds by

calculating the new reliability of the data object affected. It then moves through a series of stages to determine what and if any additional actions are required. If the data object has too few replicas, then the policy will issue inter-site replication operations to increase the count and distribution of the replicas. If the data object has enough replicas but needs further reliability, then the policy will respond by issuing proximity based replication, starting at the depot level, then the site level and finally between sites. When possible the policy will reduce the number of replicas a data object has provided that doing so does not violate the reliability or replica count of the data object. If any of the stages fails along the way due to insufficient storage space, the policy will search for data objects that have replicas in the locations under consideration and are eligible for deletion. If and when such a replica can be located, the policy will attempt to delete that allocation and use its storage space before continuing to the next stage.

Table 7 summarizes each of the policies with the policy frameworks hooks they implement.

Table 7. Policy hooks.

	Simple	Reactive	SMART	CFDR	Elerath
New Disk		✓	✓	✓	✓
Disk Failed			✓	✓	✓
Disk Diagnostics			✓	✓	✓
Allocation Request	✓	✓	✓	✓	✓
New Allocation	✓	✓	✓	✓	✓
Allocation Loss		✓	✓	✓	✓

How Analysis Was Performed

A target of between 100 to 500 simulations per configuration was chosen. The number of simulations for each configuration was dictated by the variability seen in the initial runs, the HPC resources remaining and the compute time taken for the particular configuration. As many simulations as possible were performed to cover the independent, random events for the Monte Carlo simulation methodology. The permutation of simulations covered each of the four failure

models: SMART error disk age, SMART error count, Age (CFDR) and Manufacturer, Model and Vintage (Elerath). Then for each model the gamut of simple, reactive and proactive policies was executed. For the simple and reactive policies, the only configurable parameter, replica count, was varied from 1 to 3 per data object. The number and specifications for the proactive policies varied as each of the proactive implementations was specific to a particular failure model. An exact run down of the configurations is provided in the following results chapter.

Data files

Each simulation produced multiple, simulation-domain specific database data stores for out-of-core storage. The simulation implementation kept them nearly consistent using write-back on the "dirty" objects when the various caches were full or extensive queries were needing to be executed that required too much memory. At the end of the simulation mission time or when a checkpoint operation is invoked, the simulation will dump the entire contents of the caches to the data stores. It then ensures that all state between the simulation and the data stores is consistent and complete in its totality. These data stores provide the raw results used for analyzing the outcome of the simulation run.

Each of the backend data stores are domain specific for each element of the simulation and are only accessible from within that particular component. This design prevents snooping and knowledge overlaps that could result in inaccurate states. The data object database holds the state of every data object including the current and lost replicas for the data objects. The replicas were subdivided based on their current status into separate tables for performance and storage space savings. The next database, disks, contains the state of all the drives that have ever been in the simulation as well as a table of all of the existing disk allocations. Allocation information upon disk failure was not retained to save storage space as the location information is available for post-run inspection in the data objects database. The network and bus databases hold traffic information for each network hop (segment) and depot bus. Most of the information for them was kept solely in memory since their instance count was low enough for the topology configuration not to require caching. The well-structured schemas of these data stores made writing analysis code succinctly easier. The schemas and important properties for these databases are described in Table 8.

There are several more data stores not used for analysis but for debugging and verification. The database events stores all of the events that have occurred including their time, type and details. The simulator database contains the initial configuration of the simulator to ensure that the description of the resulting tar file matches the experiment executed. While not a data store, per se, there was also a log file produced, called sim.log, that contains information about the simulation process itself including the time it took for execution and resources consumed. Finally, the LoDN database is a transient,

Table 8. Simulator data store schemes.

Data Objects		
Table	Property	Description
Data Objects	Data ID	The unique id of the data object.
	Source ID	The unique id of the client that generated the data object.
	Size	The size of the data object
	Date Created	The time of the data objects arrival in the simulation
	Date Lost	The time the data object was lost (no more replicas). Null value for present data objects
Replicas	Data ID	The unique id of the data object that the replica represents.
	Diskmap ID	The unique id of the disk that the replica resides on.
	Date Created	The time the replica was completed.
Lost Replicas	Data ID	The unique id of the data object that the replica represents.
	Diskmap ID	The unique id of the disk that the replica existed on.
	Date Created	The time the replica was created.
	Date Lost	The time the replica was lost.
Disks		
Disks	Disk ID	The unique id of the disk
	Creation Time	The time the disk arrived in the simulation.
	Failed Time	The time the disk failed.
	Amount Read	The amount of data that was read from the disk.
	Amount Wrote	The amount of data that was written to the disk.
Allocations	Disk ID	The unique id of the disk that the allocation resides on.
	Allocation ID	The unique id of the allocation.
	Amt Filled	The portion of the allocation that is filled.
	Data ID	The unique id of the data object that is stored in the allocation.
Network		
Hops	Hop ID	The unique id of the network hop
	Amt Transferred	The amount of data sent across the hop.
Bus		
Buses	Bus ID	The unique id of the bus.
	Amt Read	The amount of data read from the bus.
	Amt Wrote	The amount of data wrote across the bus.

out-of-core backing store for LoDN and the data dispatcher subsystem. It functions much like the metadata store of its real life counterpart and is accessible and extensible by the various policy implementations. As such it is not very useful for analysis since it contains no historical data and is a limited, outdated view of the system state by LoDN.

At the end of a simulation run, the data store files were collected and organized into a file system hierarchy. The records for a run where stored in a directory named using the configuration parameters and compressed into a tar file bearing the same name. In addition to the configuration parameters, the tar file was tagged with the UUID and timestamp of the simulation. These result tar balls were relocated into a directory structure that like the tar files were named after the configuration options. This simplified later retrieval for analysis.

Processing

Numerous simulations were removed due to software or hardware errors generating failures before completion. The most prevalent were internal JVM crashes, followed by database errors primarily in LoDN's backing store. These events lessened over time as updates were pushed out by the respective software developers. Failures were also triggered by the insufficient allocation of memory, CPU and IO when deploying new simulation configurations. With the introduction of a checkpointing implementation for the simulator, there were instances of simulations being corrupted due to errors in the checkpointing process as well as with resumption when validity checks failed. The issues with the checkpointing implementation were quickly resolved and any of the runs that used the implementation were rejected from inclusion in the final analysis. There was also a series of hardware of issues with ECC RAM errors and node crashes that caused simulations to fail. As the configurations became increasingly complex job especially for the proactive policy jobs, job cancellation due to timeouts became an increasing problem. Fortunately, the aforementioned ability to checkpoint handled these issues and allowed for configurations that would take weeks to compute. Finally, during the analysis, it was discovered that a few of the compressed results sets had become corrupted, so these were filtered from the analysis process. In all of these cases, every effort was made to verify that causes were not due to a particular simulation configuration or would cause results that would shew analysis later. With so many results sets used for the analysis and no correlation discovered between the type failures seen and the results to that point in the simulation, these errors are not believed to have influenced the outcomes.

Several of the configurations under study had fewer results sets than others due to a number of factors. Some of the configurations required significantly greater resources; memory, time and processor; than most of the rest so executing them was quite challenging. Additionally, available grant allocations on HPC resources were limited and exhausted over time. By the end of the data generation and collection period very few of the HPC resources were

still available for use, so the costlier simulations had reduced priority in favor of a broader spread of configurations.

The analysis of the results was performed in multiple stages. The initial result sets were too large to be analyzed all at once, requiring incremental processing and result refinement. The first tier of the analysis was done per simulation by custom software to extract the relevant information from the databases and process it into smaller, more manageable data files. The result files for each simulation were grouped into a directory matching the name and UUID for that simulation. These simulation result directories were aggregated together into population directories bearing the name of the simulation type. The generated result files are simple comma-separated value, CSV, files.

The processing is divided up into separate components tasked with computing the discrete aspects that are of interest to the research goals. The data survival and reliability of a policy is calculated from the data stored in the data objects database. Using the date lost and size fields, the analysis program tracks the number of lost data objects and magnitude of loss over time in ascending one-year increments.

The storage and replica usage over time is also taken from the data objects database by examining the arrival and loss times of the replicas. These events are sorted to create a timeline of all of the replicas for the active data objects and processed linearly. When a replica is created, the storage usage is incremented by the size of the replica and decremented when a replica is lost. Instead of using a single global tracker for the number of replicas, the replica is mapped to its corresponding data object and the replica counter for that data object is adjusted. When data objects are lost, i.e. no more replicas, the data objects are excluded from consideration in calculating replica counts. The storage usage and moving average replica count per data object are published in one-month resolution intervals.

Information regarding data flow is found by using the bus and network data stores to summarize the data flows into the following categories: wide area, local area, bus read and write, local area only, and bus-only traffic. The total wide area traffic was ascertained by recording the traffic from the single outgoing link for each site. The traffic from the incoming site backbone link was used as the proxy for the total local area traffic since all local area traffic is required to traverse it. The bus total was the summation of the amounts of data that traversed each bus in each direction, read and write. The local area only traffic, i.e. intra-site depot-to-depot copying, was computed by subtracting the total local area traffic by the total amount for the wide area. Likewise, the bus only traffic, i.e. intra-depot copying, was found by taking the total write based traffic and subtracting the total local area traffic. Write direction was used since read and write are mostly mirror images of each other except for when a transfer is cancelled due to disk errors.

Already presented in this dissertation were the results of the disk failure analysis. While not useful for the policy analysis, this information was used to

verify that the disks and allocations were lost according to the failure model and consistent with the research collected. This data principally relied upon the information in the disks database using each disk's creation and failed dates as well as any additional information about the particular failure model such SMART errors and vintage information.

The count and simulation run time was found from the sim.log files to provide perspective on the number of simulations and time spent producing the results.

The second tier of the analysis was done at the simulation population level. Unlike the first tier of analysis, this level utilizes prewritten software namely, MATLAB [66] and SciPy [83], to combine the results of each simulation via aggregate descriptive statistics. After the second tier, datasets were ready to begin comparing population results against one another. Python, in particular, matlibplot [84] and xlwt [85] packages, were used to produce the final result data files, graphs and spreadsheets. Excel was employed to assist with doing regression analysis on the data sets and create the tables for this dissertation.

Results

The results are organized and presented by the disk failure model and policy using these criteria.

Data survival and reliability.

The most critical criteria to consider is the long-term reliability and survival of the data objects under the policies' care. In addition to the ultimate 100-year reliability, the reliability results are published at ten-year intervals to gain a better perspective on the relationship between policies over their entire lifetimes. Also, regression analysis is utilized to understand the resulting loss distributions and the characteristics to make comparisons simpler.

Storage usage and number of replicas used.

The next most important facet is storage utilization since not only must the reliability improve but storage usage must be reduced to prove the validity of this dissertation's supposition. The entire lifetime of the storage utilization is presented and analyzed. The final usage values are less important than understanding the maximum and minimum requirements since any future storage system must be able to fulfill those requirements. Storage utilization is subjected to regression analysis in the different phases of simulations runs.

Closely related to storage utilization is the mean number of replicas per data object. This is different from storage utilization, though, as it only considers those data objects that survived to that time. Unlike storage utilization which should have a downward trend as data objects are lost, the number of replicas provides details into how the policy treats the individual remaining data objects throughout the simulation's life. The replica count distributions for some policies can be complex and challenging to compare. In those instances, the integral of

replica count with respect to time, replica-time, is provided to make the results more understandable.

Data flow

The next metric considered for each policy is the amount of traffic generated to maintain the data objects. Clearly, the less traffic, the better; however, the traffic is also analyzed and partitioned by the following types: bus-only, local area only and wide area traffic. Hence, this metric is a reflection of the number of intra-depot, intra-site and inter-site replications that had to be performed by each policy. The closer the data flow is kept to the originating source the better with the fewest resources involved.

NOMDL

In order to consider both the reliability and storage utilization results in a single space simultaneously, the NOrmalized Magnitude of Data Loss, NOMDL, will be employed with some minor modifications [57]. As defined in Equation 10 and Equation 11 below, the NOMDL creates a ratio between the data loss of the system and the total apparent storage available for use. The smaller the ratio, the better the policy is deemed. As the numerator, data loss, increases, the ratio worsens as it does when the denominator, the apparent storage, decreases. The first equation finds the magnitude of data loss at a given time *t* by computing the amount of data loss as the numerator and dividing by the number of simulations, N. The second equation takes the MDL and divides it by the apparent usable capacity of the storage system, D, to produce the NOMDL.

Equation 10. Magnitude of Data Loss. t is the time that has past within the system. N is the number of simulations run. I(F<t) evaluates to 1 when data loss occurs and 0 if not before time t. C is the size of the total data loss.

$$MDL_t = \sum_{i=1}^{N} \frac{I(F_i < t) * C_i}{N} \quad where \;\; I(F < t) = \begin{cases} 0, no\ data\ loss \\ 1, data\ loss \end{cases}$$

Equation 11. Normalized Magnitude of Data Loss. D is the apparent usable capacity of the storage system, i.e. the actual amount that can stored after techniques like erasure encoding and replication have been performed.

$$NOMDL_t = \frac{MDL_t}{D}$$

For this dissertation's analysis, the formulation for D has to be altered however. The metric was originally intended for storage systems using a fixed set of redundancy by either replication or erasure encoding. For instance, the usage capacity, D, for a 3-replica reactive system would be the total storage divided by three since every data object is thrice replicated. In the case of the proactive policies, the usable capacity will be more elastic as the number of replicas per data object will vary over time. Furthermore, the 3-replica storage system would be fully utilized with no spare storage slots. Since the total storage is static for all of the configurations performed, some configurations will have significant quantities of free space that would be available for other purposes. To account for these differences, the usage capacity, D, will be redefined to be the size of the data in the system and the total free storage at a given time t.

While there are multiple sets of the simple, reactive and proactive policies, key emphasis will be on comparisons between the 2 and 3-replica reactive policies and the proactive policies. The 2 and 3-replica reactive most resemble what LoDN currently does for data distribution and maintenance. The simple policies provide a baseline to determine what the data object decay rate would be without aid and repair and the minimum amount resources consumed to perform the initial upload and distribution.

Graphs and Tables

The results are summarized in each section with figures and tables. For the reliability and storage usage, the data is presented graphically as a time series of Tukey box plots with the median, first and third quartiles and 1.5 times the interquartile range for each time unit. When feasible, regression analysis is performed to improve comparison between the different policies. The data flow information is presented as stacked bar graphs with each policy bar being partitioned into segments by the type of data flow. The height of each bar segment indicates the amount of that particular traffic and the total height of the bar is the total data flow of the policy. For each proactive policy, short tables in the reliability and storage utilization sections are provided with comparisons to the 2 and 3-replica reactive policies.

CHAPTER FOUR
RESULTS AND DISCUSSION

Introduction

The results of the simulations for the four failure models and data dispatcher policies are described in this chapter. The results are organized into sections and subsections based on the failure model and analysis type. The first two sections cover the SMART error failures models, disk age and error count. Followed by a section for the CFDR age failure model and concluding with the manufacturer, model and vintage failure model section. For each failure model, the results of three different data dispatcher policies are covered. All of the sections share the same two policies: simple and reactive with the number of requested replicas between 1 and 3. These instances serve as a baseline for comparison against the sections' final proactive specific policy mechanisms. The exact configurations of these proactive policies vary depending on the failure model and policy being employed. Their individual attributes and behavior are discussed in the methodology chapter. Every section examines the same four aspects of the policies: long-term data survival, storage and replicas used, and bandwidth usage. Of the greatest importance is data survival, since resource utilization is of little concern if all or most of the data is lost. Of the two resources, storage is often viewed as more important than bandwidth as storage usage is more permanent than transient bandwidth utilization. The NOMDL is included after the reliability and storage analyses as it aids in examining the potential tradeoff between reliability and storage usage that is not as readily apparent when viewed separately.

As a reminder, the simulations individually covered a mission time of 100 years during which 3 million 1-gigabyte data objects entered the Logistical Networking system at one-minute intervals from the start of the simulation. The storage consisted of roughly 9+ petabytes spread equally across the nine different sites of REDDnet. The storage was divided amongst the 16 depots at every site into 32 2-gigabyte drives per depot. The depots at a site shared a 10 Gb/s interconnect with one another and a 10 GB interconnect to the site gateway. The available bandwidth and route of the wide area network links between sites depended on their real life counterparts. Chapter 3 has further details about the models, policies and configuration of the simulation environment.

Apart from the analysis, the simulations and raw data collection alone took close to 4 years requiring substantial processor and storage allocations to achieve. This work was made possible to the generous participation of staff and hardware at several HPC facilities including:

- The Advanced Computing Center for Research and Education (ACCRE) at Vanderbilt University, Nashville, TN.
- The Department of Electrical Engineering and Computer Science (EECS) at the University of Tennessee, Knoxville.
- National Institute for Computational Sciences (NICS) at the University of Tennessee, Knoxville and Oak Ridge National Laboratory.
- The Newton HPC Program at the University of Tennessee, Knoxville
- The Data Observation Network for Earth (DataONE), a collaboration among multiple organizations funded by the US National Science Foundation (NSF)

Table 9, provides the number of simulations performed and analyzed as well as the number of node-hours required for their execution. The table partitions the simulations by the failure model, policy and policy configuration parameters to provide a breakdown of the individual populations used in the following sections' analyses.

SMART Error Age

Data Survival

Simple

Figure 35 and Table 10 provide an overview of the reliability analysis of the simple policy runs on the SMART Error Age Failure model. As expected, the data objects' survival did not fare well under any of the three replication settings. Without the ability to create new replicas after a failure event, even with three initial replicas, almost all of the data objects were lost for every simulation.

As all of the loss curves pass the 40-year mark, the difference in reliability between them decreases as the 2 and 3-replica policies approach near total data loss. The final mean data object loss is 2,998,257 for the 1-replica policy and 2,994,430 for the 3-replica policy, a different of less than 4,000 data objects out of a 3 million. All three simple policies have simulations where they suffer complete data loss. Of the over 1,500 simulations performed, the earliest total data loss occurs at 86, 94, 92 years for the 1-replica, 2-replica and 3-replica policies respectively. However, the figure also reveals that every additional replica does slow the pace of data loss. For instance, a mean data loss of 50% for each policy occurs at 8 years for 1 replica, 16 years for 2 replicas and finally 21 years for 3 replicas. The greatest mean difference in data object loss is in the 11^{th} year for 1-replica and 2-replica with 747,890 objects, and the 9^{th} year for 1-replica and 3-replica policies with 1,152,000 objects. The smallest difference occurs for all policies in the final year with 2,039 and 3,827 objects.

Reactive

Figure 36 and Table 11 provide a summary of the results for the reactive policy against the SMART error age model for all of the simulations executed.

Table 9. Simulations performed with count and compute hours.

Failure	Policy	Replica	Reliability/Health	Count	Hours
SMART Error Disk Age	Simple	1		531	1923.69
		2		534	3181.60
		3		510	4113.90
	Reactive	1		514	2737.40
		2		553	13464.09
		3		516	26429.53
	SMART Error Proactive	1	1.0	501	3668.03
		2	1.0	504	17602.79
		2	1.5	537	12327.75
		2	2.0	519	24531.64
SMART Error Count	Simple	1		511	1585.22
		2		503	2690.13
		3		621	4763.27
	Reactive	1		501	6391.52
		2		502	7057.81
		3		514	21587.07
	SMART Error Proactive	1	1.0	504	3923.97
		2	1.0	503	12144.55
		2	1.5	509	10711.82
		2	2.0	507	18506.56
CFDR Age	Simple	1		504	6831.82
		2		519	2566.16
		3		506	4485.52
	Reactive	1		508	6039.94
		2		499	7457.55
		3		522	12451.18
	CFDR Age Proactive	1	0.9	214	1367.77
		1	0.99	204	3795.09
		1	0.999	207	10968.95
		1	0.9999	200	15827.69
		1	0.99999	23	22837.34
		2	0.999	204	12425.53
		2	0.9999	208	12624.57
		2	0.99999	124	20457.57
		3	0.99999	207	26727.89
Vintage	Simple	1		243	292.56
		2		207	446.28
		3		207	684.94
	Reactive	1		213	293.60
		2		204	5408.79
		3		208	6913.35
	Vintage Proactive	1	0.99	156	18084.92
		1	0.999	218	8158.40
		1	0.9999	280	11517.93
		1	0.99999	183	13979.71
		2	0.9999	106	5841.20
		2	0.99999	122	9515.67
		2	0.999999	129	44709.48
		2	0.9999999	108	40765.72
Total				17627	532819.46

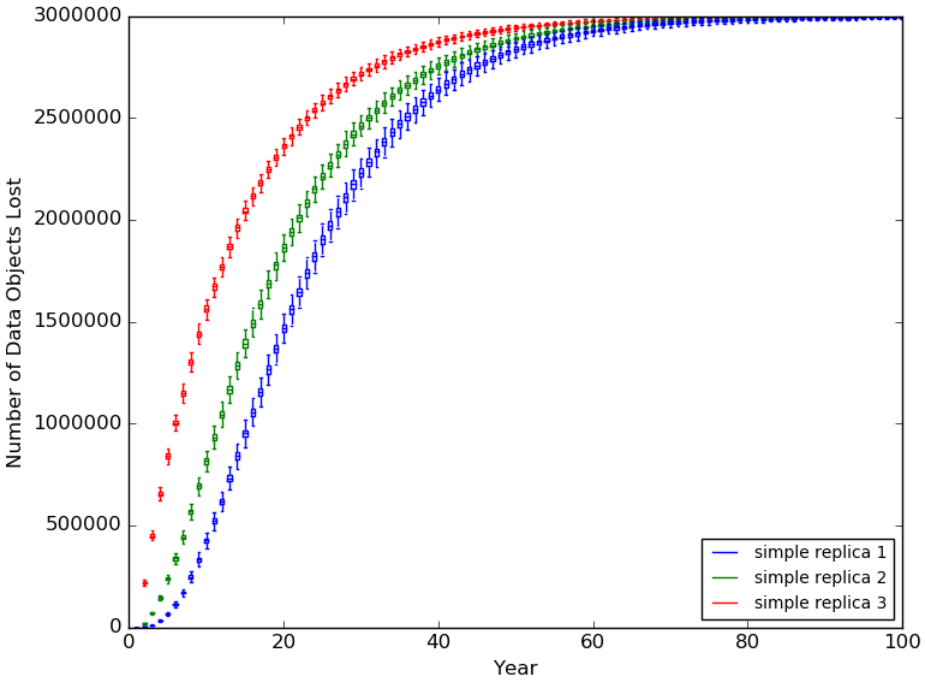


Figure 35. Simple policy reliability against the SMART Error Age Model. The cumulative data object loss for every run of the simple policy on the SMART Error Age Model grouped by the replication setting of the policy as a time series of Tukey box plots.

There is a substantial greater difference between the three different reactive policy configurations than for the simple policy. In fact, while 1-replica has near complete data loss for every run, the 2 and 3-replica reactive policies complete the 100-year time frame retaining most of their data objects. The mean data object loss for the 2-replica is 292,432, 9.75%, and only 2,956, 0.099%, of the 3 million objects. These results with a tight standard deviation of 5,586 and 263 data objects illustrate exactly how much better at maintaining data a reactive policy is and how the reactive approach improves for each additional replica.

The 2-replica reactive policy loses 98.8 times more data objects than the 3-replica by the end of the mission time. The 1-replica policy performs 10 and 1015 times worse than the 2 and 3-replica policies. By allowing new replicas to be created after a storage failure, maintaining just two additional replicas is enough to provide 3 nine's reliability for 100 years. Not surprisingly in Figure 37, the 1-replica reactive performs the same as the 1-replica simple since the reactive policy logically reduces to the simple policy when there is not another replica from which to generate a replacement.

Table 10. Data Object loss by the simple policies against the SMART error age model.

Data Object Loss by Policy				
1-Replica Simple				
Year	Mean	Std. Dev.	2-Replica Reactive difference	3- Replica Reactive difference
5	1003874	17130	985614 (55.0 x)	1003663(4760.6 x)
10	1672253	19615	1636294 (46.5 x)	1671856(4219.5 x)
20	2411294	16391	2342039 (34.8 x)	2410557(3270.4 x)
30	2738606	11694	2637112 (27.0 x)	2737538(2564.3 x)
40	2884035	8299	2750636 (21.6 x)	2882635(2061.0 x)
50	2948671	5546	2783803 (17.9 x)	2946937(1699.8 x)
60	2977283	3667	2781274 (15.2 x)	2975213(1438.3 x)
70	2989974	2566	2763565 (13.2 x)	2987575(1246.2 x)
80	2994927	1796	2746903 (12.1 x)	2992324(1150.7 x)
90	2997038	1404	2731601 (11.3 x)	2994304(1096.1 x)
100	2998257	1156	2705825 (10.3 x)	2995302(1014.4 x)
2-Replica Simple				
Year	Mean	Std. Dev.	2-Replica Reactive difference	3- Replica Reactive difference
5	336734	10987	318474 (18.4 x)	336523(1596.9 x)
10	932506	21837	896548 (25.9 x)	932110(2352.9 x)
20	1940276	25571	1871021 (28.0 x)	1939539(2631.6 x)
30	2502533	20125	2401039 (24.7 x)	2501465(2343.2 x)
40	2774208	14617	2640810 (20.8 x)	2772809(1982.5 x)
50	2899006	10566	2734138 (17.6 x)	2897271(1671.2 x)
60	2955114	7398	2759106 (15.1 x)	2953044(1427.6 x)
70	2979576	5011	2753166 (13.2 x)	2977177(1241.9 x)
80	2989412	3624	2741388 (12.1 x)	2986809(1148.6 x)
90	2993777	2749	2728340 (11.3 x)	2991043(1094.9 x)
100	2996218	2228	2703786 (10.2 x)	2993262(1013.7 x)
3-Replica Simple				
Year	Mean	Std. Dev.	2-Replica Reactive difference	3- Replica Reactive difference
5	112609	5442	94349 (6.2 x)	112399(534.0 x)
10	520349	16805	484390 (14.5 x)	519952(1313.0 x)
20	1560006	30076	1490751 (22.5 x)	1559269(2115.8 x)
30	2284598	29070	2183104 (22.5 x)	2283530(2139.2 x)
40	2667078	22172	2533679 (20.0 x)	2665679(1905.9 x)
50	2849933	14973	2685065 (17.3 x)	2848198(1642.9 x)
60	2932631	10172	2736623 (15.0 x)	2930561(1416.8 x)
70	2969429	7064	2743020 (13.1 x)	2967030(1237.7 x)
80	2984169	5001	2736145 (12.0 x)	2981567(1146.6 x)
90	2990767	3814	2725330 (11.3 x)	2988033(1093.8 x)
100	2994430	3190	2701998 (10.2 x)	2991475(1013.1 x)

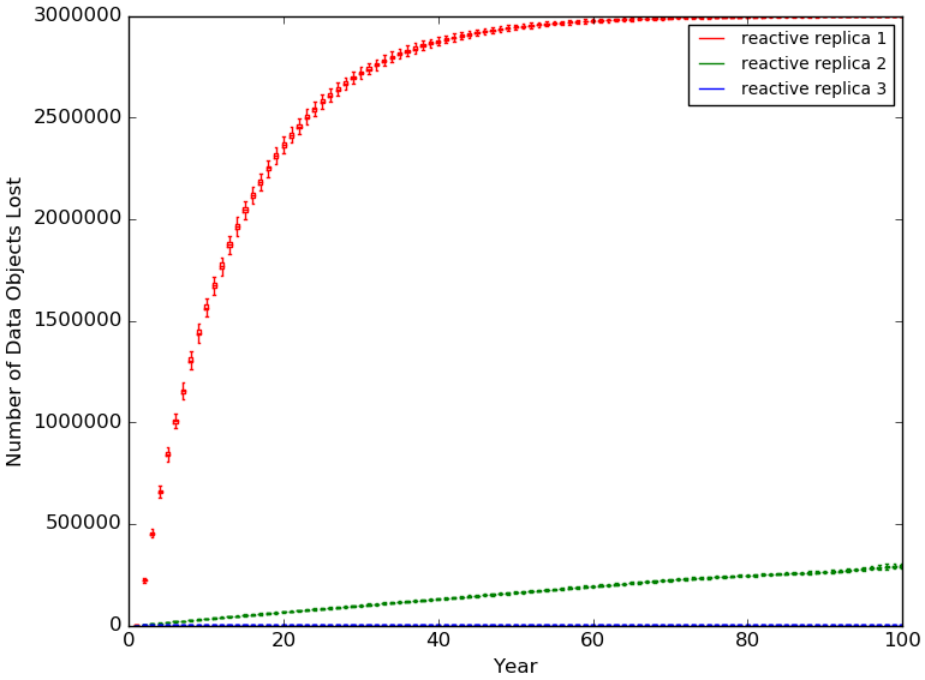


Figure 36. Reactive policy reliability on the SMART error model. The cumulative data object loss for every run of the reactive policy on the SMART Error Age Model grouped by the replication setting of the policy as a time series of Tukey box plots.

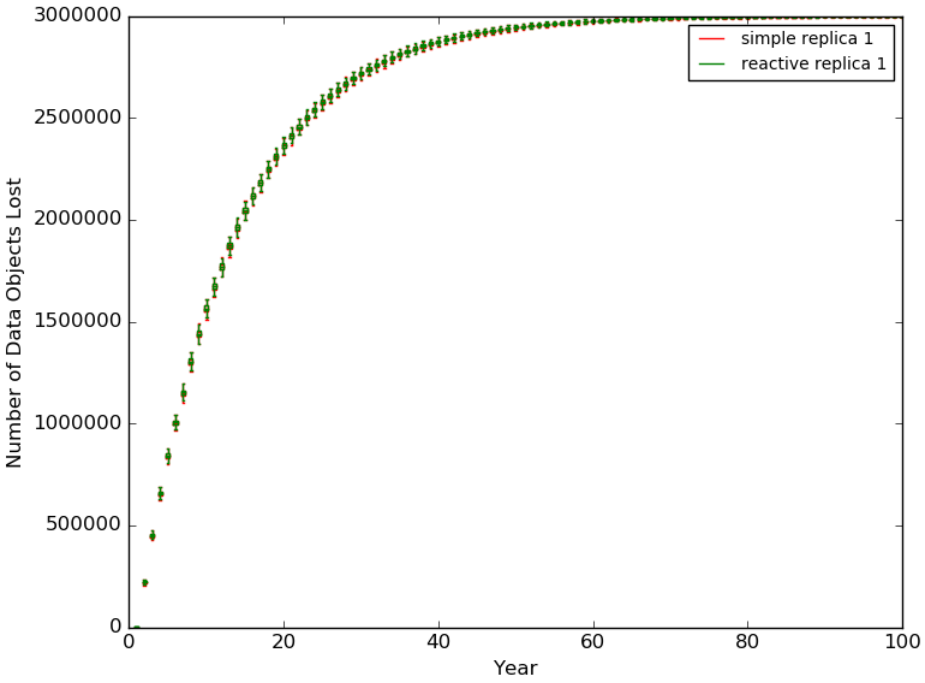


Figure 37. 1-replica reactive vs. 1-replica simple policy reliability on the SMART error model. The cumulative data object loss for both policies as a time series of Tukey box plots.

Table 11. Data Object loss by the reactive policies against the SMART error age model.

Data Object Loss by Policy				
1-Replica Reactive				
Year	Mean	Std. Dev.	2-Replica Reactive Difference	3-Replica Reactive Difference
5	1005447	15028	987187 (55 x)	1005236(4768 x)
10	1673345	17620	1637387 (47 x)	1672949(4222 x)
20	2414208	14833	2344954 (35 x)	2413471(3274 x)
30	2741398	10962	2639904 (27 x)	2740330(2567 x)
40	2886158	7501	2752759 (22 x)	2884758(2062 x)
50	2950357	5038	2785489 (18 x)	2948622(1701 x)
60	2978703	3264	2782695 (15 x)	2976633(1439 x)
70	2991155	2110	2764745 (13 x)	2988755(1247 x)
80	2996007	1384	2747982 (12 x)	2993404(1151 x)
90	2998105	879	2732668 (11 x)	2995370(1096 x)
100	2999322	381	2706890 (10 x)	2996367(1015 x)
2-Replica Reactive				
Year	Mean	Std. Dev.	2-Replica Reactive Difference	3-Replica Reactive Difference
5	18260	625	0 (1 x)	18049(86.6 x)
10	35958	1112	0 (1 x)	35562(90.7 x)
20	69255	1588	0 (1 x)	68518(93.9 x)
30	101494	1858	0 (1 x)	100426(95.0 x)
40	133399	2006	0 (1 x)	131999(95.3 x)
50	164868	2127	0 (1 x)	163134(95.0 x)
60	196008	2260	0 (1 x)	193938(94.7 x)
70	226409	2221	0 (1 x)	224010(94.4 x)
80	248024	2200	0 (1 x)	245421(95.3 x)
90	265437	2633	0 (1 x)	262702(97.1 x)
100	292432	5586	0 (1 x)	289476(98.9 x)
3-Replica Reactive				
Year	Mean	Std. Dev.	2-Replica Reactive Difference	3-Replica Reactive Difference
5	211	19	-18049 (0.012 x)	0(1 x)
10	396	27	-35562 (0.011 x)	0(1 x)
20	737	37	-68518 (0.011 x)	0(1 x)
30	1068	45	-100426 (0.011 x)	0(1 x)
40	1399	56	-131999 (0.010 x)	0(1 x)
50	1735	65	-163134 (0.011 x)	0(1 x)
60	2070	77	-193938 (0.011 x)	0(1 x)
70	2399	92	-224010 (0.011 x)	0(1 x)
80	2603	101	-245421 (0.010 x)	0(1 x)
90	2734	138	-262702 (0.010 x)	0(1 x)
100	2956	263	-289476 (0.010 x)	0(1 x)

SMART Error-Aware Proactive

Figure 38 provides the results of the reliability analysis of the SMART error-aware proactive policy for the various parameters. As can be clearly seen, the policy when to set to maintain one replica with a health of 1.0 performs so significantly worse than all of the other policies that it completely dominates the graph obscuring the other results. Therefore to provide better detail, Figure 39 leaves it out to provide better detail on the results of the rest of the policies.

For the three remaining policies in Figure 39, every increase of the data health metric lessens the median data loss from 4,636 objects (.155%) to 2,359 (.079%) and ultimately to 1,396 (.047%). This means that for the same replica count of 2, requesting two SMART error free disks versus just 1 improved the reliability 3.3 times.

Figure 40 and Figure 41 show the relationship between the 2 and 3-replica reactive and the 2-replica proactive policies. The mean lost data objects for the 2-replica reactive policy is 292,432 which is 63 times greater than the worst performing 2-replica proactive policy at 4,636 objects. The 3-replica reactive policy beats out the first 2-replica proactive policy but is itself outperformed by the 2-replica, 1.5 and 2.0-health proactive. Only the worst simulation runs of the

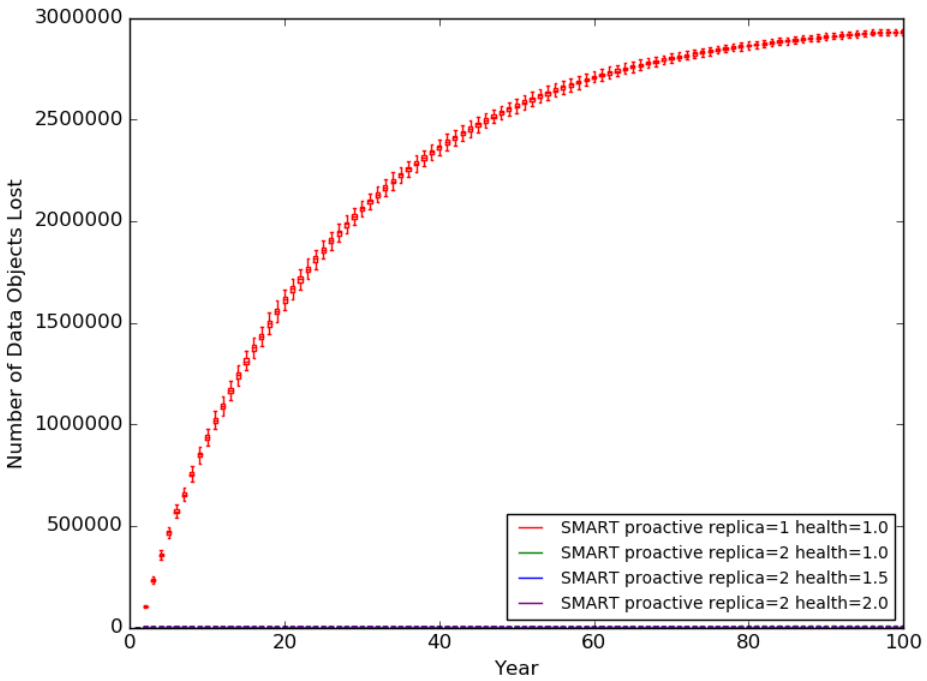


Figure 38. SMART error proactive policy reliability against the SMART error age model. The cumulative data object loss for every run of the proactive policy grouped by the replication and health parameters as a time series of Tukey box plots.

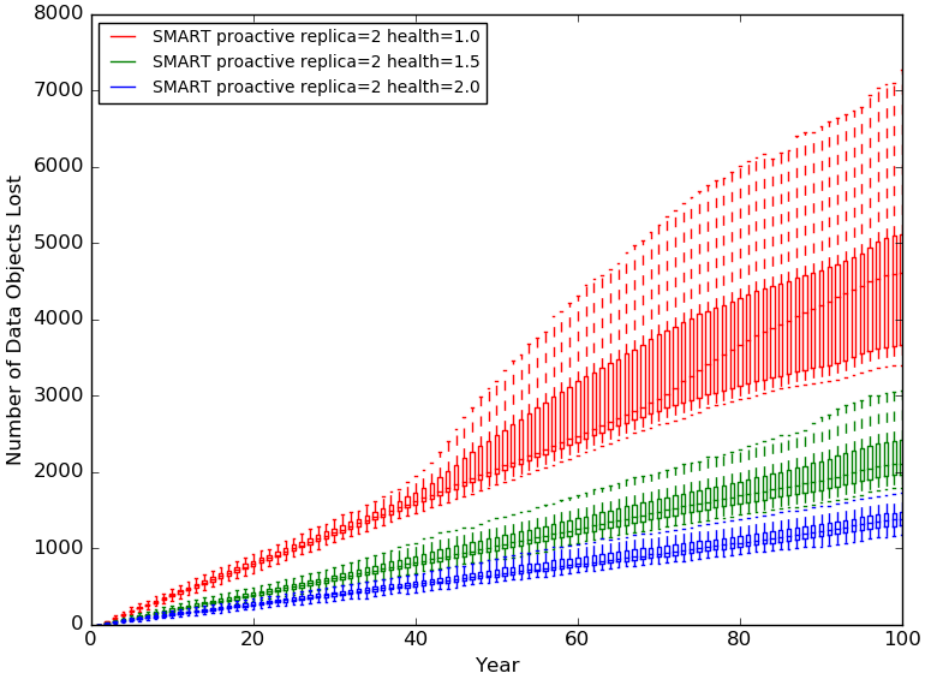


Figure 39. SMART error proactive policies against the SMART Error failure model.

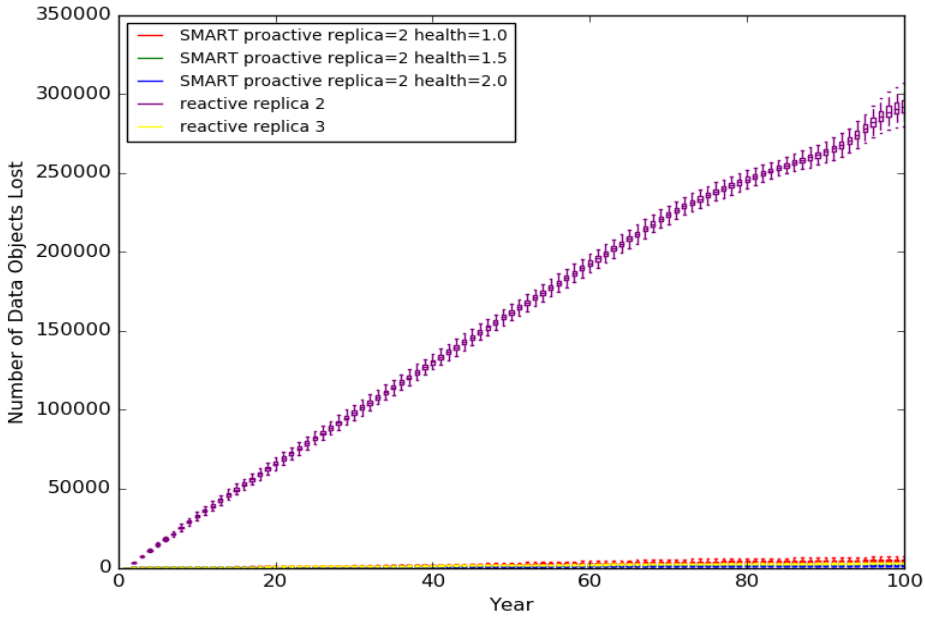


Figure 40. SMART error aware proactive reliability vs. reactive policies for the SMART error Age Model.

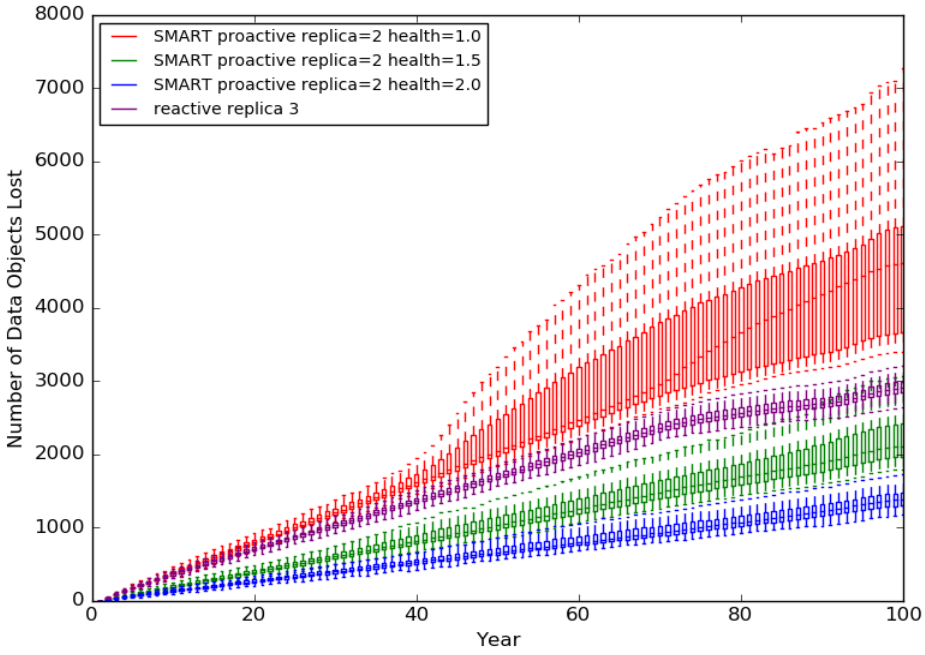


Figure 41. SMART error aware proactive reliability vs. 3- replica reactive for the SMART Error Age Model.

1.5-health proactive policy landed within the best to median reliability results of the 3-replica reactive. Finally, the 2.0-health proactive policy did better than both those policies without equivocation. On average, the 3-replica reactive policy suffered 1.3 and 2.1 times more data loss than the 1.5 and 2.0-health proactive policies respectively.

Re-examining the 1-replica proactive policy in the next figure, Figure 42 plots the reliability of the policy against 1 and 2-replica reactive policies. As previously stated, even though the 1-replica proactive policy does very poorly, it still achieves better reliability than its 1-replica reactive policy counterpart. While only preserving over 2% of the total data objects, it still illustrates how some knowledge of the state of the underlying storage infrastructure can improve reliability.

Table 12 summarizes these results of the data object loss for the SMART error age failure model by the proactive policy employed. The table provides results for the first five years and then in 10-year intervals beyond that. As well as the mean, it provides the standard deviation and the number of additional data objects lost relative to the 2 and 3-replica reactive policies with their multiplicative differences.

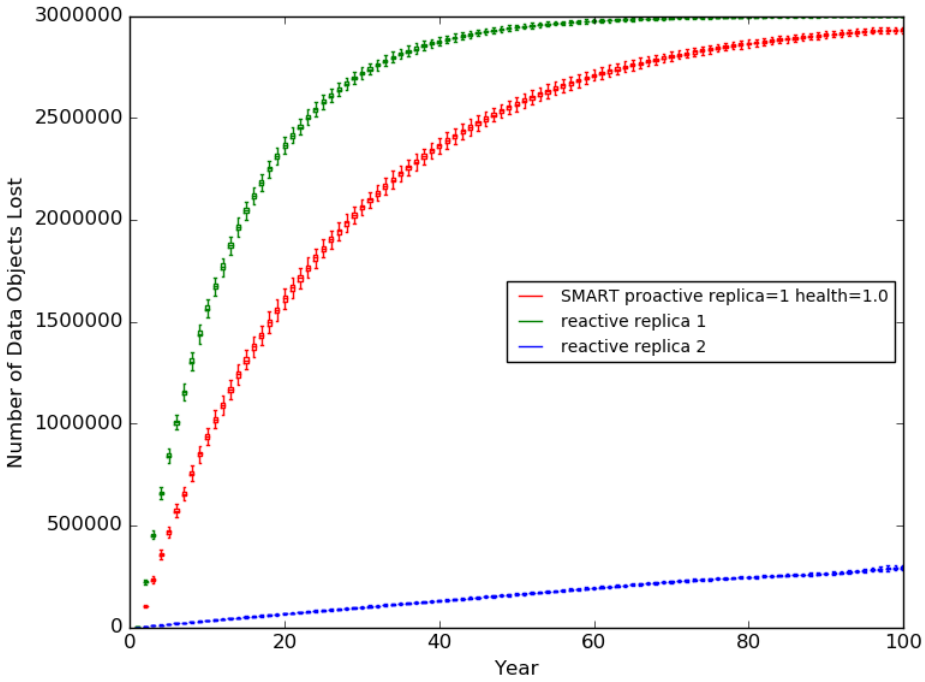


Figure 42. SMART error aware proactive with 1 copy and health of 1.0 vs. reactive policy.

Table 12. Data Object loss by the proactive policies against the SMART Error Age Model. For each of the of the proactive policies, the table provides the mean and standard deviation for the data lost for each policy as well as the additive and multiplicative difference to the 2 and 3-replica reactive policies.

Data Object Loss by Policy				
1-Replica 1.0-health SMART error proactive				
Year	Mean	Std. Dev.	2-Replica Reactive Difference	3-Replica Reactive Difference
5	572454	12818	554194 (31.3 x)	572243(2714.7 x)
10	1019684	18551	983726 (28.4 x)	1019288(2572.9 x)
20	1666650	19290	1597395 (24.1 x)	1665913(2260.5 x)
30	2098319	16958	1996825 (20.7 x)	2097251(1964.7 x)
40	2389285	15085	2255886 (17.9 x)	2387886(1707.4 x)
50	2586287	13182	2421419 (15.7 x)	2584552(1490.9 x)
60	2719864	10598	2523856 (13.9 x)	2717794(1314.0 x)
70	2810007	8500	2583598 (12.4 x)	2807608(1171.2 x)
80	2870936	7346	2622912 (11.6 x)	2868334(1103.1 x)
90	2912519	6362	2647083 (11.0 x)	2909785(1065.2 x)
100	2933185	5932	2640753 (10.0 x)	2930230(992.4 x)

Table 12. Continued.

2-Replica 1.0-health SMART error proactive				
Year	Mean	Std. Dev.	2-Replica Reactive Difference	3-Replica Reactive Difference
5	221	23	-18039 (0.01 x)	10(1.0 x)
10	435	51	-35524 (0.01 x)	38(1.1 x)
20	870	130	-68385 (0.01 x)	133(1.2 x)
30	1326	232	-100169 (0.01 x)	258(1.2 x)
40	1809	353	-131590 (0.01 x)	410(1.3 x)
50	2317	474	-162551 (0.01 x)	582(1.3 x)
60	2852	596	-193156 (0.01 x)	782(1.4 x)
70	3402	714	-223008 (0.02 x)	1002(1.4 x)
80	3877	830	-244147 (0.02 x)	1275(1.5 x)
90	4272	936	-261164 (0.02 x)	1538(1.6 x)
100	4636	1010	-287796 (0.02 x)	1681(1.6 x)
2-Replica 1.5-health SMART error proactive				
Year	Mean	Std. Dev.	2-Replica Reactive Difference	3-Replica Reactive Difference
5	108	16	-18152 (0.006 x)	-103(0.51 x)
10	216	30	-35742 (0.006 x)	-180(0.55 x)
20	436	50	-68818 (0.006 x)	-301(0.59 x)
30	674	64	-100820 (0.007 x)	-394(0.63 x)
40	921	77	-132478 (0.007 x)	-478(0.66 x)
50	1172	86	-163696 (0.007 x)	-563(0.68 x)
60	1428	93	-194580 (0.007 x)	-642(0.69 x)
70	1678	101	-224732 (0.007 x)	-722(0.70 x)
80	1913	107	-246111 (0.008 x)	-689(0.74 x)
90	2143	117	-263294 (0.008 x)	-592(0.78 x)
100	2359	127	-290073 (0.008 x)	-596(0.80 x)
2-Replica 2.0-health SMART error proactive				
Year	Mean	Std. Dev.	2-Replica Reactive Difference	3-Replica Reactive Difference
5	78	16	-18182 (0.004 x)	-133(0.37 x)
10	150	30	-35808 (0.004 x)	-246(0.38 x)
20	288	50	-68967 (0.004 x)	-450(0.39 x)
30	424	64	-101070 (0.004 x)	-644(0.40 x)
40	564	77	-132835 (0.004 x)	-836(0.40 x)
50	699	86	-164169 (0.004 x)	-1035(0.40 x)
60	835	93	-195173 (0.004 x)	-1235(0.40 x)
70	971	101	-225438 (0.004 x)	-1428(0.40 x)
80	1111	107	-246913 (0.004 x)	-1492(0.43 x)
90	1260	117	-264177 (0.005 x)	-1475(0.46 x)
100	1397	127	-291035 (0.005 x)	-1559(0.47 x)

Storage Used & Copies Used

Simple

Figure 43 provides two graphs with three boxplot time series representing the storage utilized by the three simple policies over the 100-year time scale. Each of the storage utilization curves possess the same overall shape where the first 5.66 years is spent handling the arrival of new data objects from the clients and replicating them to the various sites. The policies peak at this point with mean values of 2.41, 4.83, and 7.24 petabytes. Had no data loss occurred to this point in time then the expected storage utilization should have 3, 6 and 9 petabytes in order for each data object to have been replicated sufficiently according to the given policy. Once the peak has been reached, the storage utilization plummets as replicas for the data objects are lost. A summary of the results is provided in Table 13.

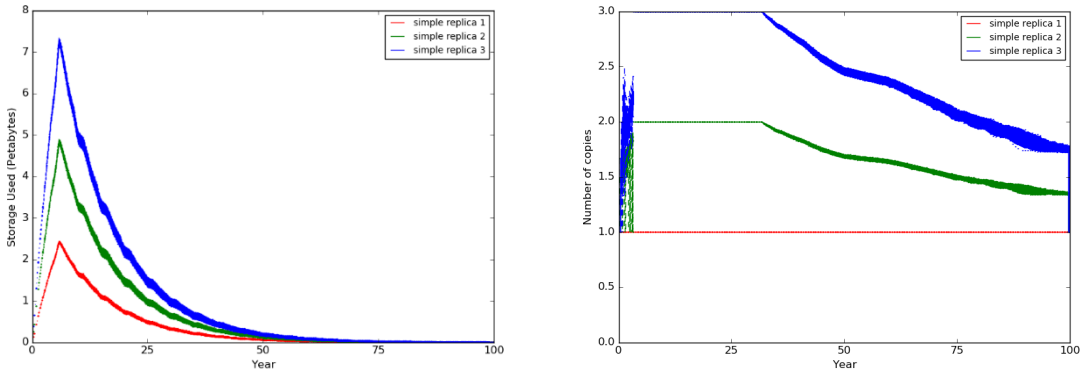


Figure 43. Storage and copies used over time for the simple policies. The left figure gives the storage (in petabytes) used over time by the three simple policies. The right figure gives the number of replicas per data object for data objects that survived to that time.

Reactive

An overview of the results for the reactive policy is given in Figure 44 and Table 14. The storage usage for the 1-replica reactive policy is the same as the simple policy counterpart, but the storage usage for the 2 and 3-replica reactive policies are significantly more consistent owing to the policy being able to replace lost replicas. After the initial populate phase, all three policies approach their expected storage usages of 3, 6 and 9 petabytes. The 1-replica policy was noticeability less than 3 petabytes having already lost a million data objects. The 2 and 3-replica policies trend downwards over time with the 2-replica trending downwards slightly faster.

Table 13. Storage utilization and replica count for the simple policies with the SMART error age model

1-Replica Simple						
	Replicas		Storage Used (PB)			
Year	Count	Std. Dev.	Mean	Std. Dev.	2-Replica Reactive Difference	3-Replica Reactive Difference
10	1.00	0.00	1.64	0.022	-4.29 (0.3 x)	-7.33 (0.2 x)
20	1.00	0.00	0.73	0.019	-5.12 (0.1 x)	-8.23 (0.08 x)
30	1.00	0.00	0.32	0.013	-5.46 (0.06 x)	-8.62 (0.04 x)
40	1.00	0.00	0.14	0.009	-5.57 (0.02 x)	-8.80 (0.02 x)
50	1.00	0.00	0.06	0.006	-5.59 (0.01 x)	-8.87 (0.007 x)
60	1.00	0.00	0.03	0.004	-5.56 (0.005 x)	-8.91 (0.003 x)
70	1.00	0.00	0.01	0.003	-5.51 (0.002 x)	-8.92 (0.001 x)
80	1.00	0.00	0.01	0.002	-5.48 (0.001 x)	-8.94 (0.0007 x)
90	1.00	0.00	0.00	0.001	-5.46 (0.0006 x)	-8.95 (0.0004 x)
100	1.00	0.00	0.00	0.001	-5.41 (0.0004 x)	-8.93 (0.0003 x)
2-Replica Simple						
	Replicas		Storage Used (PB)			
Year	Count	Std. Dev.	Mean	Std. Dev.	2-Replica Reactive Difference	3-Replica Reactive Difference
10	2.00	0.00	3.27	0.041	-2.65 (0.6 x)	-5.69 (0.4 x)
20	2.00	0.00	1.45	0.037	-4.40 (0.2 x)	-7.50 (0.2 x)
30	2.00	0.00	0.64	0.025	-5.14 (0.1 x)	-8.30 (0.07 x)
40	1.84	0.01	0.28	0.018	-5.43 (0.05 x)	-8.65 (0.03 x)
50	1.69	0.01	0.13	0.012	-5.52 (0.02 x)	-8.81 (0.01 x)
60	1.64	0.01	0.06	0.009	-5.53 (0.01 x)	-8.88 (0.006 x)
70	1.54	0.01	0.02	0.006	-5.50 (0.004 x)	-8.91 (0.003 x)
80	1.46	0.01	0.01	0.004	-5.48 (0.002 x)	-8.93 (0.001 x)
90	1.39	0.02	0.01	0.003	-5.45 (0.001 x)	-8.94 (0.0008 x)
100	1.33	0.08	0.00	0.002	-5.41 (0.0009 x)	-8.93 (0.0005 x)
3-Replica Simple						
	Replicas		Storage Used (PB)			
Year	Count	Std. Dev.	Mean	Std. Dev.	2-Replica Reactive Difference	3-Replica Reactive Difference
10	3.00	0.00	4.91	0.060	-1.020 (0.8 x)	-4.050 (0.5 x)
20	3.00	0.00	2.17	0.053	-3.680 (0.4 x)	-6.780 (0.2 x)
30	3.00	0.00	0.96	0.040	-4.820 (0.2 x)	-7.980 (0.1 x)
40	2.74	0.01	0.43	0.027	-5.290 (0.07 x)	-8.510 (0.05 x)
50	2.46	0.01	0.19	0.017	-5.460 (0.03 x)	-8.750 (0.02 x)
60	2.36	0.02	0.08	0.012	-5.500 (0.01 x)	-8.850 (0.009 x)
70	2.17	0.02	0.04	0.01	-5.49 (0.007 x)	-8.89 (0.004 x)
80	1.99	0.03	0.02	0.01	-5.47 (0.003 x)	-8.92 (0.002 x)
90	1.84	0.04	0.01	0.00	-5.45 (0.002 x)	-8.94 (0.001 x)
100	1.72	0.17	0.01	0.00	-5.40 (0.001 x)	-8.92 (0.0008 x)

Table 14. Storage utilization and replica count for the reactive policies with the SMART error age model

1-replica reactive						
	Replicas		Storage Used (PB)			
Year	Count	Std. Dev.	Mean	Std. Dev.	2-replica reactive difference	3-replica reactive difference
10	1.00	0.00	1.630	0.019	-4.290 (0.3 x)	-7.330 (0.2 x)
20	1.00	0.00	0.722	0.016	-5.130 (0.1 x)	-8.230 (0.08 x)
30	1.00	0.00	0.320	0.013	-5.460 (0.06 x)	-8.620 (0.04 x)
40	1.00	0.00	0.142	0.009	-5.570 (0.02 x)	-8.800 (0.02 x)
50	1.00	0.00	0.063	0.006	-5.590 (0.01 x)	-8.870 (0.007 x)
60	1.00	0.00	0.028	0.004	-5.560 (0.005 x)	-8.910 (0.003 x)
70	1.00	0.00	0.01	0.00	-5.51 (0.002 x)	-8.92 (0.001 x)
80	1.00	0.00	0.01	0.00	-5.48 (0.001 x)	-8.94 (0.0007 x)
90	1.00	0.00	0.00	0.00	-5.46 (0.0007 x)	-8.95 (0.0004 x)
100	1.00	0.00	0.00	0.00	-5.41 (0.0005 x)	-8.93 (0.0003 x)
2-replica reactive						
	Replicas		Storage Used (PB)			
Year	Count	Std. Dev.	Mean	Std. Dev.	2-replica reactive difference	3-replica reactive difference
10	1.99	0.00	5.930	0.005	0.0 (1 x)	-3.04 (0.7 x)
20	1.99	0.00	5.850	0.007	0.0 (1 x)	-3.10 (0.7 x)
30	1.99	0.00	5.780	0.008	0.0 (1 x)	-3.16 (0.6 x)
40	1.99	0.00	5.710	0.009	0.0 (1 x)	-3.22 (0.6 x)
50	1.99	0.00	5.650	0.009	0.0 (1 x)	-3.29 (0.6 x)
60	1.99	0.00	5.590	0.009	0.0 (1 x)	-3.35 (0.6 x)
70	1.99	0.00	5.53	0.01	0.0 (1 x)	-3.41 (0.6 x)
80	1.99	0.00	5.49	0.01	0.0 (1 x)	-3.46 (0.6 x)
90	1.99	0.00	5.46	0.01	0.0 (1 x)	-3.49 (0.6 x)
100	1.99	0.00	5.41	0.01	0.0 (1 x)	-3.52 (0.6 x)
3-replica reactive						
	Replicas		Storage Used (PB)			
Year	Count	Std. Dev.	Mean	Std. Dev.	2-replica reactive difference	3-replica reactive difference
10	2.98	0.00	8.96	0.009	3.04 (1.5 x)	0.0 (1 x)
20	2.98	0.00	8.95	0.009	3.10 (1.5 x)	0.0 (1 x)
30	2.98	0.00	8.94	0.011	3.16 (1.5 x)	0.0 (1 x)
40	2.98	0.00	8.94	0.010	3.22 (1.6 x)	0.0 (1 x)
50	2.98	0.00	8.94	0.011	3.29 (1.6 x)	0.0 (1 x)
60	2.98	0.00	8.93	0.012	3.35 (1.6 x)	0.0 (1 x)
70	2.98	0.00	8.93	0.011	3.41 (1.6 x)	0.0 (1 x)
80	2.98	0.00	8.94	0.010	3.46 (1.6 x)	0.0 (1 x)
90	2.99	0.00	8.95	0.009	3.49 (1.6 x)	0.0 (1 x)
100	2.98	0.00	8.93	0.011	3.52 (1.7 x)	0.0 (1 x)

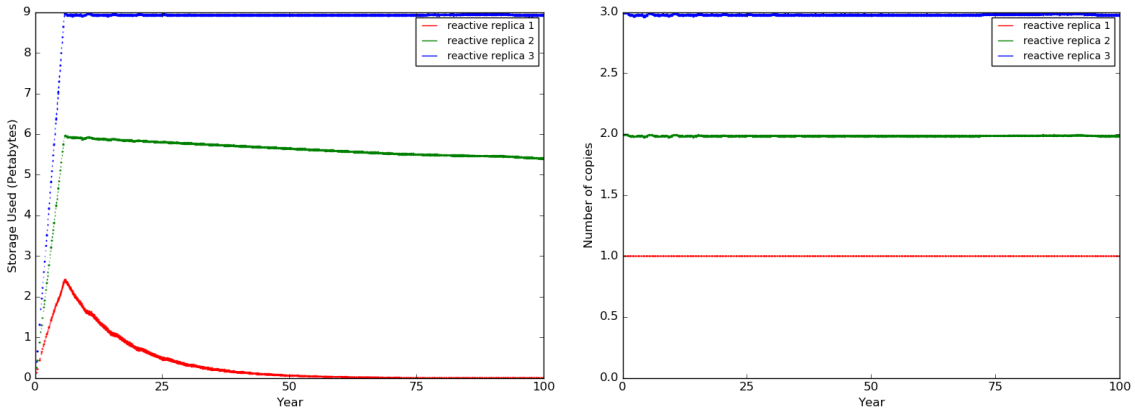


Figure 44. Storage and copies used over time for reactive policies. The left figure gives the storage (in petabytes) used over time by the three simple policies. The right figure gives the number of replicas per data object for data objects that survived to that time.

The number of replicas per surviving data object remains mostly constant and matches the requested replica count for the policies. There is some minor fluctuation for the 2 and 3-replica policies as replicas were being lost and regenerated. The 1-replica policy has no such fluctuations as the instant a replica is lost the data object becomes defunct.

SMART Error-aware Proactive

The storage situation with SMART error-aware proactive policies is more interesting and is summarized in Figure 45 and Table 15. As seen in the data survival section, the 1-replica, 1.0-health policy preserves data significantly worse than all of the SMART error-aware policies by the end of the 100-year timeframe, only 2.2% fewer losses than 1-replica reactive. As a result, the storage usage is similar to the 1-replica simple and reactive policies albeit with a shallower curve. The second part of Figure 45 figure help illustrates the cause of its poor performance. The average number of replicas per surviving data object only barely exceeds the minimum required number of 1. The large fluctuations in the boxplots suggest that the policy is having to respond to the SMART errors and executing additional replication and data migration. Although the policy requires that all replicas must reside on SMART error free disks for nine months, the lethality associated with the SMART errors is too severe to migrate all of the data objects before disk failure. Combine this with 56% of all disk failures having no preceding SMART error and the policy cannot maintain the data sufficiently.

The case drastically improves, though, with the other SMART error-aware policies. Both the 2-replica, 1.0-health and 1-5-health policies utilize nearly six petabytes of storage indicating that they are maintaining two copies per data object on the average. This claim is further supported by the second graph that

Table 15. Replicas and Storage Used for SMART Error Proactive Policies.
This table supplies, in 10-year intervals, the mean replica count and standard deviation of every surviving data object on the left side. On the right are the total mean storage usage, standard deviation and additive and multiplicative differences for the 2 and 3-replica reactive policies in petabytes.

1-replica 1.0-health SMART-aware proactive						
	Replicas		Storage Used (PB)			
Year	Count	Std. Dev.	Mean	Std. Dev.	2-replica reactive difference	3-replica reactive difference
10	1.07	0.00	2.35	0.021	-3.57 (0.4 x)	-6.61 (0.3 x)
20	1.11	0.01	1.64	0.024	-4.21 (0.3 x)	-7.32 (0.2 x)
30	1.12	0.01	1.12	0.022	-4.66 (0.2 x)	-7.83 (0.1 x)
40	1.12	0.01	0.76	0.019	-4.96 (0.1 x)	-8.18 (0.08 x)
50	1.12	0.01	0.52	0.016	-5.13 (0.09 x)	-8.42 (0.06 x)
60	1.12	0.01	0.35	0.014	-5.24 (0.06 x)	-8.58 (0.04 x)
70	1.12	0.02	0.24	0.011	-5.29 (0.04 x)	-8.69 (0.03 x)
80	1.12	0.02	0.16	0.009	-5.33 (0.03 x)	-8.78 (0.02 x)
90	1.12	0.03	0.11	0.008	-5.35 (0.02 x)	-8.84 (0.01 x)
100	1.13	0.03	0.08	0.007	-5.33 (0.01 x)	-8.85 (0.009 x)
2-replica 1.0-health SMART-aware proactive						
	Replicas		Storage Used (PB)			
Year	Count	Std. Dev.	Mean	Std. Dev.	2-replica reactive difference	3-replica reactive difference
10	2.00	0.00	5.96	0.0007	0.03 (1 x)	-3.00 (0.7 x)
20	2.00	0.00	5.96	0.0007	0.11 (1 x)	-2.99 (0.7 x)
30	2.00	0.00	5.96	0.0011	0.18 (1 x)	-2.99 (0.7 x)
40	2.00	0.00	5.96	0.0011	0.24 (1 x)	-2.98 (0.7 x)
50	2.00	0.00	5.96	0.0015	0.31 (1.1 x)	-2.98 (0.7 x)
60	2.00	0.00	5.96	0.0016	0.37 (1.1 x)	-2.98 (0.7 x)
70	2.00	0.00	5.96	0.0020	0.43 (1.1 x)	-2.97 (0.7 x)
80	2.00	0.00	5.96	0.0020	0.48 (1.1 x)	-2.98 (0.7 x)
90	2.00	0.00	5.97	0.0023	0.51 (1.1 x)	-2.98 (0.7 x)
100	2.00	0.00	5.99	0.0023	0.58 (1.1 x)	-2.94 (0.7 x)
2-replica 1.5-health SMART-aware proactive						
	Replicas		Storage Used (PB)			
Year	Count	Std. Dev.	Mean	Std. Dev.	2-replica reactive difference	3-replica reactive difference
10	2.00	0.00	5.95	0.0009	0.03 (1 x)	-3.01 (0.7 x)
20	2.00	0.00	5.96	0.0007	0.11 (1 x)	-2.99 (0.7 x)
30	2.00	0.00	5.96	0.0010	0.18 (1 x)	-2.98 (0.7 x)
40	2.00	0.00	5.96	0.0008	0.24 (1 x)	-2.98 (0.7 x)
50	2.00	0.00	5.96	0.0011	0.31 (1.1 x)	-2.98 (0.7 x)
60	2.00	0.00	5.96	0.0010	0.37 (1.1 x)	-2.97 (0.7 x)
70	2.00	0.00	5.96	0.0012	0.44 (1.1 x)	-2.97 (0.7 x)
80	2.00	0.00	5.97	0.0013	0.48 (1.1 x)	-2.98 (0.7 x)

Table 15. Continued.

90	2.00	0.00	5.97	0.0015	0.51 (1.1 x)	-2.98 (0.7 x)
100	2.00	0.00	5.99	0.0014	0.58 (1.1 x)	-2.94 (0.7 x)
2-replica 2.0-health SMART-aware proactive						
	Replicas		Storage Used (PB)			
Year	Count	Std. Dev.	Mean	Std. Dev.	2-replica reactive difference	3-replica reactive difference
10	2.13	0.006	6.38	0.019	0.45 (1.1 x)	-2.58 (0.7 x)
20	2.19	0.007	6.52	0.021	0.68 (1.1 x)	-2.43 (0.7 x)
30	2.20	0.007	6.56	0.021	0.78 (1.1 x)	-2.39 (0.7 x)
40	2.20	0.007	6.56	0.021	0.85 (1.1 x)	-2.37 (0.7 x)
50	2.20	0.007	6.57	0.022	0.92 (1.2 x)	-2.37 (0.7 x)
60	2.20	0.007	6.57	0.022	0.98 (1.2 x)	-2.37 (0.7 x)
70	2.20	0.007	6.57	0.020	1.05 (1.2 x)	-2.36 (0.7 x)
80	2.18	0.007	6.52	0.022	1.04 (1.2 x)	-2.42 (0.7 x)
90	2.17	0.007	6.49	0.020	1.03 (1.2 x)	-2.46 (0.7 x)
100	2.20	0.007	6.59	0.022	1.18 (1.2 x)	-2.34 (0.7 x)

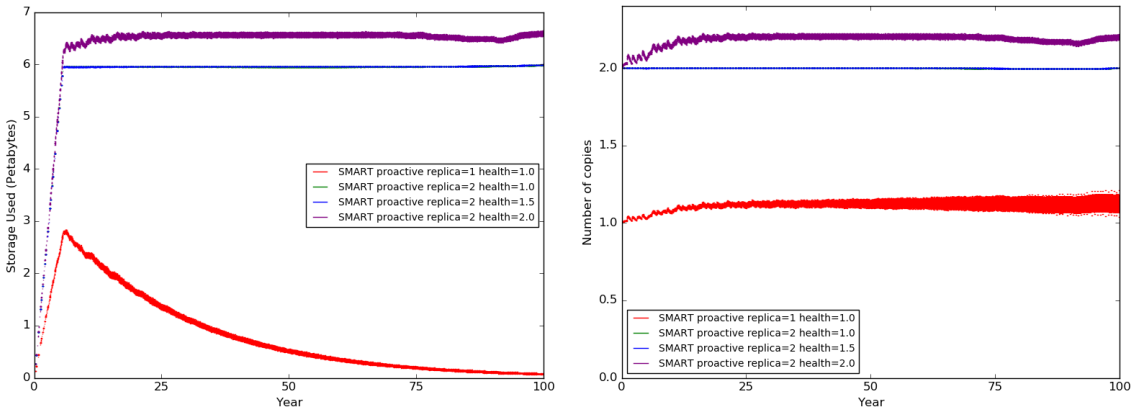


Figure 45. Storage and copies used over time for SMART error aware proactive policies. The left figure gives the storage (in petabytes) used over time by the three policies. The right figure gives the number of replicas per data object for data objects that survived to that time.

shows a tight grouping exactly around 2 replicas per data object with a standard deviation 3 to 4 orders of magnitude less than the mean. Both policies can effectively maintain the health requirement for the data objects through data migration alone without the need to resort to using three replicas leaving three petabytes of available storage.

The final policy, 2-replica 2.0-health, does require more than the minimum two replicas to guarantee that at least two replicas are on SMART error free disks requiring between 2.13 to 2.2 replicas per data object. While this necessitates more storage, the storage usage is still 2.62 to 2.41 petabytes less than what is used by three complete replicas and is over twice as reliable.

Normalized Magnitude of Data Loss

Having examined the reliability and storage usage attributes for all of the policies, this section interrelates the two attributes through the NOMDL metric. Table 16 provides the results of the NOMDL calculations for the full mission time of 100 years along with several of the intermediate results.

The replica count column for the simple and reactive policies is the specified replication level of the policy. As discussed in the methodology section, the replica count for the proactive policies is more complicated due to their dynamic nature. The replica count is taken from the maximum of the mean replica counts over the mission time. The storage used column is computed by taking the replica count of the policy and multiplying it by the total number of data objects, 3 million, inserted into the system and thus represents the storage that would have been used had no data objects ever been lost. The MDL, Magnitude of Data Loss, is the mean number of lost data objects multiplied by their size of 1 GB. Finally, the D column is the apparent amount of storage used by the data objects from the user's perspective combined with the amount of free storage usable for other purposes. The implicit apparent storage size is always three petabytes for having one replica of every data object.

There were not many surprises in the resulting NOMDL calculations. All three simple policies and the first reactive and proactive policies all did quite poorly, sharing relatively the same MDL score. Of these five policies, the lower required replicas' NOMDL scores were benefited by providing so much additional available storage to the user. Even taking this into consideration, though, there was still an order of magnitude difference between them and the next least reliability policy, the 2-replica reactive, and three orders between them and all of the 2-replica proactive policies.

Because the 2-replica, 1.0 and 1.5-health proactive policies utilize nearly the same storage quantities but the 1.5-health policy loses almost half of the data objects of the 1.0-health policy, their NOMDL values are proportionally as different as their reliability. While the 2-replica, 2.0-health policy is 1.7 times more reliability than the 1.5-health policy, its NOMDL is only 1.5 times better due to the extra space requirements.

The most interesting tidbit is the relationship between the 3-replica reactive and the 2-replica, 1.0-health policies. As seen in the reliability section, 3-replica reactive outperforms the 2-replica, 1.0-heatlh policy reliability wise (1.6 times worse) but their rankings for NOMDL are reversed due to the three petabytes saved by the proactive policy. This illustrates an important consideration for storage systems users, “are the three petabytes of saved storage worth more than 2,000 data objects?” Thankfully, in this scenario, the 1.5-health proactive policy saves as much storage and trumps the 3-replica reactive so no trade off would be required.

Table 16. NOMDL for SMART error age policies.

Policy	Lost Data Objects	Replica Count	Storage Used (PB)	Storage Free (PB)	MDL (PB)	D (PB)	NOMDL
1-replica Simple	2,998,257.41	1.00	3.00	6.22	3.00	9.22	0.33
2-replica Simple	2,996,218.15	2.00	6.00	3.22	3.00	6.22	0.48
3-replica Simple	2,994,430.21	3.00	9.00	0.22	2.99	3.22	0.93
1-replica Reactive	2,999,322.28	1.00	3.00	6.22	3.00	9.22	0.33
2-replica Reactive	292,432.07	2.00	6.00	3.22	0.29	6.22	0.047
3-replica Reactive	2,955.67	3.00	9.00	0.22	0.0030	3.22	0.00092
1-replica 1.0-health	2,933,185.40	1.13	3.38	5.84	2.93	8.84	0.33
2-replica 1.0-health	4,636.39	2.00	6.00	3.22	0.0046	6.22	0.00075
2-replica 1.5-health	2,359.19	2.00	6.00	3.22	0.0024	6.22	0.00038
2-replica 2.0-health	1,396.93	2.20	6.59	2.62	0.0014	5.62	0.00025

Bandwidth Used

Simple

The stacked bar chart in Figure 46 provides the breakdown of the mean data flows of the simple policies organized by policy and partitioned by the type of data traffic. The bus represents traffic that only flowed over a bus within a depot indicating single depot disk-to-disk replication. While the local area traffic is traffic that travels via a bus and intra-site links for single site depot-to-depot replication. Finally, the wide area traffic is for inter-site data copying and has data traversing buses, site links and of course wide area hops.

1-replica simple policy is somewhat unique in that it is the only policy where all of the traffic was limited solely to local area traffic with three petabytes being used. The policy created a single allocation for every data object at the corresponding site for the data source client and thus limited all data traffic to from the client to the allocation. With three million 1 GB data objects the total traffic flow was exactly three petabytes.

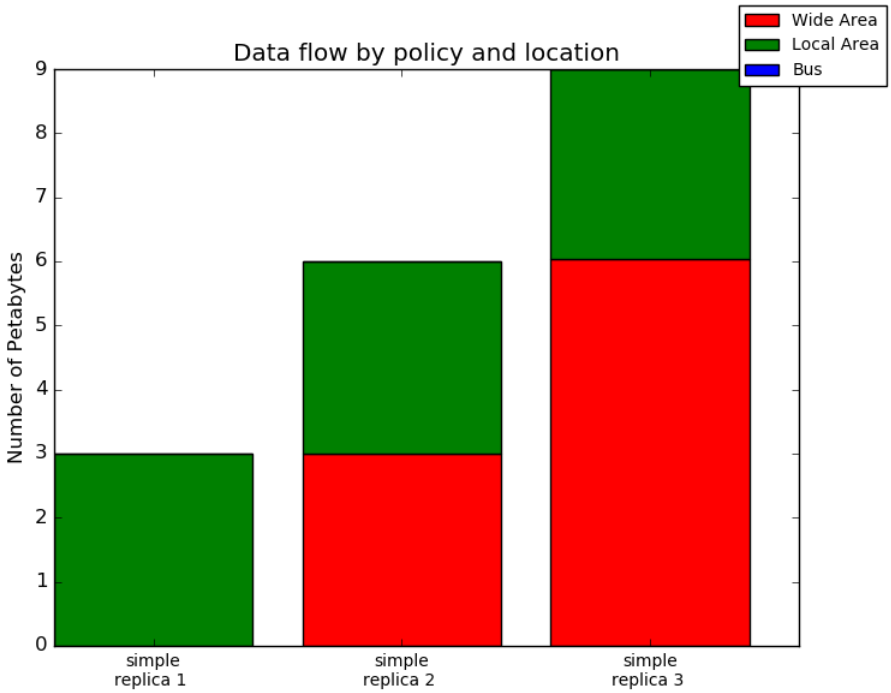


Figure 46. Simple policy data flows run against the SMART error age model. This stacked bar plot provides the mean data flows for each of the simple policies categorized by the type of traffic: bus only, local and bus only, and wide area.

The data flows for the 2 and 3-replica policy are as expected. They both expend roughly three petabytes of local area only bandwidth to perform the initial upload of the dataset, the same as the 1-replica policy. Then use another three petabytes of wide area bandwidth for each additional inter-site replica requested of them. The 3-replica policy does utilize an extra 40 terabytes in the wide area and 15 gigabytes worth of bus-only traffic to handle some data shuffling as storage state approaches maximum capacity. These results are expanded upon in Table 17 below.

Reactive

The data flows for the 2 and 3-replica reactive policies are around eight times higher, as seen in Figure 47, than their simple policy counterparts due to replicas between being replaced as disks fail. The single replica reactive policy is essentially the same as the simple 1-replica policy for obvious reasons. The 2 and 3-replica policies are having to devote 87 to 88% of their total bandwidth expenditure to maintain the data objects to the best of their abilities. Almost the entirety of the additional bandwidth usage, 41 and 65 petabytes, is traversing the wide area as there are no local, same site or same depot replicas available

Table 17. Simple policies data flow for the SMART error age model.

Simple Policy Data Flows				
1-Replica Simple				
Traffic	Mean	Std. Dev	2-replica reactive difference	3-replica reactive difference
Bus	0.0	0.0	0.00 (x)	0.00000(x)
Local	3.0	0.0	0.00 (1.00 x)	0.00031(1.00 x)
Wide	0.0	0.0	-44.57 (0.00 x)	-71.28977(0.00 x)
Total	3.0		-44.57 (0.06 x)	-71.28945(0.40 x)
2-replica simple				
Traffic	Mean	Std. Dev	2-replica reactive difference	3-replica reactive difference
Bus	0.00	0.00	(x)	(x)
Local	3.00	0.00	0.00 (1.00 x)	0.00(1.00 x)
Wide	3.00	0.00	-41.57 (0.07 x)	-68.29(0.04 x)
Total	6.00		-41.57 (0.10 x)	-68.29(0.08 x)
3-replica simple				
Traffic	Mean	Std. Dev	2-replica reactive difference	3-replica reactive difference
Bus	0.00	0.00	0.00 (x)	0.00(1.00 x)
Local	2.96	0.00	-0.04 (1.00 x)	-0.04(0.08 x)
Wide	6.04	0.00	-38.52 (0.10 x)	-65.25(0.10 x)
Total	9.00		-38.57 (0.20 x)	-65.29(1.00 x)

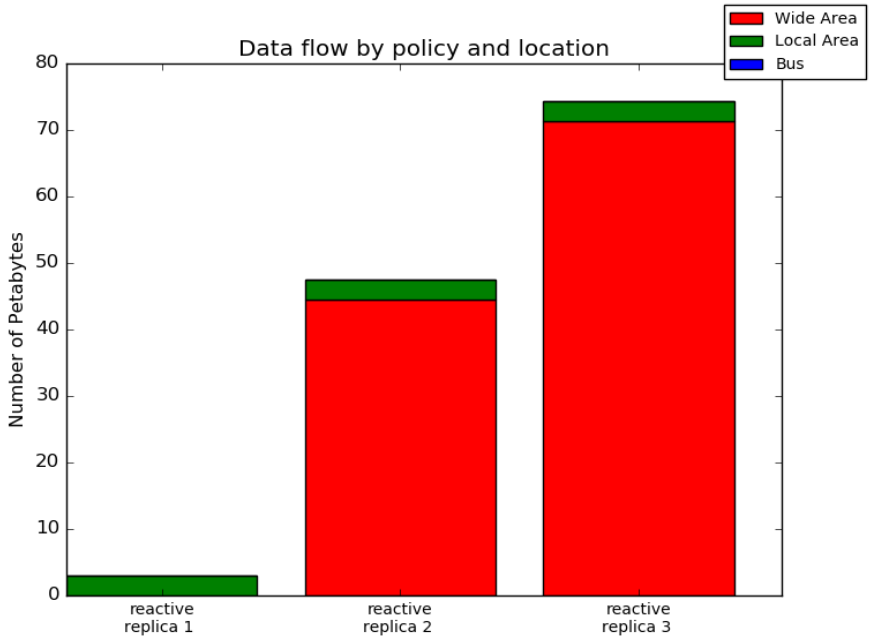


Figure 47. Data flow reactive policies run against SMART error age model.

making these replacements more expensive and time-consuming. All of the results are given in below in Table 18.

Table 18. Reactive policy data flows for the SMART error age failure model.

Reactive Policy Data Flows				
1-Replica Reactive				
Traffic	Mean	Std. Dev	2-Replica Reactive Difference	3-Replica Reactive Difference
Bus			(x)	(x)
Local	3.00	0.0	(1 x)	0.00(1 x)
Wide		0.0	-44.57 (0.00 x)	-71.29(0.00 x)
Total	3.00	0.0	-44.57 (0.06 x)	-71.29(0.04 x)
2-Replica Reactive				
Traffic	Mean	Std. Dev	2-Replica Reactive Difference	3-Replica Reactive Difference
Bus			(x)	(x)
Local	3.00		(1 x)	0.00(1.00 x)
Wide	44.57	0.12	(1 x)	-26.72(0.60 x)
Total	47.57		(1 x)	-26.72(0.60 x)
3-Replica Reactive				
Traffic	Mean	Std. Dev	2-Replica Reactive Difference	3-Replica Reactive Difference
Bus			(x)	(x)
Local	3.00	0.00	0.00 (1 x)	(1 x)
Wide	71.29	0.11	26.72 (1.6 x)	(1 x)
Total	74.29		26.72 (1.6 x)	(1 x)

SMART Error-Aware Proactive

The SMART proactive policies are better at preserving bandwidth usage especially between sites as seen in Figure 48. The 1-replica proactive policy has the same three petabytes of local area bandwidth usage as the reactive for the initial upload but uses a mean of 3.56 petabytes of bus-only and 1.23 terabytes of wide area to maintain the data objects' health rating providing the modest gain in reliability seen previously. The real improvement, however, is for the three 2-replica proactive policies. Not only do they provide reliability that is on the same order of magnitude as the 3-replica reactive but save up to 25 petabytes of total bandwidth. The 2-replica, 1.0-health policy uses on average only 2.1 petabytes more bandwidth than the 2-replica reactive policy while safeguarding the data orders of magnitude better. The 2-replica, 1.5 and 2.0-health policies which proved to be consistently more reliable than the 3-replica reactive also used less bandwidth, approximately 18 petabytes less than the 3-replica reactive and 8

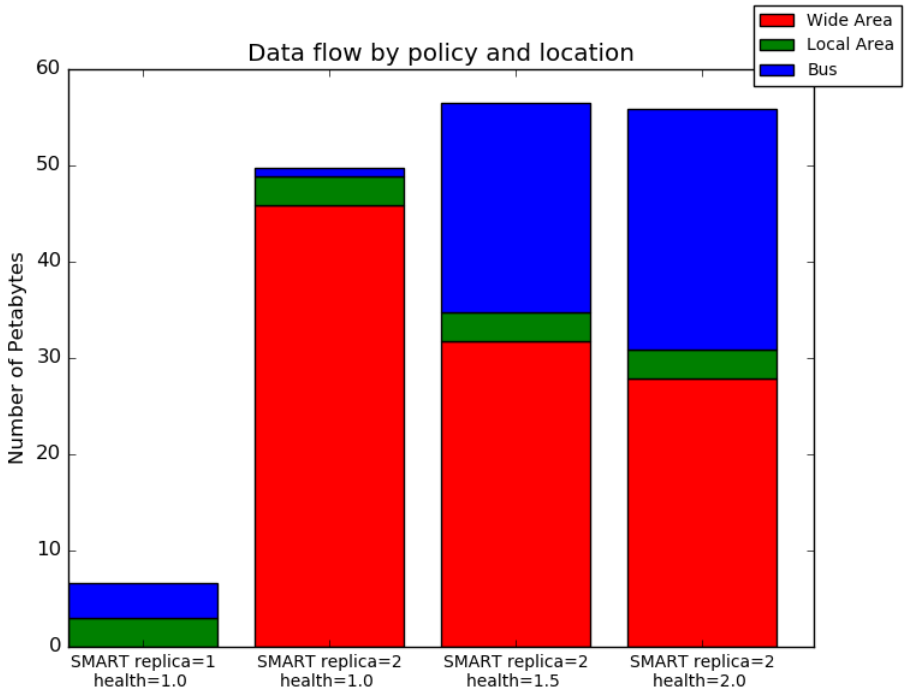


Figure 48. Data flow proactive policies run against SMART error age model.

petabytes more than 2-replica reactive. In fact, the 2.0-health proactive policy uses slightly less bandwidth than the 1.5-health policy while continuing to improve the overall reliability.

Not only is the total bandwidth reduced in comparison to the 3-replica reactive but the wide area bandwidth is even more drastically reduced with the policies being able to migrate data before predicted failure to more reliable disks on the same depot utilizing only bus traffic. Each of the three, 2-replica proactive policies uses only three petabytes of local area only bandwidth, necessary for the initial data upload. There is an inverse relationship between health and wide area traffic. As the health requirement increases, the wide area traffic decreases from nearly 46 petabytes to 28 petabytes, a 39% decrease. The final proactive policy uses only 28 petabytes over the wide area. This is 39% and 63% of the usage of the 3 and 2-replica reactive policies, respectively.

Transitioning the policy's health metric from 1.0 to 1.5 and 2.0 results in substantially more traffic across the buses from nearly 0 to 25 petabytes for the 2.0-health metric. Interestingly enough, most of the increase comes from the 1.0 to 1.5 jump while the bus traffic increase from 1.5 to 2.0 is only 3.4 petabytes, a 14% change.

What these results illustrate is that the proactive policy can successfully use SMART errors as a near-term measure of disk reliability to reduce the need for wide area replication at the cost of more bus-only traffic. In fact, bus-only

traffic is the sole data flow to increase as local area restricted traffic remains flat being used mostly for initial client upload. Despite the increase in bus traffic, the total traffic diminishes as the costlier wide area traffic falls faster than the bus traffic increases. The limited increase in bus traffic also demonstrates that the proactive policy does not result in replication trashing between disks to maintain the health setting. These results are summarized Table 19.

SMART Error Count

In this section, the results from the analysis of the simple, reactive and SMART Error aware policies are discussed in regards to the SMART Error count failure model.

Data Survival

Simple

A summary of the results from the analysis of the three simple policies on the SMART Error count failure model are presented in Table 20. From Figure 49, the data loss curves for the Count model are nearly identical to the simple policy runs against the Age model. Although, as shown in the table, the loss rates are slightly less for all of the simple policies with this failure model. The more replicas the simple policy initially creates, the more gradual the data loss. Still, by the end of the 100 years, there is only an average of 9,678 data objects separating the 1 and 3-replia policies with them having lost 99.83% to 99.51% of the data respectively.

The largest difference between the policies is during the 10 to 20-year period with year 13 typically possessing the greatest disparity of 1,154,892 data objects. Between 40 to 60 years the policies' reliabilities start rapidly converging before becoming nearly identical by year 80. Unlike the Age model, no simulation instances resulted in complete data loss for any of the three simple policies. The greatest loss for each policy from 1 to 3 replicas was 2,998,879, 2,999,774 and 2,999,540 in that order. The final results were fairly tightly coupled for each of the policies having a relatively small standard deviation and no significant outliers.

Reactive

For the three reactive policies, the same general trends and outcomes are observed as for the previous model. As shown in Figure 50 and Figure 51, every additional replica that the reactive policy is allowed to maintain results in significantly better reliability, rising from 0.164% to 93.15% to 99.94%. The 1-replica reactive is no better than its 1-replica simple counterpart having virtually the same CDF's making it woefully inadequate for maintaining data even over the short-term. Even before the populate phase of the simulations is complete, 28% of the total data objects have already been lost. However, by allowing for replacement copies over the course of the mission lifetime, the 2 and 3-replica

Table 19. Proactive policy data flows on the SMART error age model.

Proactive Data Flows				
1-replica 1.0-health				
Traffic	Mean	Std. Dev	2-replica reactive difference	3-replica reactive difference
Bus	3.56	0.06	3.56 (x)	3.56(1 x)
Local	3.00	0.00	0.00 (1 x)	0.00(0.00002 x)
Wide	0.00	0.00	-44.57 (0.00003 x)	-71.29(0.09 x)
Total	6.56		-41.00 (0.1 x)	-67.73(1 x)
2-replica 1.0-health				
Traffic	Mean	Std. Dev	2-replica reactive difference	3-replica reactive difference
Bus	0.80	0.01	0.80 (x)	0.80(x)
Local	3.00	0.00	0.00 (1.0 x)	0.00(1 x)
Wide	45.89	0.07	1.32 (1.0 x)	-25.40(0.6 x)
Total	49.69		2.12 (1.0 x)	-24.60(0.7 x)
2-replica 1.5-health				
Traffic	Mean	Std. Dev	2-replica reactive difference	3-replica reactive difference
Bus	21.64	0.16	21.64 (x)	21.64(x)
Local	3.00	0.00	0.00 (1 x)	0.00(1 x)
Wide	31.80	0.10	-12.77 (0.7 x)	-39.49(0.4 x)
Total	56.44		8.87 (1.2 x)	-17.85(0.8 x)
2-replica 2.0-health				
Traffic	Mean	Std. Dev	2-replica reactive difference	3-replica reactive difference
Bus	25.03	0.65	25.03 (x)	25.03(x)
Local	3.00	0.00	0.00 (0.6 x)	0.00(1 x)
Wide	27.85	0.08	-16.72 (1.2 x)	-43.44(0.4 x)
Total	55.88		8.31 (1.0 x)	-18.41(0.8 x)

Table 20. Data Loss by the simple policies against the SMART Error Count Model

Data Object Loss by Policy				
1-Replica Simple				
Year	Mean	Std. Dev.	2-Replica Reactive difference	3- Replica Reactive difference
5	842,424	14,866	830,622 (71.4x)	842,315(7771.9x)
10	1,447,185	17,905	1,423,803 (61.9x)	1,446,975(6911.5x)
20	2,196,303	18,502	2,150,773 (48.2x)	2,195,903(5482.8x)
30	2,583,408	14,493	2,516,488 (38.6x)	2,582,822(4407.1x)
40	2,783,947	10,820	2,695,854 (31.6x)	2,783,175(3603x)
50	2,888,184	7,768	2,779,159 (26.5x)	2,887,223(3007x)
60	2,942,232	5,755	2,812,471 (22.7x)	2,941,082(2558.6x)
70	2,970,179	4,299	2,819,810 (19.8x)	2,968,837(2214x)
80	2,984,507	3,040	2,813,665 (17.5x)	2,982,975(1948x)
90	2,991,881	2,278	2,800,690 (15.6x)	2,990,157(1735.6x)
100	2,994,961	1,810	2,789,605 (14.6x)	2,993,101(1610.5x)
2-Replica Simple				
Year	Mean	Std. Dev.	2-Replica Reactive difference	3- Replica Reactive difference
5	236,467	8,478	224,665 (20x)	236,358(2181.6x)
10	698,921	18,861	675,540 (29.9x)	698,712(3337.9x)
20	1,610,162	28,100	1,564,631 (35.4x)	1,609,761(4019.6x)
30	2,226,662	24,363	2,159,742 (33.3x)	2,226,076(3798.5x)
40	2,584,469	20,131	2,496,375 (29.3x)	2,583,696(3344.8x)
50	2,780,730	15,066	2,671,705 (25.5x)	2,779,769(2895.1x)
60	2,884,930	11,034	2,755,170 (22.2x)	2,883,781(2508.8x)
70	2,940,339	8,285	2,789,970 (19.6x)	2,938,998(2191.8x)
80	2,968,997	6,409	2,798,155 (17.4x)	2,967,464(1937.8x)
90	2,983,980	4,566	2,792,789 (15.6x)	2,982,256(1731.1x)
100	2,990,129	3,758	2,784,773 (14.6x)	2,988,269(1607.9x)
3-Replica Simple				
Year	Mean	Std. Dev.	2-Replica Reactive difference	3- Replica Reactive difference
5	66,138	3,660	54,336 (5.6x)	66,030(610.2x)
10	336,355	13,801	312,973 (14.4x)	336,146(1606.4x)
20	1,175,865	28,623	1,130,335 (25.8x)	1,175,465(2935.4x)
30	1,915,884	31,307	1,848,964 (28.6x)	1,915,298(3268.3x)
40	2,399,729	25,757	2,311,636 (27.2x)	2,398,957(3105.7x)
50	2,678,130	20,449	2,569,105 (24.6x)	2,677,169(2788.3x)
60	2,830,306	16,582	2,700,546 (21.8x)	2,829,156(2461.3x)
70	2,911,762	12,582	2,761,392 (19.4x)	2,910,420(2170.5x)
80	2,954,569	9,042	2,783,727 (17.3x)	2,953,037(1928.4x)
90	2,976,575	6,135	2,785,384 (15.6x)	2,974,852(1726.8x)
100	2,985,283	4,976	2,779,927 (14.5x)	2,983,423(1605.2x)

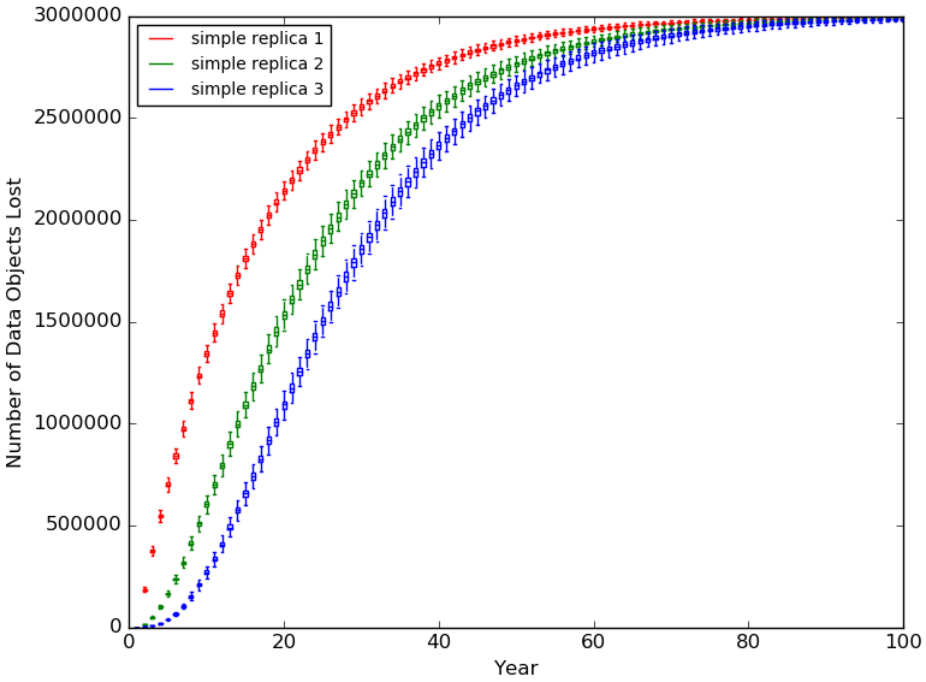


Figure 49. Simple policy reliability against the SMART error count model. The cumulative data object loss for every run of the simple policy on the SMART Error Count Model grouped by the replication setting of the policy as a time series of Tukey box plots.

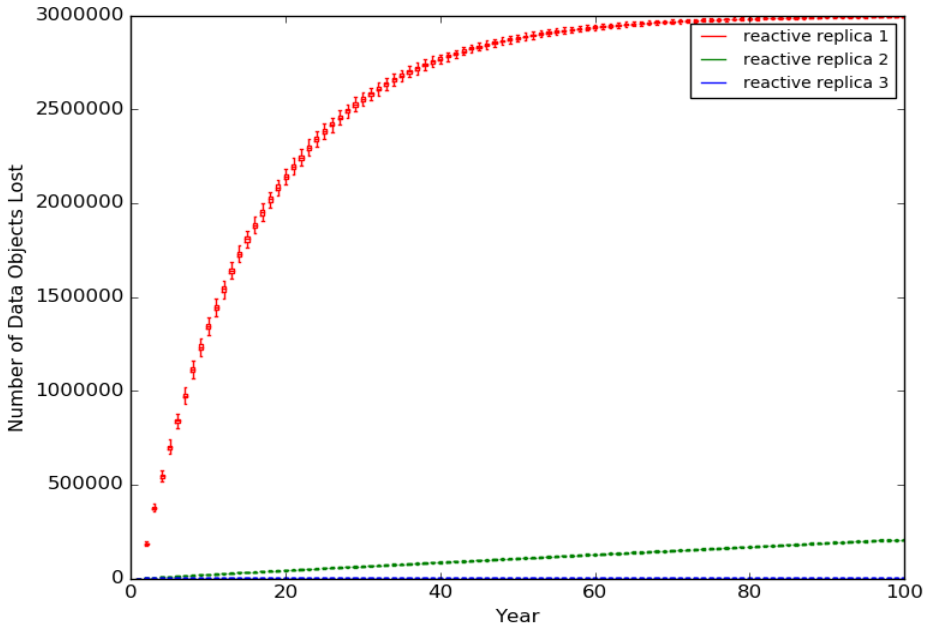


Figure 50. Reactive policy reliability on the SMART error count model.

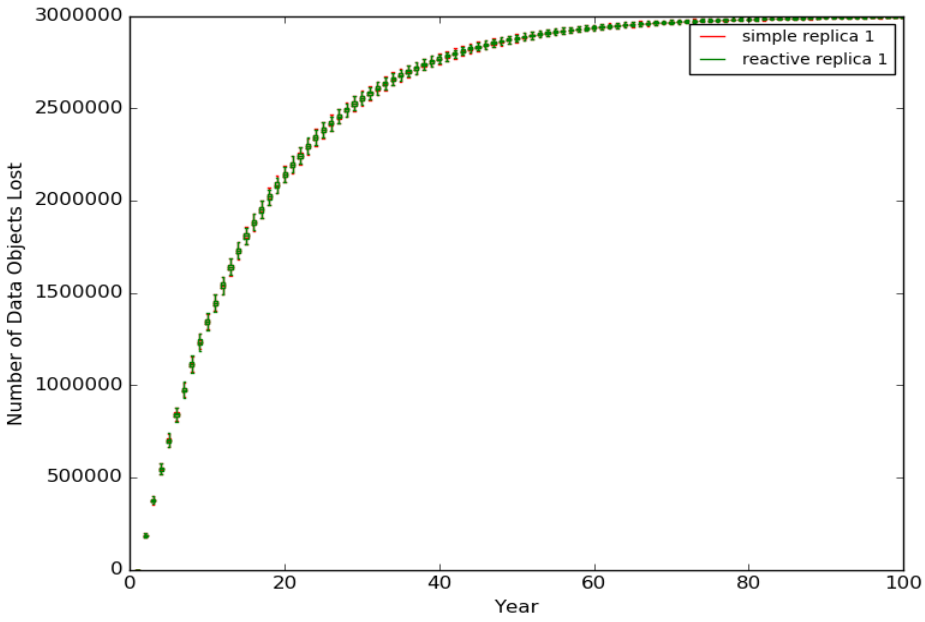


Figure 51. 1-replica reactive vs. 1-replica simple policy reliability on the SMART error count model.

reactive policies prove to be 10's to 1000's times more reliability than the simple policies confirming something already well understood in data redundancy. The 2-replica reactive surpassed expectations managing to retain the majority of the data objects, and 3-replica lost only 1,860 of the data objects. Unlike the simple policies, these policies do not converge and remain separate from the outset. The results for the three reactive policies are provided in Table 21.

SMART Error-Aware Proactive

Figure 52, Figure 53, Figure 54 and Table 22 provide the results from the proactive policies runs on the SMART Error Count model. The reliability performance of the 1-replica, 1.0-health proactive policy is poor. It is a clear outlier from the other proactive policies, and its performance lies between the 1 and 2-replica reactive policies being closer to former than latter, as revealed in Figure 53. The 2-replica reactive is 42 to 14 times more reliable than this proactive policy, demonstrating that a single replica proactive policy for this failure model is not sufficient. While failing to impress, the 1-replica proactive policy does still outperform the other single replica policies. This fact at least does back up part of this work's assertion that proactive policies with some knowledge of the storage media state can benefit reliability.

Table 21. Data object loss by the reactive policies against the SMART error count model

Data Object Loss by Policy				
1-Replica Reactive				
Year	Mean	Std. Dev.	2-Replica Reactive Difference	3-Replica Reactive Difference
5	840,517	15,313	828,715 (71.2x)	840,408(7754.3x)
10	1,446,418	18,218	1,423,036 (61.9x)	1,446,208(6907.9x)
20	2,194,527	17,020	2,148,996 (48.2x)	2,194,126(5478.4x)
30	2,582,954	13,797	2,516,034 (38.6x)	2,582,367(4406.3x)
40	2,784,614	10,342	2,696,520 (31.6x)	2,783,841(3603.8x)
50	2,888,512	7,807	2,779,488 (26.5x)	2,887,552(3007.3x)
60	2,942,377	5,732	2,812,616 (22.7x)	2,941,227(2558.7x)
70	2,970,353	4,141	2,819,983 (19.8x)	2,969,011(2214.2x)
80	2,984,579	3,053	2,813,737 (17.5x)	2,983,047(1948x)
90	2,992,146	2,211	2,800,955 (15.7x)	2,990,423(1735.8x)
100	2,995,080	1,707	2,789,725 (14.6x)	2,993,221(1610.5x)
2-Replica Reactive				
Year	Mean	Std. Dev.	2-Replica Reactive Difference	3-Replica Reactive Difference
5	11,802	499	(1x)	11,694(108.9x)
10	23,382	846	(1x)	23,172(111.7x)
20	45,530	1,122	(1x)	45,130(113.7x)
30	66,920	1,367	(1x)	66,334(114.2x)
40	88,093	1,569	(1x)	87,321(114x)
50	109,025	1,663	(1x)	108,064(113.5x)
60	129,761	1,741	(1x)	128,611(112.8x)
70	150,369	1,743	(1x)	149,028(112.1x)
80	170,842	1,678	(1x)	169,310(111.5x)
90	191,191	1,626	(1x)	189,467(110.9x)
100	205,356	1,762	(1x)	203,496(110.4x)
3-Replica Reactive				
Year	Mean	Std. Dev.	2-Replica Reactive Difference	3-Replica Reactive Difference
5	108	13	-11,694 (0.009x)	(1x)
10	209	18	-23,172 (0.009x)	(1x)
20	401	24	-45,130 (0.009x)	(1x)
30	586	32	-66,334 (0.009x)	(1x)
40	773	40	-87,321 (0.009x)	(1x)
50	961	49	-108,064 (0.009x)	(1x)
60	1,150	55	-128,611 (0.009x)	(1x)
70	1,342	63	-149,028 (0.009x)	(1x)
80	1,532	71	-169,310 (0.009x)	(1x)
90	1,724	79	-189,467 (0.009x)	(1x)
100	1,860	86	-203,496 (0.009x)	(1x)

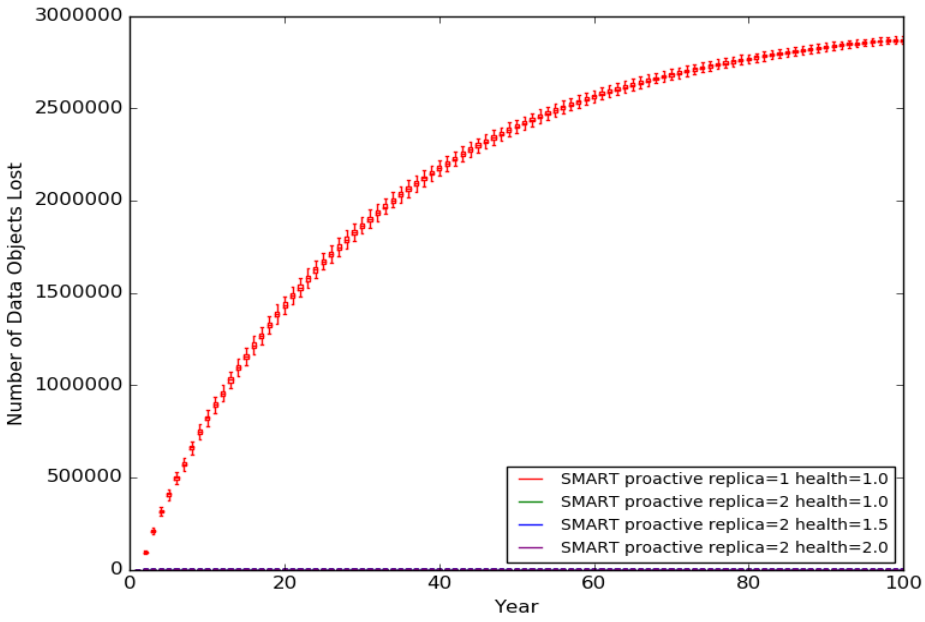


Figure 52. SMART error proactive policy reliability against the SMART error count model.

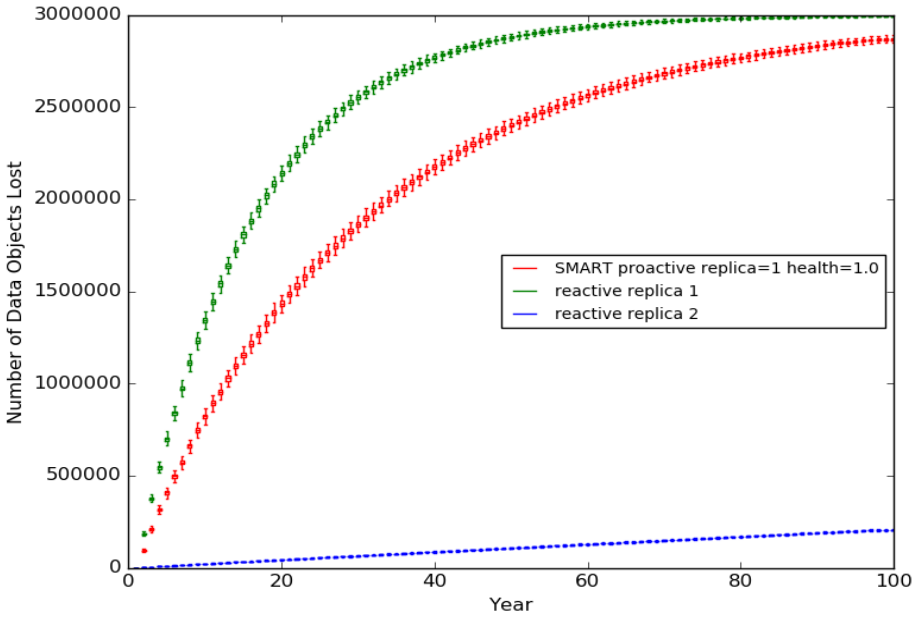


Figure 53. 1-replica, 1.0-health SMART error proactive vs. reactive policies.

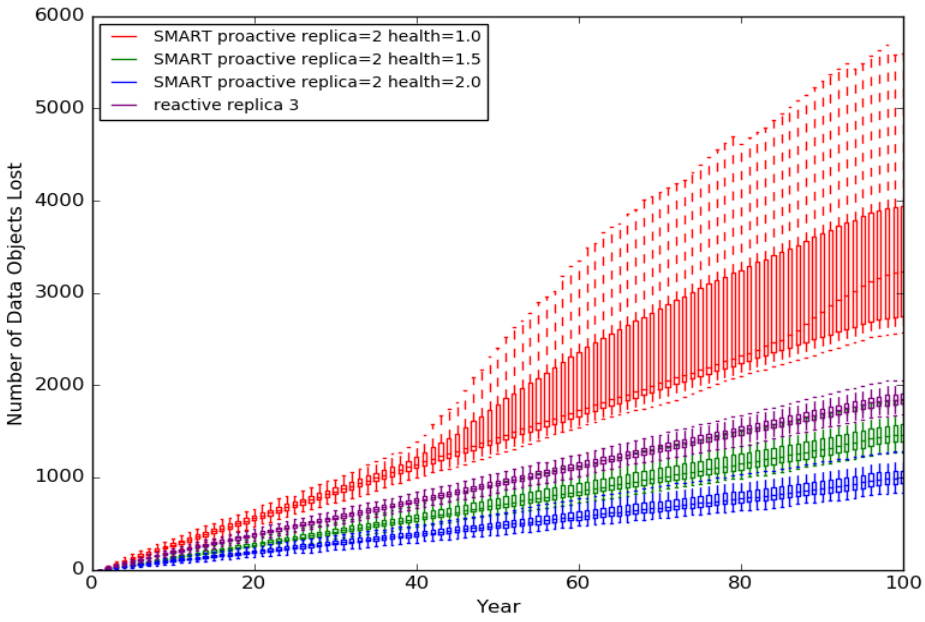


Figure 54. SMART error proactive policies vs. 3-replica reactive on the SMART Error Count model.

In Figure 54, the three 2-replica proactive policies are shown in more detail along with the 3-replica reactive for comparison. As the health metric increases, the total data loss decreases per policy. The lowest health rated policy ends up losing 2.29 and 3.42 times as many data objects as the next two proactive policies. All of the 2-replica proactive policies do substantially better than the 2-replica reactive, and only the 1.0-health policy does worse the than 3-replica reactive. The 2-replica reactive completes with an average loss factor of 59 to 200 times more than these proactive policies. While the 3-replica reactive policy beats the 1.0-health policy by a factor of 2, the two remaining policies averaged 1.25 to 2 times better. The 3-replica replica reactive does have instances that overlap with the 2-replica, 1.5-health proactive but is limited to only the upper 1.5 IQR whisker range of the proactive policy. The interquartile box for the 2-replica, 1.5-health policy is clear of any overlap with the 3-replica reactive indicating decisively that the proactive policy is more reliable. The 2.0-health policy, in turn, outperforms all of the other policies. There is a small degree of overlap between the 1.5-health and 2.0-health policies as indicated by the minor intersection of the lower whisker tip and the upper whisker tip of the two policies' boxplots. However, this only occurs along the very edges of the 1.5 IQR of the lower and upper quantiles.

From the previous figures, the 2 and 3-replica reactive as well as all of the 2-replica proactive policies appear to lose data objects linearly. This made further investigation possible by conducting a regression analysis on their mean

Table 22. Data object loss by the proactive policies against the SMART error count model

Data Object Loss by Policy				
1-Replica 1.0-health SMART error proactive				
Year	Mean	Std. Dev.	2-Replica Reactive Difference	3-Replica Reactive Difference
5	496,540	12,357	484,738 (42.1x)	496,431 (4580.9x)
10	892,884	16,837	869,502 (38.2x)	892,674 (4264.3x)
20	1,485,234	19,590	1,439,704 (32.6x)	1,484,834 (3707.7x)
30	1,901,589	18,016	1,834,669 (28.4x)	1,901,002 (3243.9x)
40	2,202,890	16,764	2,114,797 (25x)	2,202,117 (2851x)
50	2,420,997	14,283	2,311,973 (22.2x)	2,420,037 (2520.6x)
60	2,579,110	13,325	2,449,349 (19.9x)	2,577,960 (2242.8x)
70	2,693,866	11,517	2,543,497 (17.9x)	2,692,525 (2008.1x)
80	2,777,248	9,992	2,606,406 (16.3x)	2,775,716 (1812.7x)
90	2,838,343	8,660	2,647,152 (14.8x)	2,836,620 (1646.6x)
100	2,870,772	7,893	2,665,417 (14x)	2,868,912 (1543.7x)
2-Replica 1.0-health SMART error proactive				
Year	Mean	Std. Dev.	2-Replica Reactive Difference	3-Replica Reactive Difference
5	155	21	-11,647 (0.01x)	46 (1.4x)
10	305	43	-23,077 (0.01x)	95 (1.5x)
20	614	114	-44,916 (0.01x)	214 (1.5x)
30	942	207	-65,977 (0.01x)	356 (1.6x)
40	1,297	313	-86,796 (0.01x)	524 (1.7x)
50	1,668	426	-107,357 (0.02x)	707 (1.7x)
60	2,045	531	-127,715 (0.02x)	895 (1.8x)
70	2,428	634	-147,941 (0.02x)	1,087 (1.8x)
80	2,813	726	-168,029 (0.02x)	1,281 (1.8x)
90	3,206	808	-187,985 (0.02x)	1,482 (1.9x)
100	3,485	866	-201,870 (0.02x)	1,625 (1.9x)
2-Replica 1.5-health SMART error proactive				
Year	Mean	Std. Dev.	2-Replica Reactive Difference	3-Replica Reactive Difference
5	74	17	-11,728 (0.006x)	-34 (0.7x)
10	149	32	-23,233 (0.006x)	-61 (0.7x)
20	296	65	-45,234 (0.007x)	-104 (0.7x)
30	447	92	-66,473 (0.007x)	-139 (0.8x)
40	600	112	-87,494 (0.007x)	-173 (0.8x)
50	753	133	-108,271 (0.007x)	-207 (0.8x)
60	907	155	-128,854 (0.007x)	-243 (0.8x)
70	1,063	177	-149,306 (0.007x)	-278 (0.8x)
80	1,217	201	-169,625 (0.007x)	-315 (0.8x)
90	1,377	220	-189,814 (0.007x)	-347 (0.8x)
100	1,523	234	-203,833 (0.007x)	-337 (0.8x)

Table 22. Continued.

2-Replica 2.0-health SMART error proactive				
Year	Mean	Std. Dev.	2-Replica Reactive Difference	3-Replica Reactive Difference
5	58	12	-11,744 (0.005x)	-50(0.5x)
10	110	21	-23,271 (0.005x)	-99(0.5x)
20	210	37	-45,320 (0.005x)	-190(0.5x)
30	311	50	-66,609 (0.005x)	-275(0.5x)
40	407	59	-87,687 (0.005x)	-366(0.5x)
50	506	69	-108,518 (0.005x)	-454(0.5x)
60	606	77	-129,155 (0.005x)	-544(0.5x)
70	707	88	-149,662 (0.005x)	-635(0.5x)
80	809	97	-170,033 (0.005x)	-723(0.5x)
90	916	107	-190,275 (0.005x)	-807(0.5x)
100	1,019	113	-204,336 (0.005x)	-840(0.5x)

loss values. Table 23 provides these results along with their R^2 confidence levels for verification of the trend lines. With such strong confidence levels, the linearity of data loss in respect to the variable time (in years) for these policies can be safely assumed. The derivatives of the CDF's, the data object loss rates, for these policies are constant over time with the slope supplying the rate values.

Storage Used & Copies Used

Simple

The storage utilization and replica count for the three simple policy configurations are presented in Figure 55 for the 100-year mission timeframe. The storage utilization for all three policies follow the same basic trend. During the initial populate phase the storage utilization rises linearly with the steepness of the curves being dictated by the number of replicas expected for each policy. As the policy simulations' approach the end of the populate phase, around 5.66 years, their storage utilization peak falling short of their expected storage targets of 3, 6 and 9 petabytes. Due to the loss of data objects during the populate phase, the policies peak slightly before reaching the end of the phase around 5.66 years and then with no further data objects being injected the storage utilization exponential decays with a decay rate of 0.066. The peak utilization would be 2.527, 5.055, 7.584 petabytes which are 84% of their storage targets.

The starting number of copies per surviving data object is as expected with 1, 2 and 3 copies. Then as time proceeds, the 2 and 3-replica simple policy start dropping off as enough replicas have been lost to pull down the global averages. The 1-replica policy is consistently at one copy since the moment a replica for a data object is lost so is the data object and is thus excluded from the surviving set.

Table 23. Linear regression and R^2 confidence.

Policy	Regression Line	R^2	Loss Rate
2-Replica Reactive	2093.8t + 3341.1	0.99961	2093.8
3-Replica Reactive	18.901t + 18.186	0.99991	18.901
2-Replica 1.0-health	36.173t - 133.77	0.99826	36.173
2-Replica 1.5-health	15.409t - 11.901	0.99975	15.409
2-Replica 2.0-health	10.107t + 4.9519	0.99956	10.107

Table 24 provides further details on the storage and replica count of the three policies.

Reactive

The storage utilization for the three reactive policies is given in the left graph of Figure 56 and Table 25. The 1-replica reactive policy has the same exponential decay rate of .066 as the simple policies after peaking in the populate phase at 2.51 petabytes. The 2-replica reactive reaches 5.93 petabytes before starting a linear downwards taper of -4 terabytes per year. The 3-replica reactive case is more complicated. While storage usage mostly appears to flat line at 5.67 years after peaking at 8.92 petabytes, it starts an annual decline of 2 gigabytes. Then, at year 59, the trend reverses with a modest increase in storage of 1 terabyte a year until the conclusion of the simulation.

The right graph in Figure 56 and Table 25 give the replica counts for the surviving data objects at 1.99 and 2.98 falling just short of the replication targets the for 2 and 3-replica policies. The reactive policies are nearly constantly having to compensate for disk failure, so there are always a few surviving data

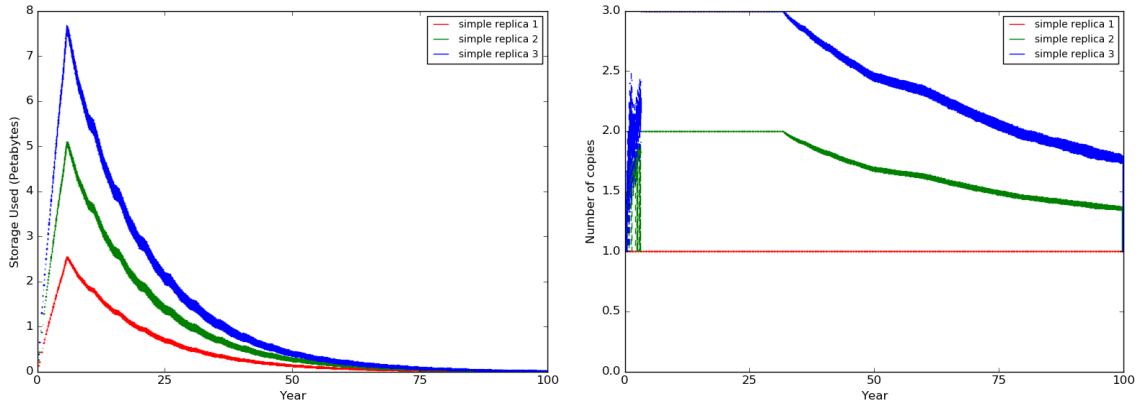


Figure 55. Storage and copies used over time for the simple policies. The left figure gives the storage (in petabytes) used over time by the three simple policies. The right figure gives the number of replicas per data object for data objects that survived to that time.

Table 24. Storage utilization and replica count for the simple policies with the SMART error count model.

1-replica simple						
	Replicas		Storage Used (PB)			
Year	Count	Std. Dev.	Mean	Std. Dev.	2-Replica Reactive Difference	3-Replica Reactive Difference
10	1.00	0.00	1.85	0.019	-4.06 (0.3 x)	-7.06 (0.2 x)
20	1.00	0.00	0.96	0.020	-4.90 (0.2 x)	-7.94 (0.1 x)
30	1.00	0.00	0.50	0.016	-5.32 (0.09 x)	-8.40 (0.06 x)
40	1.00	0.00	0.26	0.012	-5.51 (0.04 x)	-8.64 (0.03 x)
50	1.00	0.00	0.13	0.009	-5.59 (0.02 x)	-8.76 (0.01 x)
60	1.00	0.00	0.07	0.006	-5.62 (0.01 x)	-8.82 (0.008 x)
70	1.00	0.00	0.04	0.005	-5.61 (0.006 x)	-8.86 (0.004 x)
80	1.00	0.00	0.02	0.003	-5.59 (0.003 x)	-8.88 (0.002 x)
90	1.00	0.00	0.01	0.003	-5.57 (0.002 x)	-8.91 (0.001 x)
100	1.00	.00	0.01	0.002	-5.55 (0.001 x)	-8.94 (0.0007 x)
2-replica simple						
	Replicas		Storage Used (PB)			
Year	Count	Std. Dev.	Mean	Std. Dev.	2-Replica Reactive Difference	3-Replica Reactive Difference
10	2.00	.0001	3.70	0.040	-2.22 (0.6 x)	-5.22 (0.4 x)
20	2.00	.0002	1.91	0.041	-3.95 (0.3 x)	-6.99 (0.2 x)
30	2.00	.0002	0.99	0.032	-4.82 (0.2 x)	-7.91 (0.1 x)
40	1.83	.0068	0.51	0.024	-5.26 (0.09 x)	-8.38 (0.06 x)
50	1.68	.0075	0.27	0.018	-5.46 (0.05 x)	-8.63 (0.03 x)
60	1.63	.0081	0.14	0.013	-5.55 (0.02 x)	-8.75 (0.02 x)
70	1.53	.0081	0.07	0.009	-5.58 (0.01 x)	-8.82 (0.008 x)
80	1.45	.0075	0.04	0.007	-5.57 (0.007 x)	-8.86 (0.004 x)
90	1.40	.0085	0.02	0.005	-5.56 (0.003 x)	-8.90 (0.002 x)
100	1.34	.0891	0.01	0.004	-5.55 (0.002 x)	-8.93 (0.001 x)
3-replica simple						
	Replicas		Storage Used (PB)			
Year	Count	Std. Dev.	Mean	Std. Dev.	2-Replica Reactive Difference	3-Replica Reactive Difference
10	3.00	0.00	5.55	0.060	-0.36 (0.9 x)	-3.36 (0.6 x)
20	3.00	0.00	2.87	0.060	-2.99 (0.5 x)	-6.03 (0.3 x)
30	3.00	0.00	1.49	0.047	-4.32 (0.3 x)	-7.41 (0.2 x)
40	2.71	0.01	0.77	0.034	-5.00 (0.1 x)	-8.12 (0.09 x)
50	2.45	0.01	0.40	0.024	-5.33 (0.07 x)	-8.49 (0.04 x)
60	2.34	0.02	0.21	0.019	-5.48 (0.04 x)	-8.69 (0.02 x)
70	2.14	0.02	0.11	0.014	-5.54 (0.02 x)	-8.79 (0.01 x)
80	1.97	0.02	0.05	0.010	-5.56 (0.01 x)	-8.85 (0.006 x)
90	1.86	0.02	0.03	0.007	-5.55 (0.005 x)	-8.89 (0.003 x)
100	1.71	0.22	0.02	0.005	-5.54 (0.003 x)	-8.92 (0.002 x)

Table 25. Storage utilization and replica count for the reactive policies with the SMART error count model.

1-replica reactive						
	Replicas		Storage Used (PB)			
Year	Count	Std. Dev.	Mean	Std. Dev.	2-replica reactive difference	3-replica reactive difference
10	1.00	0.00	1.84	0.020	-4.08 (0.3 x)	-7.08 (0.2 x)
20	1.00	0.00	0.95	0.018	-4.91 (0.2 x)	-7.95 (0.1 x)
30	1.00	0.00	0.49	0.015	-5.32 (0.08 x)	-8.40 (0.06 x)
40	1.00	0.00	0.26	0.012	-5.51 (0.04 x)	-8.64 (0.03 x)
50	1.00	0.00	0.13	0.009	-5.59 (0.02 x)	-8.76 (0.01 x)
60	1.00	0.00	0.07	0.006	-5.62 (0.01 x)	-8.82 (0.008 x)
70	1.00	0.00	0.04	0.005	-5.61 (0.006 x)	-8.86 (0.004 x)
80	1.00	0.00	0.02	0.003	-5.59 (0.003 x)	-8.88 (0.002 x)
90	1.00	0.00	0.01	0.002	-5.57 (0.002 x)	-8.91 (0.001 x)
100	1.00	0.00	0.01	0.002	-5.55 (0.001 x)	-8.94 (0.0006 x)
2-replica reactive						
	Replicas		Storage Used (PB)			
Year	Count	Std. Dev.	Mean	Std. Dev.	2-replica reactive difference	3-replica reactive difference
10	1.99	0.0025	5.91	0.004	(1 x)	-3.00 (0.7 x)
20	1.99	0.0024	5.86	0.006	(1 x)	-3.04 (0.7 x)
30	1.99	0.0023	5.81	0.007	(1 x)	-3.08 (0.7 x)
40	1.99	0.0025	5.77	0.008	(1 x)	-3.13 (0.6 x)
50	1.99	0.0024	5.73	0.008	(1 x)	-3.17 (0.6 x)
60	1.99	0.0025	5.69	0.008	(1 x)	-3.21 (0.6 x)
70	1.99	0.0025	5.65	0.008	(1 x)	-3.25 (0.6 x)
80	1.99	0.0025	5.61	0.008	(1 x)	-3.29 (0.6 x)
90	1.99	0.0025	5.58	0.008	(1 x)	-3.34 (0.6 x)
100	1.99	0.0025	5.56	0.008	(1 x)	-3.38 (0.6 x)
3-replica reactive						
	Replicas		Storage Used (PB)			
Year	Count	Std. Dev.	Mean	Std. Dev.	2-replica reactive difference	3-replica reactive difference
10	2.98	0.0033	8.91	0.008	3.00 (1.5 x)	(1 x)
20	2.98	0.0032	8.90	0.009	3.04 (1.5 x)	(1 x)
30	2.98	0.0034	8.90	0.009	3.08 (1.5 x)	(1 x)
40	2.98	0.0033	8.89	0.009	3.13 (1.5 x)	(1 x)
50	2.98	0.0030	8.89	0.010	3.17 (1.6 x)	(1 x)
60	2.98	0.0034	8.89	0.010	3.21 (1.6 x)	(1 x)
70	2.98	0.0034	8.89	0.009	3.25 (1.6 x)	(1 x)
80	2.98	0.0032	8.90	0.010	3.29 (1.6 x)	(1 x)
90	2.98	0.0034	8.92	0.010	3.34 (1.6 x)	(1 x)
100	2.98	0.0034	8.94	0.010	3.38 (1.6 x)	(1 x)

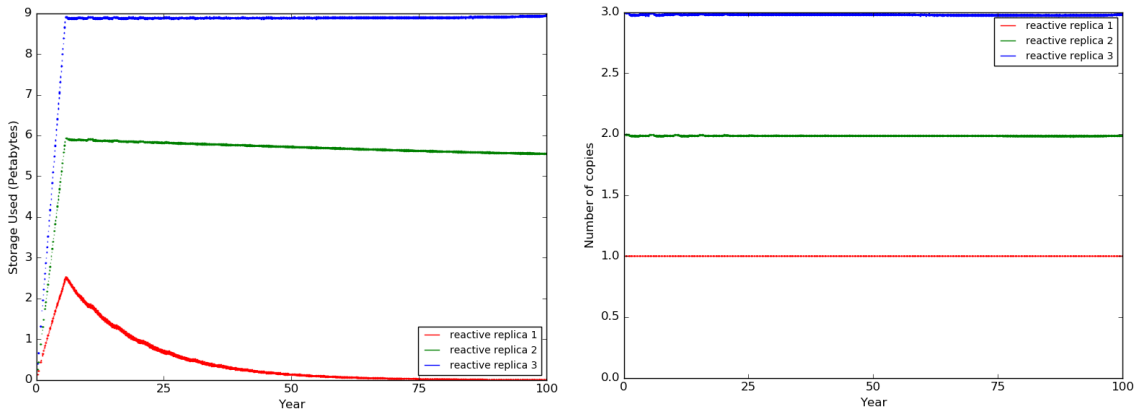


Figure 56. Storage and copies used over time for reactive policies with the SMART Error Count Model.

objects with less than expected replication. The storage utilization and replica counts are what would be anticipated for reactive policies with their maximum storage utilization being less than each policy's replication level multiplied by the combined size of data objects and matching requested replication levels. In order for the reactive policies to lose a data object, they must lose all replicas in quick succession before additional replication can occur.

SMART Aware Proactive

The SMART Error aware policies storage characteristics are displayed in Figure 57 and Table 26 below. The first graph has the storage utilization time series for each policy, and the second graph provides each policy's replication level time series.

The 1-replica proactive has a similar storage usage trend as the 1-replica simple and reactive policies; however, the proactive achieves a higher mean utilization at 2.8 petabytes owing to its slightly improved reliability. The post-populate exponential decay rate is 0.032, half the rate of the other 1-replica policies. The 2-replica, 1.0 and 1.5-health policies are nearly identical with storage utilization curves predominantly flat-lining after the populate phase just shy of the six petabyte mark by .04 to 0.01 petabytes over their lifetime, using 1 to 1.2 times the storage of the reactive policy. To be more precise the 1.0-health policy rises to 5.96 petabytes until 5.83 years and starts a decline of -40 gigabytes per year until around year 50. At that point, the storage starts climbing for the rest of the period at 600 gigabytes annually. The 1.5-health policy has no period of storage decline with an average annual rise of 300 gigabytes. The final policy, 2-replica 2.0-health, consumes more storage than its minimum replica count of 2 would designate using between 0.41 to 0.68 petabytes more than necessary for double replication. Like the 1.5-health policy, its storage trends are positive for the duration of the simulation with the post-populate phase being

Table 26. Replicas and storage used for SMART error proactive policy on the SMART error count model

1-replica 1.0-health SMART Error-aware proactive						
	Replicas		Storage Used (PB)			
Year	Count	Std. Dev.	Mean	Std. Dev.	2-replica reactive difference	3-replica reactive difference
10	1.08	0.004	2.49	0.021	-3.42 (0.4 x)	-6.42 (0.3 x)
20	1.13	0.006	1.86	0.024	-4.00 (0.3 x)	-7.04 (0.2 x)
30	1.15	0.008	1.37	0.024	-4.44 (0.2 x)	-7.53 (0.2 x)
40	1.15	0.010	1.00	0.021	-4.77 (0.2 x)	-7.90 (0.1 x)
50	1.15	0.012	0.73	0.018	-5.00 (0.1 x)	-8.17 (0.08 x)
60	1.16	0.013	0.53	0.017	-5.16 (0.09 x)	-8.36 (0.06 x)
70	1.16	0.015	0.39	0.015	-5.26 (0.07 x)	-8.51 (0.04 x)
80	1.16	0.019	0.28	0.013	-5.33 (0.05 x)	-8.62 (0.03 x)
90	1.16	0.022	0.20	0.011	-5.38 (0.04 x)	-8.71 (0.02 x)
100	1.16	0.024	0.15	0.010	-5.41 (0.03 x)	-8.79 (0.02 x)

2-replica 1.0-health SMART Error-aware proactive						
	Replicas		Storage Used (PB)			
Year	Count	Std. Dev.	Mean	Std. Dev.	2-replica reactive difference	3-replica reactive difference
10	2.00	0.00014	5.96	0.0006	0.05 (1 x)	-2.95 (0.7 x)
20	2.00	0.00023	5.96	0.0006	0.10 (1 x)	-2.94 (0.7 x)
30	2.00	0.00025	5.96	0.0010	0.15 (1 x)	-2.94 (0.7 x)
40	2.00	0.00032	5.96	0.0010	0.19 (1 x)	-2.94 (0.7 x)
50	2.00	0.00024	5.96	0.0014	0.23 (1 x)	-2.93 (0.7 x)
60	2.00	0.00033	5.96	0.0014	0.27 (1 x)	-2.93 (0.7 x)
70	2.00	0.00030	5.96	0.0018	0.31 (1.1 x)	-2.93 (0.7 x)
80	2.00	0.00034	5.96	0.0018	0.35 (1.1 x)	-2.94 (0.7 x)
90	2.00	0.00029	5.97	0.0021	0.39 (1.1 x)	-2.94 (0.7 x)
100	2.00	0.00030	5.99	0.0021	0.43 (1.1 x)	-2.95 (0.7 x)

2-replica 1.5-health SMART Error-aware proactive						
	Replicas		Storage Used (PB)			
Year	Count	Std. Dev.	Mean	Std. Dev.	2-replica reactive difference	3-replica reactive difference
10	2.00	0.00019	5.95	0.0007	0.04 (1 x)	-2.96 (0.7 x)
20	2.00	0.00022	5.96	0.0006	0.10 (1 x)	-2.95 (0.7 x)
30	2.00	0.00019	5.96	0.0008	0.14 (1 x)	-2.94 (0.7 x)
40	2.00	0.00025	5.96	0.0005	0.19 (1 x)	-2.94 (0.7 x)
50	2.00	0.00018	5.96	0.0007	0.23 (1 x)	-2.94 (0.7 x)
60	2.00	0.00026	5.96	0.0006	0.27 (1 x)	-2.93 (0.7 x)
70	2.00	0.00023	5.96	0.0008	0.31 (1.1 x)	-2.93 (0.7 x)
80	2.00	0.00029	5.96	0.0007	0.35 (1.1 x)	-2.94 (0.7 x)
90	2.00	0.00022	5.97	0.0010	0.39 (1.1 x)	-2.94 (0.7 x)
100	2.00	0.00023	5.99	0.0007	0.43 (1.1 x)	-2.95 (0.7 x)

Table 26. Continued.

2-replica 2.0-health SMART Error-aware proactive									
	Replicas		Storage Used (PB)						
Year	Count	Std. Dev.	Mean	Std. Dev.	2-replica reactive difference			3-replica reactive difference	
10	2.15	0.0067	6.41	0.021	0.50	(1.1	x)	-2.50	(0.7 x)
20	2.21	0.0078	6.59	0.023	0.73	(1.1	x)	-2.31	(0.7 x)
30	2.22	0.0074	6.63	0.023	0.82	(1.1	x)	-2.27	(0.7 x)
40	2.23	0.0074	6.64	0.023	0.87	(1.2	x)	-2.26	(0.7 x)
50	2.23	0.0076	6.64	0.023	0.91	(1.2	x)	-2.25	(0.7 x)
60	2.23	0.0074	6.64	0.022	0.96	(1.2	x)	-2.25	(0.7 x)
70	2.23	0.0074	6.64	0.022	1.00	(1.2	x)	-2.25	(0.7 x)
80	2.22	0.0078	6.65	0.023	1.04	(1.2	x)	-2.25	(0.7 x)
90	2.22	0.0077	6.66	0.023	1.08	(1.2	x)	-2.25	(0.7 x)
100	2.23	0.0077	6.68	0.023	1.12	(1.2	x)	-2.26	(0.7 x)

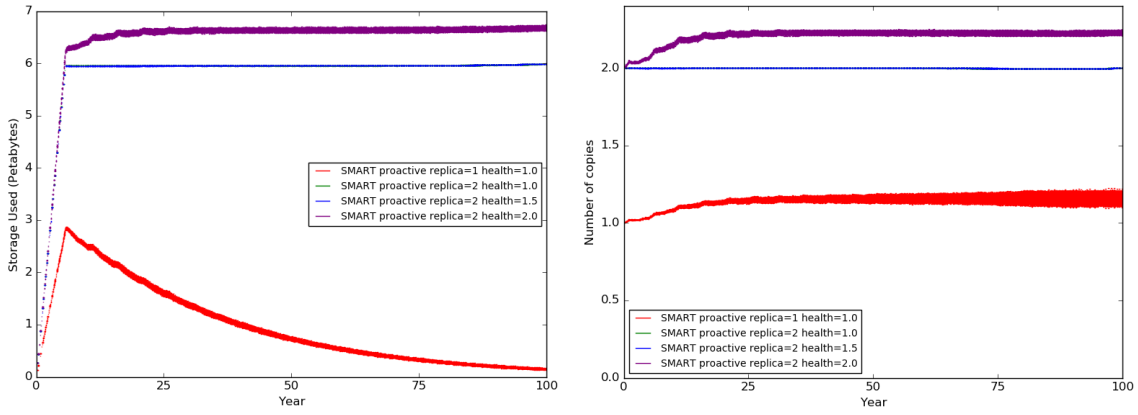


Figure 57. Storage and copies used over time for SMART error aware proactive policy on the SMART error count model.

dividable into two periods. The first period, lasting until year 20, has an annual increase of 20 terabytes before steadying out to 600 gigabytes for the second period.

The second graph in Figure 57, provides illumination on how each policy applies its storage expenditures. The 2-replica, 1.0 and 1.5-health policies maintain a consistent two copies of all of the surviving data objects with both of them using more storage than the 2-replica reactive because of their slower data loss rate. The other two policies demonstrate greater fluctuations. The 1-replica, 1.0-health policy averages between 1.08 and 1.16 copies per data object, and the 2-replica, 2.0-health possesses 2.23 copies once the simulations settled out. While these two policies are more reliable, they do require more replication, and hence storage, than their reactive counterparts.

Normalized Magnitude of Data Loss

In this section, the reliability and storage usage results are analyzed together through the NOMDL metric to shed further light upon how the policies stack up against one another after the full mission time. The number of lost data objects and the maximum mean replica count per policy are given in Table 27.

For the simple and reactive policies, the replica count is the requested replication level of the policy. Since the replication level for the proactive policy replica count serves as the base, the maximum of the mean replica count for the 100-year time frame is used. The resulting storage used, the replica count multiplied by the combined size of all the data objects, is the worst case for the proactive policies. D is taken from the apparent storage taken up by the data objects, three petabytes, and the remaining free storage in the system. The magnitude of data loss, MDL, is the number of data objects lost multiplied by their size.

Table 27. NOMDL for SMART error count policies.

Policy	Lost Data Objects	Replica Count	Storage Used (PB)	Storage Free (PB)	MDL (PB)	D (PB)	NOMDL
1-replica Simple	2,994,961	1.00	3.00	6.22	2.99	9.22	0.32
2-replica Simple	2,990,129	2.00	6.00	3.22	2.99	6.22	0.48
3-replica Simple	2,985,283	3.00	9.00	0.22	2.99	3.22	0.92
1-replica Reactive	2,995,080	1.00	3.00	6.22	3.00	9.22	0.32
2-replica Reactive	205,356	2.00	6.00	3.22	0.21	6.22	0.033
3-replica Reactive	1,860	3.00	9.00	0.22	0.0019	3.22	0.00058
1-replica 1.0-health	2,870,772	1.16	3.48	5.73	2.87	8.73	0.33
2-replica 1.0-health	3,485	2.00	6.00	3.22	0.0035	6.22	0.00056
2-replica 1.5-health	1,523	2.00	6.00	3.22	0.0015	6.22	0.00024
2-replica 2.0-health	1,019	2.23	6.69	2.53	0.0010	5.53	0.00018

The first four policies all perform very poorly approaching a value of 1. For the three simple policies, even though slightly fewer data objects are lost as the replication level rises, the NOMDL increases 1.5 times for each increase in replication due to the storage usage. In the case of the reactive policies, however, despite using an extra three petabytes for every replica required, the reactive policies drastically perform better on the NOMDL with the final reactive policy scoring 552 times better than the single replica.

While the 1-replica, 1.0-health policy loses over 100,000 fewer data objects than the 1-replica simple policy, its NOMDL score is slightly poorer having been projected to use an additional 0.48 petabytes of storage for all of the data objects. The 2-replica, 1.0-health proactive policy lost 1.9 times as much data as the 3-replica reactive, but after saving three petabytes of storage did marginally better according to its NOMDL value. While the 2-replica, 1.0 and 1.5-health policies both consumed nearly equal amounts of storage, the 1.5-health NOMDL scored over two times lower owing to its better reliability. The 2-replica, 2.0-health in spite of using more storage than any of the other proactive policies achieved the best NOMDL ranking of all.

Bandwidth Used

Simple

Figure 58 and Table 28 below show the distribution of data flows types by their locality and policy for the set of simple policy configurations.

Each of the policies expends the same three petabytes worth of bandwidth in the local area for the initial upload of the data objects to the clients' local sites. There is some variance in the 3-replica policy most likely due to lack of storage space as the sites ran low on space towards to the end of the populate phase. A small fraction of the later arriving data objects had to travel across the wide area to other nearby sites for their initial upload. The three petabyte steps in the wide area traffic seen across the policies were the result of the additional requested replicas. The wide area traffic was essentially equal to the number of replicas - 1 multiplied by the total size of the data objects. There was some bus only traffic in the 3-replica policy due to the aforementioned phenomena. With storage space nearly exhausted, the 3-replica policy had to resort to occasionally having multiple replicas at the same site to meet demands and with depot-to-depot replication being the most efficient placed the multiple copies at the same depot. Based on the values in the table below this event occurred for about 4 data objects per simulation.

The behavior of the simple policies can serve as a baseline when analyzing the reactive and proactive policies in the next two sections.

Reactive

The results of the data flow analysis for the reactive policies are given in Figure 59 and Table 29. The reactive policies produce the same amount of local area

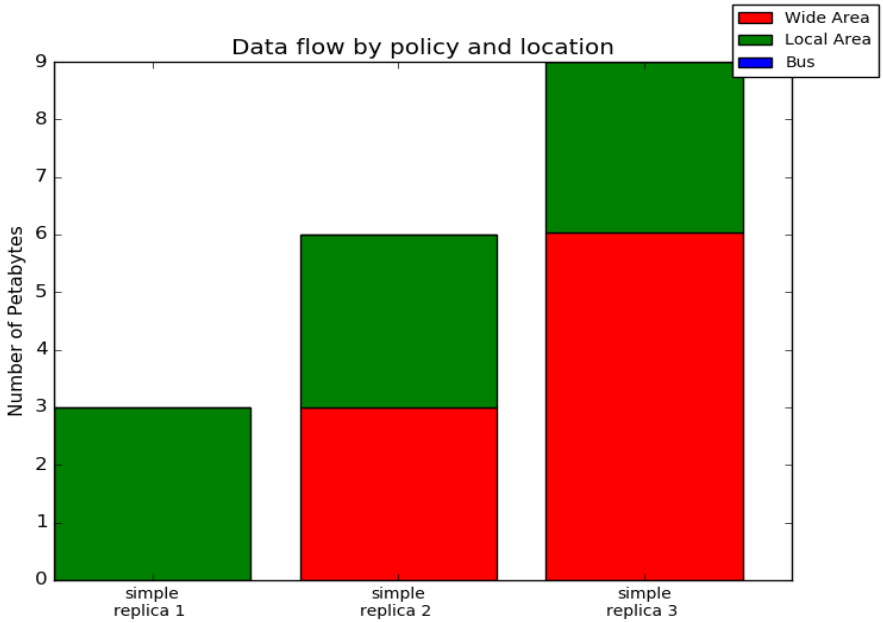


Figure 58. Data flow for the simple policy and SMART error count Model.

Table 28. Simple policies data flow for SMART error count model.

Simple Policy Data Flows				
1-Replica Simple				
Traffic	Mean (PB)	Std. Dev (PB)	2-Replica Reactive Difference (PB)	3-Replica Reactive Difference (PB)
Bus			0.00 (x)	(x)
Local	3.00	0.00	0.00 (1x)	0.00014 (1.00x)
Wide			-39.42 (0x)	-62.96 (0x)
Total	3.00		-39.42 (0.07x)	-62.96 (0.05x)
2-Replica Simple				
Traffic	Mean (PB)	Std. Dev (PB)	2-Replica Reactive Difference (PB)	3-Replica Reactive Difference (PB)
Bus			0.00 (x)	0.00 (x)
Local	3.00	0.00003	-0.0000015 (1x)	0.00014 (1x)
Wide	3.00	0.00003	-36.42 (0.08x)	-59.96 (0.05x)
Total	6.00		-36.42 (0.1x)	-59.96 (0.09x)
3-Replica Simple				
Traffic	Mean (PB)	Std. Dev (PB)	2-Replica Reactive Difference (PB)	3-Replica Reactive Difference (PB)
Bus	0.000004	0.000003	0.000004 (x)	0.000004(x)
Local	2.97	0.003	-0.03 (1x)	-0.03(1x)
Wide	6.03	0.003	-33.39 (0.2x)	-56.93(0.1x)
Total	9.00		-33.42 (0.2x)	-56.96(0.1x)

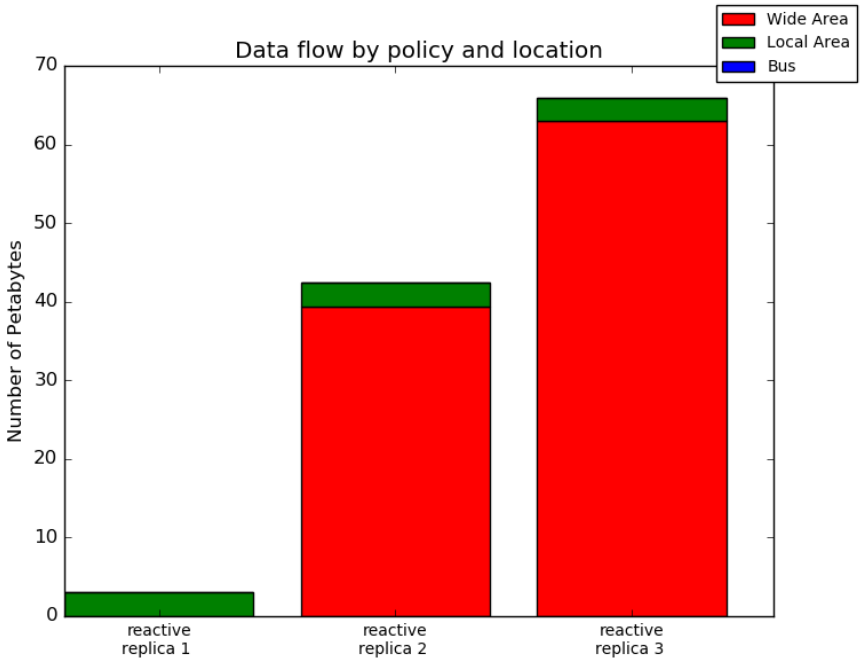


Figure 59. Reactive policy data flows for SMART error count model.

Table 29. Reactive policy data flows for the SMART error count model.

Reactive Policy Data Flows				
1-Replica Reactive				
Traffic	Mean	Std. Dev	2-Replica Reactive Difference	3-Replica Reactive Difference
Bus			(x)	(x)
Local	3.00		0.00 (1.0x)	0.00014(1.0x)
Wide			-39.42 (0x)	-62.96(0.0x)
Total	3.00		-39.42 (0.07x)	-62.96(0.05x)
2-Replica Reactive				
Traffic	Mean	Std. Dev	2-Replica Reactive Difference	3-Replica Reactive Difference
Bus	0.00	0.00	(x)	0.00(x)
Local	3.00	0.00	(1x)	0.00014(1.0x)
Wide	39.42	0.12	(1x)	-23.54(0.6x)
Total	42.42		(1x)	-23.54(0.6x)
3-Replica Reactive				
Traffic	Mean	Std. Dev	2-Replica Reactive Difference	3-Replica Reactive Difference
Bus	0.00	0.0000	0.0000 (x)	(x)
Local	3.00	0.0004	-0.0001 (1x)	(1x)
Wide	62.96	0.09	23.54 (1.6x)	(1x)
Total	65.96		23.54 (1.6x)	(1x)

only traffic as the simple policies for the initial data object injection and almost nil bus-only traffic. The 1-replica policy also has the same bandwidth utilization and distribution as the 1-replica simple policy. The 2 and 3-replica policies generate substantially more wide area traffic than their simple policy counterparts due to their repair mechanism. With no ability to predict near term disk failure, all the replication repair work requires replicas to traverse the wide area. Subtracting the amount of traffic needed for the initial site distribution from their wide area usage of 39.42 and 62.96 petabytes, means that 36.42 and 56.96 petabytes are used between sites for replacement work. 86% of the total replication work the policies performed was solely for repair, equating to 36 to 57 million extra replication events.

SMART Aware Proactive

Figure 60 and Table 30 provide an overview of the results of the data flow generated by the SMART Error-Aware proactive policies. All of the proactive policies have the same quantity of local area traffic, three petabytes, for the same reason as the reactive and simple policies. The interesting results lie in the variability of the bus-only and wide area traffic patterns.

The 1-replica, 1-health policy has the lowest traffic with only 6.57 petabytes ever moved in the simulations. Like the 1-replica reactive policy, it spends three petabytes in the local area handling data object arrival from on-site data clients. Unlike the 1-replica reactive, though, it then uses 3.57 petabytes in

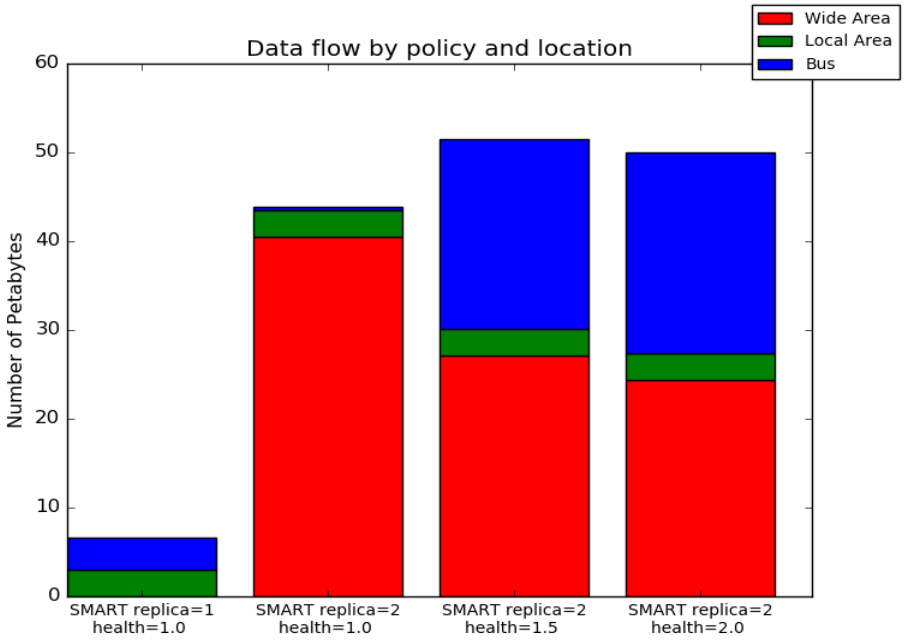


Figure 60. Proactive policy data flows run against the SMART error count model.

Table 30. Proactive policy data flows on the SMART error failure model.

Proactive Data Flows				
1-replica 1.0-health				
Traffic	Mean	Std. Dev	2-Replica Reactive Difference	3-Replica Reactive Difference
Bus	3.57	0.06	3.57 (x)	3.57(x)
Local	3.00	0.00	0.00 (1.0x)	0.00014(1.0x)
Wide	0.0008	0.0007	-39.42 (0.00002x)	-62.96(0.00001x)
Total	6.57		-35.84 (0.2x)	-59.39(0.1x)
2-replica 1.0-health				
Traffic	Mean	Std. Dev	2-Replica Reactive Difference	3-Replica Reactive Difference
Bus	0.4	0.004	0.42 (x)	0.42(x)
Local	3.0	0.0	0.0 (1.0x)	0.00014(1.0x)
Wide	40.5	0.06	1.10 (1.0x)	-22.44(0.6x)
Total	43.9		1.52 (1.0x)	-22.03(0.7x)
2-replica 1.5-health				
Traffic	Mean	Std. Dev	2-Replica Reactive Difference	3-Replica Reactive Difference
Bus	21.33	0.16	21.33 (x)	21.33(x)
Local	3.00	0.00	0.00 (1.0x)	0.00014(1.0x)
Wide	27.13	0.09	-12.29 (0.7x)	-35.84(0.4x)
Total	51.45		9.03 (1.2x)	-14.51(0.8x)
2-replica 2.0-health				
Traffic	Mean	Std. Dev	2-Replica Reactive Difference	3-Replica Reactive Difference
Bus	22.63	0.57	22.63 (x)	22.63(x)
Local	3.00	0.00	0.00 (1x)	0.00(1.0x)
Wide	24.33	0.08	-15.08 (0.6x)	-38.63(0.4x)
Total	49.97		7.55 (1.2x)	-15.99(0.8x)

bus traffic shuffling data objects between disks on the same depot to keep them on SMART Error free disks. More traffic would have been generated by this policy had it not been for its poor reliability performance.

The subsequent three proactive policies, all 2-replica, create substantial more traffic. In fact, each of these policies generate more overall traffic than the 2-replica reactive ranging from 1.5 to 9 PB more; bearing in mind that while these policies do use up to 1.2 times the bandwidth, they also have far better reliability characteristics having lost 50 to 200 times less data. Had the 2-replica reactive policy managed to sustain more data objects, then its traffic would have also been higher. The 2-replica, 1.5 and 2.0-health policies produced less than 80% of the total traffic as the 3-replica reactive.

Based on bus-only traffic numbers, the proactive policy, with the exception of the 2-replica 1.0-health configuration, is taking advantage of intra-depot

replication to maintain the health of the data objects. As the health requirement rises so does the bus traffic. The proactive policy is successfully leveraging disk-to-disk replication for data migration for suspected near-term disk failure to reduce the need for replacement via wide area replication. As the policy configurations can exploit pre-emptive data migration, there is a corresponding drastic reduction in wide area traffic when compared to the reactive policies. The two latter proactive policy configurations use 12 to 15 and 35 to 38 fewer petabytes than the 2 and 3-replica reactive policies respectively. This reduces the traffic 31-38% for the 2-replica reactive and 57-61% of the 3-replica reactive.

The 2-replica, 1.0-health was not able to take as much of an advantage with its health to replica count ratio being so skewed. In order to trigger a disk migration, both disks would have to suffer SMART Errors within an 8-month timeframe for the health of the data object to drop low enough. A single disk could suffer as many SMART errors as possible as along as the second disk remained error-free. Therefore, it was much more probable for a SMART error associated disk to fail before the second disk experiences any SMART errors, triggering wide area replication replacement.

CFDR

This section presents the performance results of the policy configurations against the age based failure model devised from the CFDR datasets. The following four subsections examine the data survival, storage utilization, replica counts, data flow patterns and NOMDL for each of the policies. Each subsection discusses the 1 to 3 replica simple and reactive policies and then the specialized CFDR proactive policy with following configuration settings for replica and reliability: 1, 0.9; 1, 0.99; 1, 0.999; 1, 0.9999; 1, 0.99999; 2, 0.999; 2, 0.9999; 2, 0.99999 and 3, 0.99999.

Data Survival

Simple

The data survival analysis begins with the simple policy, a summary of which are posted below in Figure 61. From the outset of the runs, there is a very a tight relationship between the replication level and the overall reliability of the simple policy. By the end of the 100-year simulations, the majority of the data objects have been lost with only 3.2%, 6.3% and 9.3% surviving. None of the policies' runs ended with complete data loss. The maximum lost count had 1-replica with 2,923,448; 2-replica with 2,846,944 and 3-replica with 2,777,372. As the replication level rises, the quantity of data loss is pushed further out in time. The greatest difference between the policies in regards to 1-replica occurs at year 20 with 750,571 for the 2-replica and year 25 with 1,154,985 for the 3-replica. At 31 years, the 2 and 3-replica policies have diverged to their maximum extent with 441,992 objects. The difference gradual diminishes from those points up to year 100.

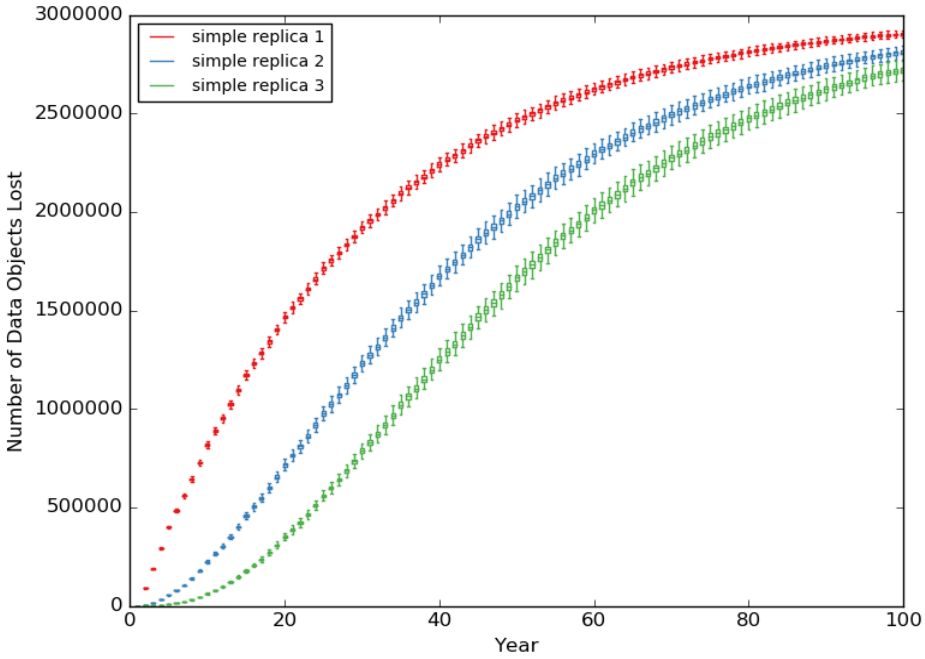


Figure 61. Simple policy reliability against the CFDR age failure model.

Performing regression analysis, the data objects' survivorship was found to decline exponentially with the following decay rates of 0.035, 0.03 and 0.026 for the policies respectively. The R^2 values indicate that the goodness of fit lessens; 0.99997, 0.99177 and 0.97552; as the replication level rises and the curves move from a pure exponential to demonstrating a logistic sigmoid curve.

Reactive

Figure 62 provides the cumulative data loss for the three reactive policies. The 1-replica reactive clearly performs far worse than the other reactive configurations, matching the data loss curve of the 1-replica simple. The survivorship exponentially decays at a rate of 0.035 per year with a R^2 value of 0.99997. By the end of 100-year run time, the policy has lost all but 3.2% of the data objects. The 2 and 3-replica policies perform significantly better ending with 97.672% and 99.987% reliabilities and decline linearly instead of exponentially. The linear survivorship slope of the 2-replica policy is -686.13 and -3.4802 for the 3-relica policy with R^2 values of 0.99199 and 0.97258. Even though the 2 and 3-replica reactive policies can only recover after a disk failure, they still manage to outperform the simple and 1-replica reactive policies by 2 to 5 orders of magnitude. The 3-replica reactive only loses on average 386 data objects after 100 years of operation.

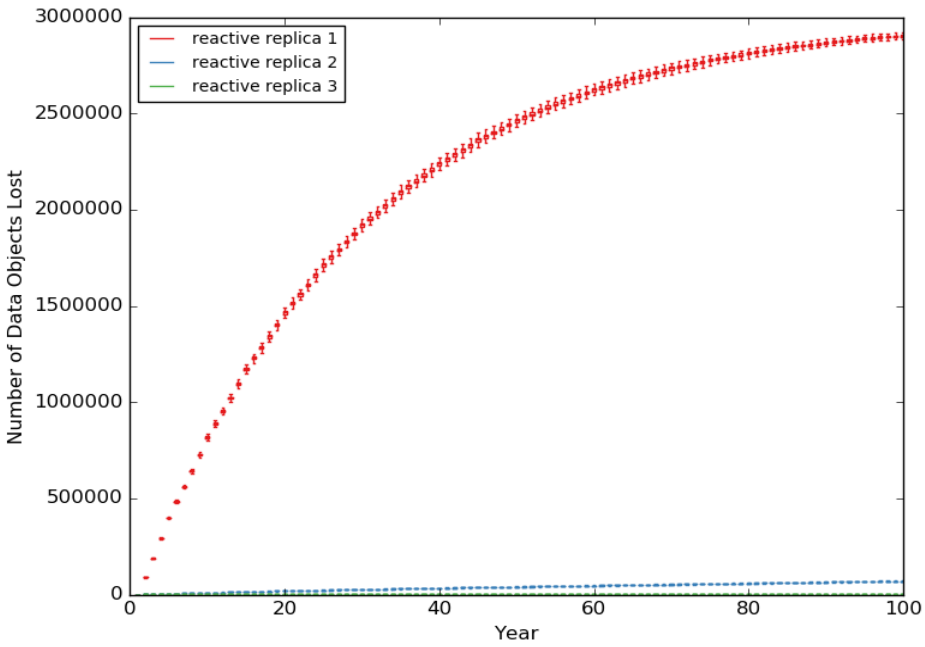


Figure 62. Reactive policies' reliability against the CFDR age model.

Disk Age Aware Proactive

Having covered how the traditional techniques would fare against the CFDR failure model, in this subsection, the reliability of the nine different configurations of the CFDR proactive policy are examined. In order to improve readability in the figures, a consistent color code for each of these configurations was chosen and when used reactive policies will appear in black.

Starting with Figure 63 the cumulative data loss for all of the proactive policies is given. The first three policies; 1-replica, 0.9, 0.99 and 0.999-reliability; perform the worst. In fact, the 1-replica, 0.9 and 0.99-reliability demonstrate the same exponential data loss pattern seen in the simple and 1-replica reactive policies. Regression analysis reveals that the 1-replica, 0.9-reliability policy has the same exponential decay rate of 0.035 as 1-replica reactive. The second proactive policy does better with a decay rate of 0.027 per year. All of the remaining seven proactive policies behave linearly with their data loss. For instance, the next policy, 1-replica, 0.999-reliability, has a slope of 4744.5 data objects lost per year.

The next figure, Figure 64, provides a visual comparison of the first three policies bounded above and below by the 1 and 2-replica reactive policies. The curves for the 1-replica, 0.9-reliability proactive policy and the 1-replica reactive are nearly identical. Before year 9, the mean data loss of the 1-replica, 0.9-reliability proactive policy preserves more data objects, but after that point, the two policies fluctuate back and forth by a few thousand data objects per year. As

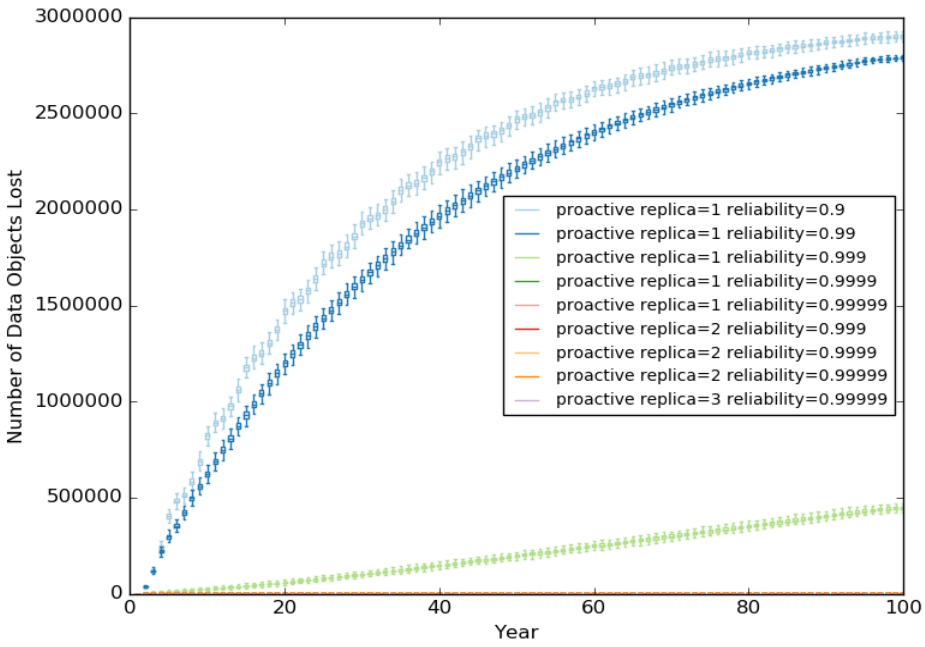


Figure 63. All of the CFDR proactive policy reliabilities.

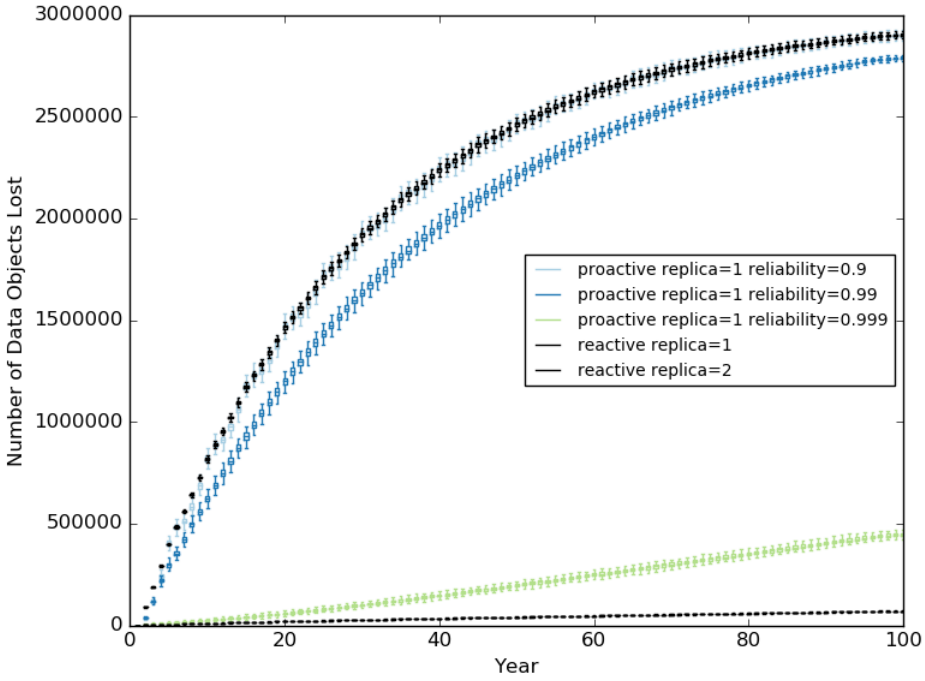


Figure 64. Reliability of the first three CFDR proactive policies vs. the reactive policies.

they produce the same regression line with an R^2 greater than 0.999, it is a statistical tie between them. The second proactive policy, 1-replica, 0.99-reliability, clearly does better than the 1-replica reactive losing 126,756 fewer data objects or about 4% less. However, it still suffers 40 to 7,194 times the data loss as the more realistically useful 2 and 3-replica reactive policies. The situation begins to change with the 1-replica proactive policies with reliabilities set to 0.999 or greater. The data loss for 1-replica, 0.999-replica policy drastically diminishes to only 6 to 1,150 times of the 2 and 3-replica reactive policies. Additionally, the loss curves transition to linear curves with the survivorship of the 1-replica, 0.999-reliability policy declining at a rate of 4,744.5 data objects a year. This is the last proactive policy configuration that performs worse than the 2-replica reactive.

In Figure 65, the final two 1-replica policies' reliabilities are shown against the reliability of the 3-replica reactive policy in black. The long term reliabilities of 1-replica, 0.9999 and 0.99999-reliability policies far exceed that of the 0.999-reliability policy. The two policies outperformed the previous policy by 371 and 1,118 times and by extension the 2-replica reactive policy by 58 and 156 times. The entire 100-year reliabilities of the policies were 99.960% and 99.987% putting them on par with the 3-replica reactive policy. The yearly data loss was linear for the policies with slopes of 9.73 and 1.53 data objects per year.

With 1,198 data objects lost, the 0.9999-reliability policy finished having lost 3.1 times as many data objects as the 3-replica reactive; however, as Figure

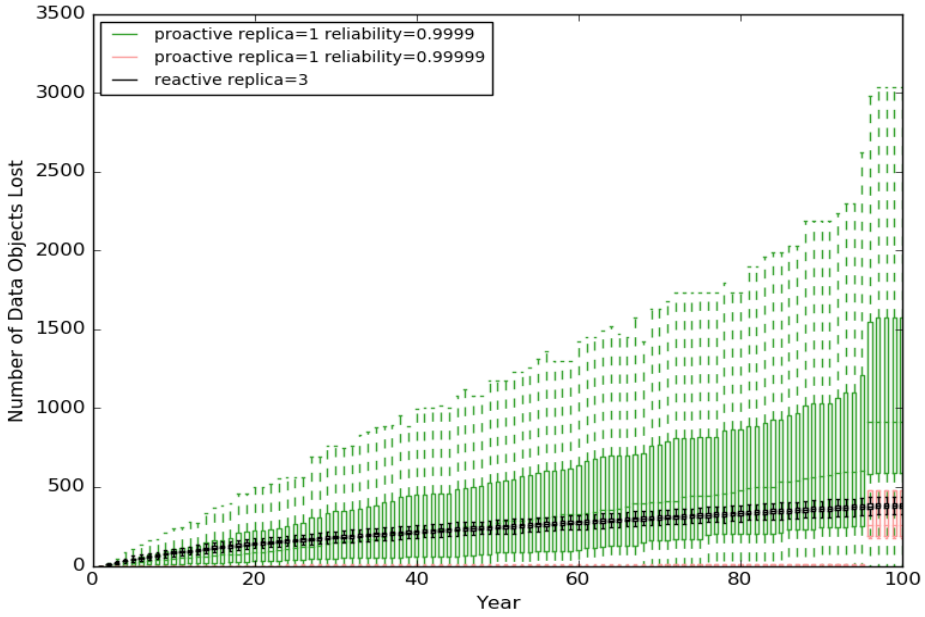


Figure 65. 1-replica, 0.9999 and 0.99999-reliability CFDR proactive reliability vs. 3-replica reactive policy.

65 shows, this final measurement is deceptive. For most of the lifetime of the simulations, this policy has a loss ratio with the 3-replica policy ranging from 0.7 to 2.3. Then in the last five years, there is a significant degradation in performance that causes the ratio to increase from 2.3 to 3.1.

The 0.99999-reliability policy suffers an average loss of 397 data objects which equates to 1.16 times worse than the 3-replica reactive. Like the 0.9999-reliability policy, though, this policy has a sudden degradation in the last five years that distorts its performance for the majority of the simulation runs. Before year 95, this configuration of the proactive policy outperforms the 3-replica reactive substantially. It loses only 126 data objects, around a third of the amount lost by the reactive policy. The cause of the jump in the last five years is unclear but could be due to sample size or an internal error with LoDN's reliability calculator.

The reliability over time of the multiple replica proactive policies with all configured reliabilities is given in Figure 66 along with the reliability of the 3-replica reactive. All of the 2 and 3-replica proactive policies soundly surpass the performance of the 2-replica reactive, by factors ranging from 63 to 250,000, and are competitive with the 3-replica reactive policy. For the four policies, the 100-year mean data loss and reliability are as follows: 1,109 data objects lost, 99.963% reliability for the 2-replica, 0.999-reliability; 815 data objects lost, 99.973% reliability for the 2-replica, 0.9999-reliaiblity; 594 data objects lost, 99.980% reliability for the 2-replica, 0.99999-reliability and 0.289855 data objects lost, 99.99999% for the 3-replica, 0.99999-relability policy.

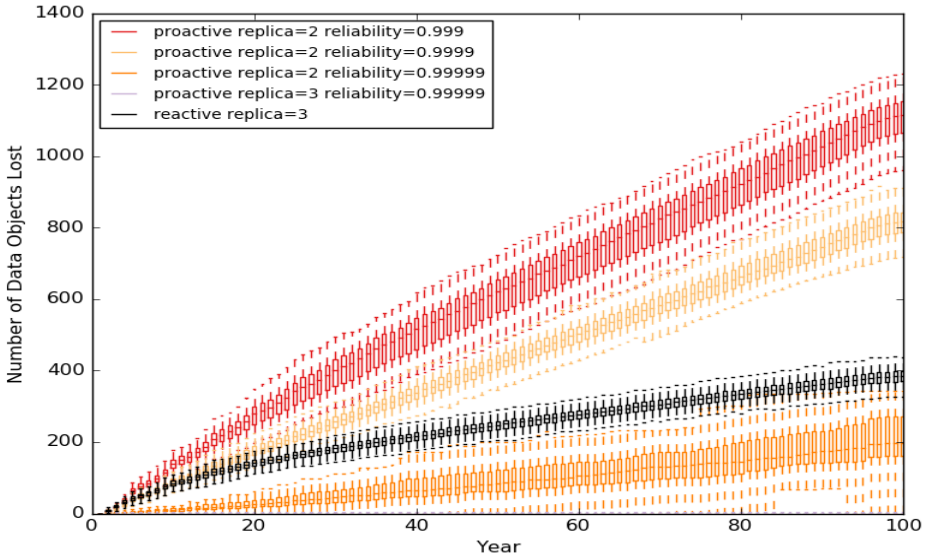


Figure 66. Reliability of 2 and 3-replica proactive CFDR policies compared to 3-replica reactive policy.

The 2-replica, 0.99999-reliability policy actually does better than the mean metric indicates as shown by its time series box plots. Due to seven poorly performing outliers, the mean is severely skewed from the median value of only 200 data objects lost. These outliers result in a standard deviation of 1,665 data objects. While this may not seem like much, using the median value instead gives the policy configuration a reliability of 99.993% and beats the 3-replica reactive policy consistently. The Tukey box plots show that all of the other simulations for the configuration are relatively well behaved with a relatively small IQR range.

Using the mean data loss, all three of the 2-replica proactive policies do worse than 386 data objects lost by the 3-replica reactive policy by 2.9, 2.1 and 1.5 times in order of increasing reliability. However, if we exclude those few outliers for the 2-replica, 0.999999-reliability policy then as indicated by Figure 66, this proactive configuration is 1.9 times better than the reactive policy. Additionally, while the 3-replica reactive policy always lost data with a minimum occurrence of 319 data objects, there were runs for this proactive policy configuration that lost no data objects. The final proactive policy, the 3-replica, 0.99999-reliability policy, has the greatest reliability of all the policies and scores over 1,300 times more reliable than the 3-replica reactive policy.

Performing linear regression analysis on the results gives the following data loss rates per year for the policies: 2-replica, 0.999-reliability with 11.002 (R^2 = 0.996); 2-replica 0.9999-reliability with 8.243 (R^2 = 0.9996); 2-replica, 0.99999-reliability with 6.6099 (R^2 = 0.927) and 3-replica, 0.99999-reliability with 0.0012 (R^2 = 0.693)

Figure 67 provides a cross comparison of the proactive policy with constant reliabilities specifications for differing replica counts to examine how significant replica count alone is for data object reliability. For the 0.999-reliability policies, there is a substantial difference between the 1 and 2-replica configurations. The 2-replica loses only 11.002 data objects per year compared to the 4,744.5 data objects lost per by year by the 1-replica policy. By the end of the simulation, there is a 400-fold difference in lost data objects. The next reliability setting, 0.9999, exhibits significantly less difference between the 1 and 2-replica policies. The mean annual data loss for 1-replica is 9.7343 versus 8.2428 for the 2-replica with an ending difference of 115 data objects. Furthermore, as shown in the graph, the median data loss for the 1-replica is lower than that of the 2-replica for 95 out of the 100 years albeit with the 1-replica having greater variability.

By the nature of having two guaranteed replicas, a data object will almost always exceed the 0.999-reliability requirement, but not the 0.9999-reliability requirement. Therefore, in the first scenario, the data objects under the 2-replica policy have greater reliability than the minimum required. In the second scenario, both policies have to routinely perform data migrations to maintain the 0.99999-reliability requirement. This conclusion is further supported by results in the data flow section that show substantial bus-only traffic for both 1 and 2-replica policies

with the 0.9999-reliability configuration and only for the 1-replica in the 0.999-reliability configuration.

For the 0.99999-reliability configurations in the bottom graph of Figure 67, the 1-replica policy outperforms the 2-replica policy with a yearly data loss of 1.53 objects versus 6.61 data objects. At the end of the simulations, the 2-replica losses on average 594 data objects against the 447 data objects of the 1-replica. As previously discussed there is a data loss spike in some the simulations that drives up the median loss for the 1-replica, but it is unclear as to what causes this. By requiring only a single copy the 1-replica policy may do better than the 2-replica with more free storage play in the system, but further simulations and analysis will need to be done to confirm this assertion. The 3-replica policy surpasses both of the other policies losing only 0.0012 data objects a year. The 3-replica performs better for the same reason as the 2-replica did for the 0.999-reliability configuration. For data objects with three replicas having a minimum reliability of 0.99999 is a near certainty and typically will result in reliabilities in excess of the requirement.

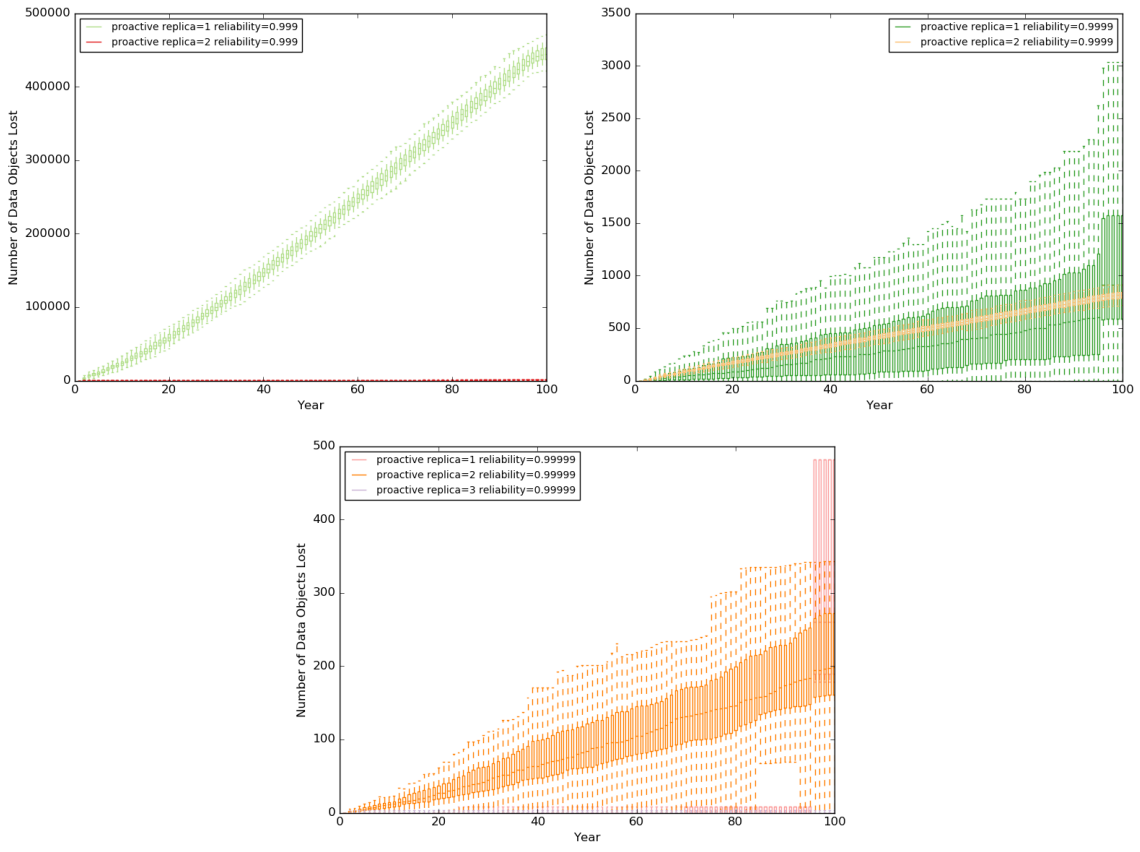


Figure 67. Reliability cross comparison between the CFDR proactive policies with the same reliability and different replica counts.

Storage Used & Copies Used

Simple

In order to understand how the reactive and proactive policy utilize storage, a baseline of the storage utilization for the three simple policy configurations is first covered in this subsection. Figure 68 provides three different curves detailing the storage utilization (left graph) and the number of copies remaining per surviving data object (right graph) for each simple policy.

Each policy produces the same basic storage utilization profile, varying by how much initial storage they consume to meet the replication demand. The profiles can be divided into two separate phases. In the first phase, LoDN is being populated by data objects and the data objects are being replicated across the sites. Despite data object loss already occurring, during this phase storage usage rises linearly with the slope of the curve dictated by the required replication count. The slopes of these curves are 0.46, 0.92 and 1.48 petabytes per year for the 1, 2 and 3-replica policies. The maximum utilization occurs near the end of the ingest phase at 5.67 years with each policy reaching 89.3% (2.68, 5.36, 8.04 petabytes) of their expected usage had no data objects been lost. After peaking, each policy's storage usage exponentially decays at the same rate of 0.035 per year with finishing storage usages of 98.45, 188.69, 291.75 terabytes for the 1, 2 and 3-replica configurations.

For the average number of copies used, each policy starts out with the replica count matching its respective settings. The 1-replica policy never falls beneath one since any data object with less than one is no longer a survivor beyond that time. With an inability to replenish lost replicas, the 2 and 3-replica policies begin an almost instant decline in the number of copies with logarithmic decay rates of -0.251 and -0.519, respectively. By the end of the 100-year

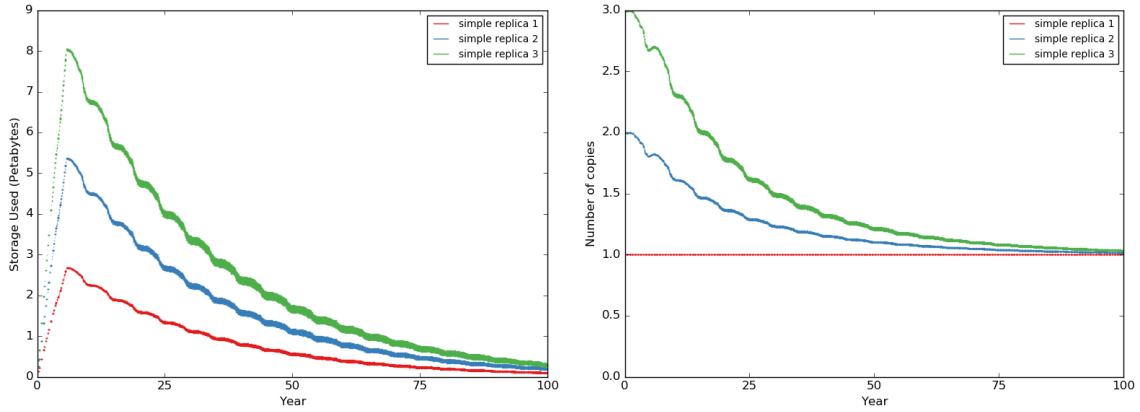


Figure 68. Storage and copies used over time for the simple policies and the CFDR policy. The left figure gives the storage (in petabytes) used over time by the three simple policies. The right figure gives the number of replicas per data object that survived to that time.

mission time, the two policies have lost so many replicas that even the surviving data objects are just barely above one replica averaging 1.02 and 1.03.

Reactive

Figure 69 provides graphs representing the storage utilization and the average number of replicas for the surviving data objects under the three reactive policy configurations. The 1-replica reactive policy mirrors the storage distribution of the 1-replica simple policy. During the populate phase, the storage consumed increases at 0.46 petabytes per year until 5.67 years when consumption peaks at 2.68 petabytes. Following the peak, storage usage decays exponentially at a rate of -0.035 per year before ending at 99.12 terabytes. As the 2 and 3-replica reactive have the ability to replace lost replicas for data objects their storage usage is more consistent. The 2-replica linearly rises at 1.05 petabytes per year during the populate phase reaching 5.95 petabytes before starting a slow, steady taper of -1.01 terabytes a year. It finishes with a usage of 5.84 petabytes. In the first part, the 3-replica's storage usage increases at 1.57 petabytes a year, but unlike the prior policies its usage does not peak at the end of the data populate phase. Instead, it continues to rise by 0.51 terabytes a year achieving a maximum usage of 8.97 petabytes.

Scrutiny of the graph on the right of the figure reveals that the average number of replicas per data object for the 2 and 3-replica policies fluctuates just shy of the replication level targets for the two policies. As replicas are being lost and having to be replenished nearly constantly, the replica count is imprecise but typically centers around 1.99 and 2.98. The small range for each plotted time demonstrates the response time and speed by which the policies are able to replace replicas. For data loss to occur, all of the replicas must be lost rapidly.

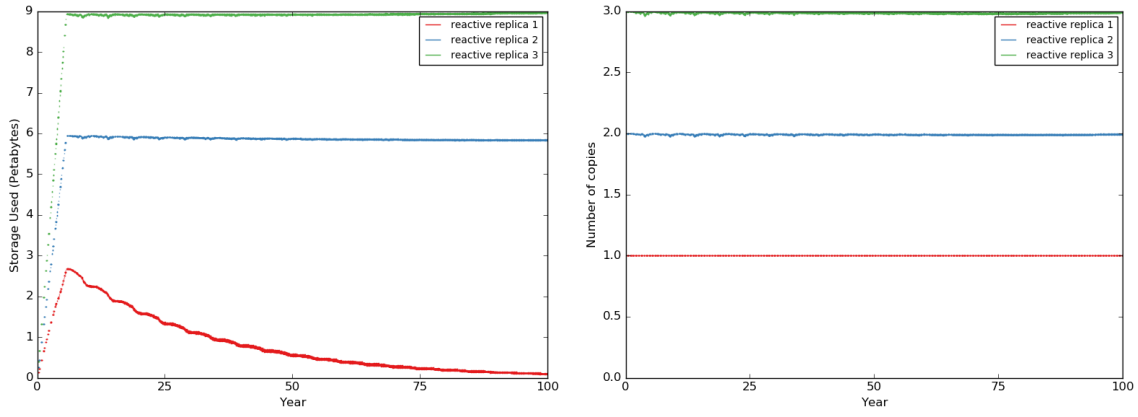


Figure 69. Storage and copies used over time for the reactive policies against the CFDR policy.

Disk Age Aware Proactive

The distribution of storage utilization and replica count, as provided in Figure 70 below, is far more intriguing and varied than what has been seen thus far. Starting with the storage usage of the 1-replica, 0.9 and 0.99-reliability policies, in Figure 71, just as their reliability curves were similar to the simple policy configurations so are their storage utilization curves. Both policies begin with a linear increase during the populate phase of 0.476 and 0.514 petabytes per year until 5.58 years when they achieve their maximum storage utilization of 2.74 and 2.78 petabytes. Once the populate phase has ended, they suffer exponential decay of -0.035 and -0.027 per year matching their data loss rates.

The 1-replica, 0.9-reliability maintains a constant one replica for every data object, never allocating more than a single replica. This is not unexpected as the data object reliability should never drop below for 0.9 for any single replica that would trigger proactive replication. Therefore, the policy degrades to the same behavioral characteristics of the 1-replica reactive policy.

The case for the 1-replica, 0.99-reliability policy, on the other hand, is quite a bit different as its boxplot time series for the number of copies reveals. Like the previous policy, each data object mostly has only one replica, but at key points, in time the reliability of the disks will drop below 0.99 triggering a sharp uptick in replication. Its storage curve being relatively smooth during these periods indicates that most of the replication is for data migration from one disk to another instead of boosting replication count. From the variability seen in the boxplots, the mean replica counts for each simulation vary widely depending

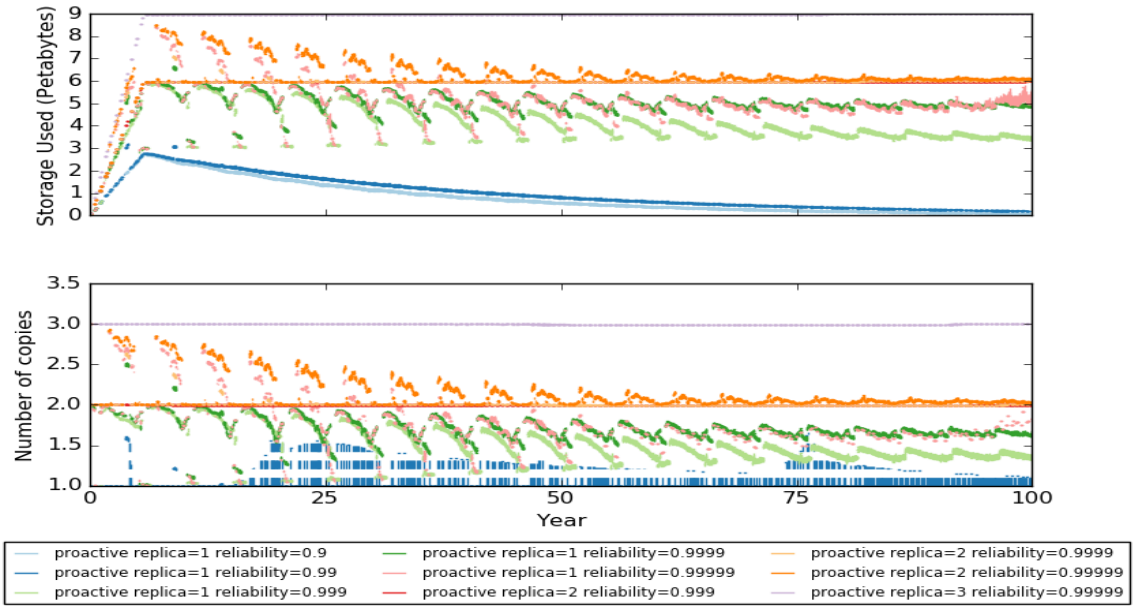


Figure 70. Storage and copies utilization over time for the CFDR age aware proactive policies.

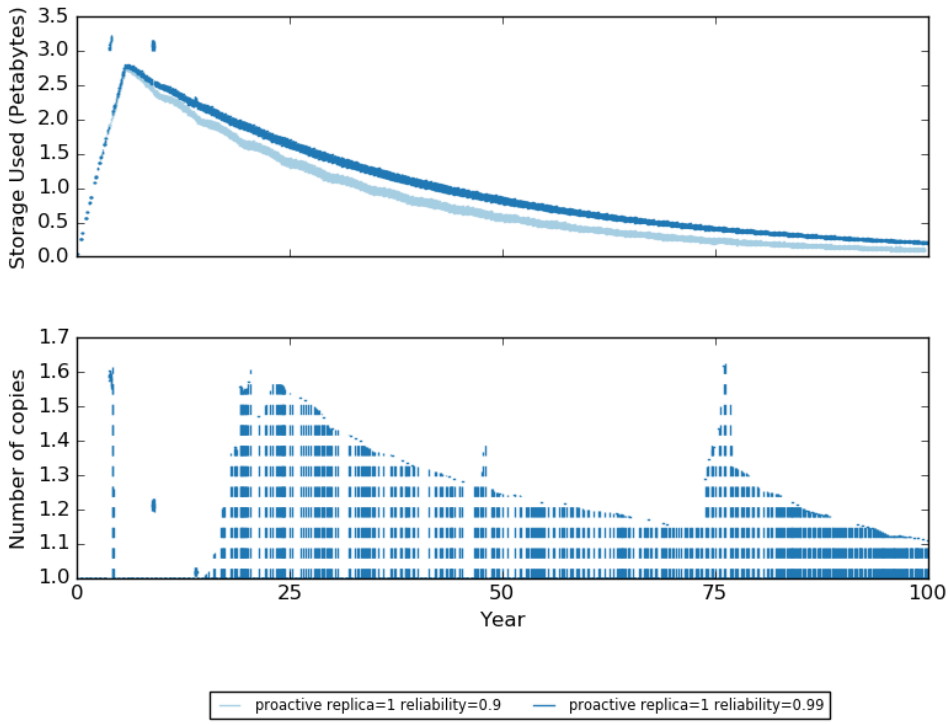


Figure 71. Storage and copies utilization over time for the 1-replica, 0.9 and 0.99-reliability CFDR disk age aware proactive policies.

upon the placement of the data objects. There are also frequently occurring "breaks" between the spikes when the mean replica counts for all simulations collapse to one. The cause of these fluctuations is from the disks being the same age and thus having the same reliabilities. Only at certain points in their lifetime does the reliability drop below the threshold triggering a response by the policy. Thus the policy has to respond to most of the disks in fixed interval sets. With that being said, this pattern diminishes over time with the breaks being less frequent and prolonged as the disk ages within the population become more staggered. The lifelong mean replica count was computed to be 1.0042 with a maximum mean replication count around 1.6.

The remaining three 1-replica policies, with reliabilities of 0.999, 0.9999 and 0.99999, have similar storage and replica characteristics to one another. Both the storage and replica utilization curves for all three exhibit a pronounced "sawtooth" phenomena alternating between peaks and troughs as seen in Figure 72. For storage utilization, it does not become evident until after the populate phase but nearly immediately for replica usage. This behavior is caused by the changes in disk reliability over time combined with the uniformity of the disk ages. Unlike the 1-replica, 0.99-reliability that manifested more of an individual replica count spread, the increased reliability requirements of these policy configurations resulted in a far more sensitive replication trigger that affected all of the data

objects. This serves as the rationale for the sharp rises and declines and the small spread witnessed within each curves' boxplots. Additionally, it explains why the effect is more drastic with higher reliabilities and fades with time. As the overall population ages, the individual disk ages become more varied.

Storage utilization rises linearly during the populate phase with 0.971, 0.843 and 0.95 petabytes gained per year. After peaking, each of the policies begins the "sawtooth" pattern. Despite the shifting nature, regression analysis reveals that each of these storage curves has a gradual linear decrease of -16.3, -4.74, -11.9 terabytes per year in order of ascending policy reliability. The maximum storage consumption witnessed in petabytes for each policy was 5.877 at 6.83 years for the 1-replica, 0.999-reliability; 6.613 at 8.83 years for 1-replica, 0.9999-reliability and 8.393 at 6.92 years for 1-replica, 0.99999-reliability.

Throughout the time series, each of the policies have a minimum mean replica count of 1.00 but an increasing maximum mean of 1.9997, 2.51 and 2.9994 as their reliability requirement increases. Despite the shifting replication count, there is a gradual decline in the average number of replicas of -0.0034, -0.0016 and -0.0045 replicas per year. Over time the "sawtooth" effect gradually diminishes, and the high and low replica counts begin to converge towards a central value. The exact value is impossible to find from the data available, but some rudimentary analysis would indicate around 1.4, 1.7 and 1.7 replicas for the three policies in order.

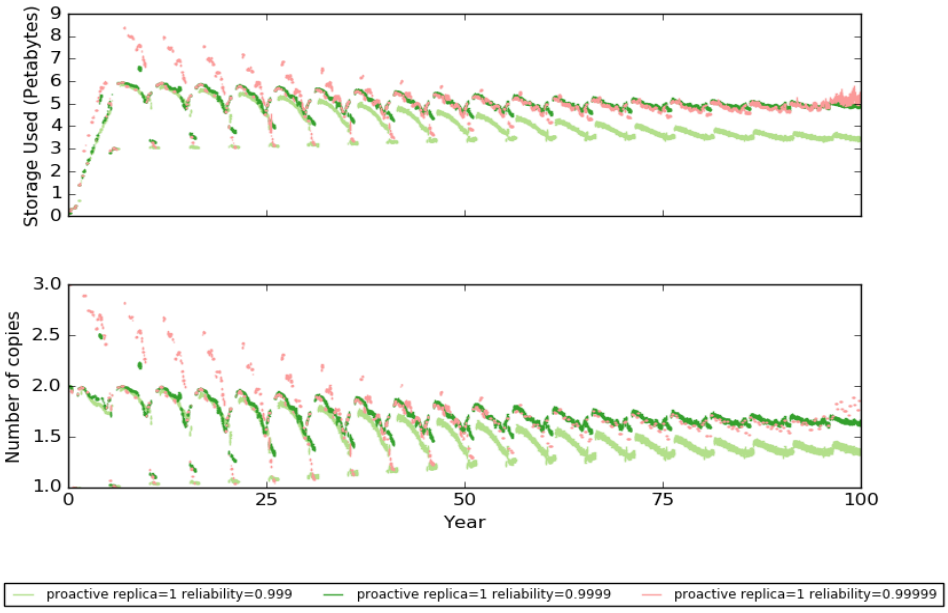


Figure 72. Storage and copies utilization over time for the 1-replica, 0.999, 0.9999 and 0.99999-reliability CFDR disk age aware proactive policies.

The storage utilization and replica curves for all of the 2 and 3-replica proactive policies are given in Figure 73. From the graphs, two different patterns emerge for the four proactive policies. The 2-replica, 0.999-reliability and the 3-replica, 0.99999-reliability policies follow the first pattern. For storage utilization, they have the initial rising linear slope as LoDN is being populated and then their storage utilization flatlines and remains consistent for the duration of the runs. The linear rise for the 2-replica, 0.999-reliability policy is 1.05 petabytes per year before steadying out at a modest annual 0.2 terabyte increase. Since the policy has a constant increase in storage usage, the maximum mean storage utilization does occur until near the end at 99.83 years with 5.99 petabytes. The 3-replica, 0.99999-reliability policy follows an increase of 1.58 petabytes a year during the populate phase and then transitions to a fairly constant 0.52 terabytes per year rise. Their replica usage likewise follows suit maintaining a near constant replica load matching their corresponding minimum replica requirements. The 2-replica, 0.999-reliability policy ranges between 1.9937 and 2.0 copies, while the 3-replica, 0.99999-reliability policy varies from 2.99203 to 2.99999. The long term replica count slope is 0 for both policies.

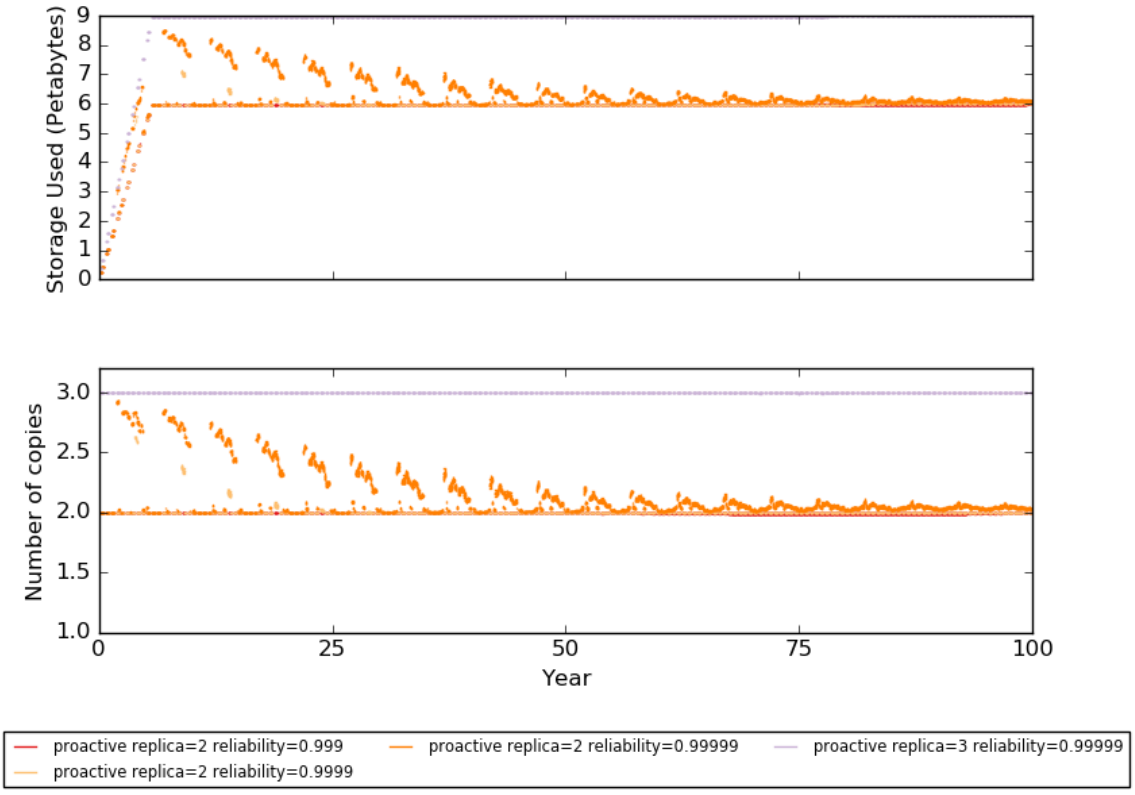


Figure 73. Storage and copies utilization over time for the 2 and 3 replica CFDR disk age aware proactive policies.

Starting with the 2-replica, 0.9999-reliability policy and more so with the 2-replica, 0.99999-reliability policies, the "sawtooth" pattern re-emerges with storage and replica utilization rising and falling periodically. As expected, the 2-replica, 0.9999 and 0.99999-reliability policies both have a linear rise in storage consuming an additional 1.08 and 1.19 petabytes per year, respectively, during their populate phase. Early on during their normal life period at 8.75 and 7.00 years, they achieve their maximum storage utilization of 7.09 and 8.49 petabytes each. Despite the jagged peaks in storage consumption, the 2-replica, 0.9999-reliability policy has a long-term near zero change in storage usage, and the 2-replica, 0.99999-reliability policy loses -4.76 terabytes annually. The two policies' mean replica count fluctuates heavily with time with a breadth of 1.9971 to 2.6277 replicas with an approximately zero slope change for the former and 1.9996 to 2.9996 replicas with a small decline of -0.0021 per year in the case of the latter.

For the first pattern, neither of the minimum reliabilities of the policy settings are high enough to be triggered by low disk health intervals; therefore, the policies' behavior degrades to being more like the reactive policies. Beginning with the 2-replica, 0.9999-reliability policy, the intervals of lowest disk health trigger data migrations and replication. This becomes visually even more pronounced in the case of the 2-replica, 0.99999-reliability policy. Over time the fluctuations settle out and converge as the disk population becomes more diversified and the policies have spread the data objects across the differing disks.

Visually, comparing the replica and storage utilization curves is rather arduous given the complexity of several of the curves generated. To simplify the comparison of the storage and replica requirements for the policies, the integral of the replica count over time, called replica-seconds, was computed for each policy. Table 31 provides a short summation of the minimum and maximum mean replica count for all of the simulations of each policy configuration as well as the replica-seconds integral.

The "sawtooth" appeared in the 1-replica, 0.999, 0.9999 and 0.99999-reliability policies as well as the 2-replica, 0.9999 and 0.99999-reliability policies. For these three, 1-replica proactive policies, their total replica-seconds are well below that of the 2 and 3-replica reactive policies despite the proactive policies having those intermittent replication peaks.

The two, 2-replica proactive policies, also have significantly smaller replica-second results than what would be expected from their maximum storage utilizations. Both policies alternate replica counts from just below 2 to as high as 2.63 and nearly 3 replicas, respectively. Their total replica-seconds though are 6.31 and 6.77 which is far closer to the 2-replica reactive policy's value of 6.28 than the 3-replica reactive policy's value of 9.42. This is important as it indicates that their storage usage is far more similar to the 2-replica reactive. For storage systems that have some flexibility in their allotments, this realization could drastically improve the usefulness of these policies especially since their

Table 31. The minimum, maximum and integral replica-seconds for each of the CFDR policies.
The proactive policies with the "sawtooth" behavior are highlighted.

Policy	Min	Max	Replica-Seconds (in billions)
Simple			
1-replica Simple	1.0	1.0	3.1531
2-replica Simple	1.01587	1.99899	3.8465
3-replica Simple	1.03277	2.99827	4.6027
Reactive			
1-replica Reactive	1.0	1.0	3.1531
2-replica Reactive	1.97778	2.00000	6.2826
3-replica Reactive	2.96639	2.99999	9.4187
Proactive			
1-replica, 0.9-reliability	1.0	1.0	3.1452
1-replica, 0.99-reliability	1.0	1.58934	3.1665
1-replica, 0.999-reliability	1.0	1.99977	4.7168
1-replica, 0.9999-reliability	1.0	2.50662	5.3627
1-replica, 0.99999-reliability	1.0	2.99937	5.5012
2-replica, 0.999-reliability	1.99373	2.00000	6.2945
2-replica, 0.9999-reliability	1.99706	2.62773	6.3145
2-replica, 0.99999-reliability	1.99959	2.99960	6.7665
3-replica, 0.99999-reliability	2.99203	2.99999	9.4453

reliability is on par with the 3-replica reactive policy. This will be revisited in the next section when discussing the NOMDL for these policies.

Normalized Magnitude of Data Loss

Continuing with the CFDR analysis, the resulting NOMDL for each policy is computed in Table 32 to examine the interplay between storage usage and reliability. For NOMDL computation, the elements in the table below were defined as follows. The replica count used for the simple and reactive policies is the replication level requested by the policy configuration. Since the proactive policies' replication level is dynamic, the maximum of the mean replication level occurring within the 100-year timeframe is used, as it is both the worst case and the minimum level of storage that the infrastructure must support. The D element is the usable storage apparent to the end users consisting of the three petabytes for all of the data objects and any free storage that might be useable. Since the highest replication level experienced by each policy configuration was chosen, D then will be the smallest quantity of storage visible to the users. Therefore, the resulting NOMDL is guaranteed to be the worst case for the proactive policy with the NOMDL numerator being the mean lost data objects and the denominator the lowest storage available regardless of when that occurs in the simulation's timeframe.

Despite the slight improvement in reliability as the replication level increases for the simple policy, 184,513 fewer objects lost transitioning from the

Table 32. NOMDL for CFDR policies.

Policy	Lost Data Objects	Replica Count	Storage Used (PB)	Storage Free (PB)	MDL (PB)	D (PB)	NOMDL
1-replica Simple	2,903,667	1.00	3.00	6.22	2.90	9.22	0.31507
2-replica Simple	2,809,720	2.00	6.00	3.22	2.81	6.22	0.45201
3-replica Simple	2,719,154	3.00	9.00	0.22	2.72	3.22	0.84551
1-replica Reactive	2,903,853	1.00	3.00	6.22	2.90	9.22	0.31509
2-replica Reactive	69,830	2.00	6.00	3.22	6.98	6.22	0.01123
3-replica Reactive	386	3.00	9.00	0.22	3.86	3.22	0.00012
1-replica 0.9-Reliability	2,901,223	1.00	3.00	6.22	2.90	9.22	0.31480
1-replica 0.99-Reliability	2,777,097	1.59	4.77	4.45	2.78	7.45	0.37296
1-replica 0.999-Reliability	443,941	1.99977	5.999	3.22	0.444	6.22	0.07141
1-replica 0.9999-Reliability	1,198	2.51	7.53	1.69	0.0012	4.69	0.00026
1-replica 0.99999-Reliability	447	2.99937	8.998	0.218	0.000447	3.22	0.00014
2-replica 0.999-Reliability	1,109	2.00000	6.00	3.22	0.00111	6.22	0.00018
2-replica 0.9999-Reliability	815	2.62773	7.88	1.33	0.000815	4.33	0.00019
2-replica 0.99999-Reliability	594	2.99960	8.999	0.217	0.000594	3.22	0.00018
3-replica 0.99999-Reliability	0.2899	2.99999	9.00	0.216	2.9e-07	3.22	0.00000009

1-replica to the 3-replica configuration, the corresponding NOMDL worsens significantly. By tripling the storage requirement, the simple policy's NOMDL degrades by a factor of 2.68 times. Obviously, had the reliability not improved, then it would have been a full three times worse. Even though the reactive policy also increases by three petabytes for each replication level, its NOMDL scores improve due to the drastic increase in reliability that each new replica provides. Starting at the same NOMDL of 0.3151 as the 1-replica simple policy, the reactive achieves 0.00012 (2,625 times improvement) for three replicas.

For the 1-replica proactive configurations, all but the 0.99-reliability beats the 1-replica reactive policy. The 1-replica, 0.99-reliability performs worse because although it loses 126,000 fewer data objects, it also has three separate storage usage peaks that top out at 1.59 mean replicas. The first three 1-replica proactive configurations all score poorer than the 2-replica reactive policy due to higher data loss. Initially, this is somewhat surprising for the 1-replica, 0.999-reliability policy since its count replica is almost at 2. However, referring to Figure 72, the replica count used is caused by some early peaks in the "sawtooth" pattern and the replica count eventually converges beneath a mean of 1.5. The same effect is witnessed in the 1-replica, 0.9999-reliability policy with it having significantly better reliability but suffering early peaks of 2.5 replicas. The

long-term replica count for the 1-replica, 0.9999-reliability policy settles out to 1.7 replicas. Using this value, the proactive would outperform the 2-replica reactive policy on all three metrics: reliability, storage and NOMDL.

The performance of the 1-replica 0.99999-reliability is competitive with the 3-replica reactive. At first glance, it would appear that the explanation is due to a nearly matching replica count of 3, but as before this is deceptive. The proactive policy peaks early in the lifetime of the simulation eventually converging around 1.7 replicas.

An interesting phenomenon occurs among all of the 2-replica proactive policy configurations. Despite an improving data loss rate, their NOMDL scores are essentially the same because of their rapidly rising replication needs. The final configuration requires an entire replica more per data object than the starting configuration to maintain the reliability requested.

The 2-replica proactive policies, all have 62 times better NOMDL scores than the equivalent 2-replica reactive. This includes the first 2-replica proactive policy whose worst case storage usage matches the 2-replica reactive. Specifying that the two disks a data object resides on must have an aggregate reliability of 0.999 is sufficient to gain the 62 times improvement factor. While not performing quite as well as the 3-replica reactive policy due to higher data loss, the 2-replica proactive policies still achieve a NOMDL values that are consistently strong against the 3-replica reactive result (0.00018 vs. 0.00012) due to storage savings even at peak usage.

Although the 3-replica, 0.99999-reliability policy uses as much storage as the 3-replica reactive for the duration of the 100-year period, the proactive policy is 1,333 times better with almost zero data loss.

In Table 33 below, the NOMDL is recomputed for the proactive policies using the mean replica counts from the replica-second statistics in Table 31. Dividing the replica-seconds by the 100-year duration of the simulation provides the long-term mean count of the replicas per data object. Creating a new round of NOMDL calculations based on the average replica count assists in removing some the distortion generated by the “sawtooth” peaking phenomena. All of the affected policies saw drastic improvements in their NOMDL scores. The 1-replica, 0.999, 0.9999 and 0.99999-reliability policies saw improvements of 19.7%, 34.6% and 57.1%, respectively. The 2-replica, 0.9999 and 0.99999-reliability policies had a 31.6% and 44.4% increase.

These changes affected the NOMDL rankings stated previously. Now all of the 1-replica proactive policy configurations outperform the 1-replica reactive policy. The 1-replica, 0.9999 and 0.99999-reliability increase their performance over the 2-replica reactive from 43 and 80 times to 66 and 187 times. The 2-replica proactive policies also enhance their results against the 2-replica reactive between 62 to 112 times. Finally, both 0.99999-reliability proactive policies now achieve better NOMDL scores than the 3-replica reactive. The 1-replica, 0.99999-reliability policy overtakes the 3-replica reactive policy by a factor of 2

with 1.74 mean replicas, and the 2-replica, 0.99999-reliability policy, requiring 2.14 mean replicas, is 1.2 times better the 3-replica reactive policy.

An important caveat to these adjusted results, though, is that sufficient storage must still be supplied to the storage system when these peaks occur in order for the policies to continue to provide their level of resilience. This could be achieved by either purchasing additional storage in times of need or already having slack flexible storage that is used for other purposes until needed. Another alternative is to make sure that the disks within the system are of varied ages similar to what occurs as the simulation progresses and not all of the same starting age.

Table 33. Mean NOMDL for CFDR policies.
The policies exhibiting the "sawtooth" pattern are highlighted.

Policy	Lost Data Objects	Replica Count	Storage Used (PB)	Storage Free (PB)	MDL (PB)	D (PB)	NOMDL
1-replica 0.9-Reliability	2,901,223	1.00	2.99	6.23	2.90	9.23	0.31446
1-replica 0.99-Reliability	2,777,097	1.00	3.01	6.21	2.78	9.21	0.30167
1-replica 0.999-Reliability	443,941	1.49	4.484	4.73	0.444	7.73	0.05742
1-replica 0.9999-Reliability	1,198	1.70	5.10	4.12	0.0012	7.12	0.00017
1-replica 0.99999-Reliability	447	1.74	5.230	3.99	0.000447	6.99	0.00006
2-replica 0.999-Reliability	1,109	1.99	5.98	3.23	0.00111	6.23	0.00018
2-replica 0.9999-Reliability	815	2.00	6.00	3.21	0.000815	6.21	0.00013
2-replica 0.99999-Reliability	594	2.14	6.433	2.78	0.000594	5.78	0.00010
3-replica 0.99999-Reliability	0.2899	2.99	8.98	2.37	2.9e-7	3.24	0.00000009

Bandwidth Used

Simple

The results of the data flow analysis begin with the simple policy's three configurations. The stacked bar graphs in Figure 74 provide a breakdown of the different data flow types; wide, local and bus-only; for the three different replica requirement configurations.

The total data flow utilized by the 1-replica policy is three petabytes equaling the three petabytes of data from the local client data uploads. Without any need to perform further replication there is no need for bus-only or wide area traffic. The 2-replica policy has the same local area traffic quantity and pattern for the initial upload but then generates nearly another three petabytes of wide

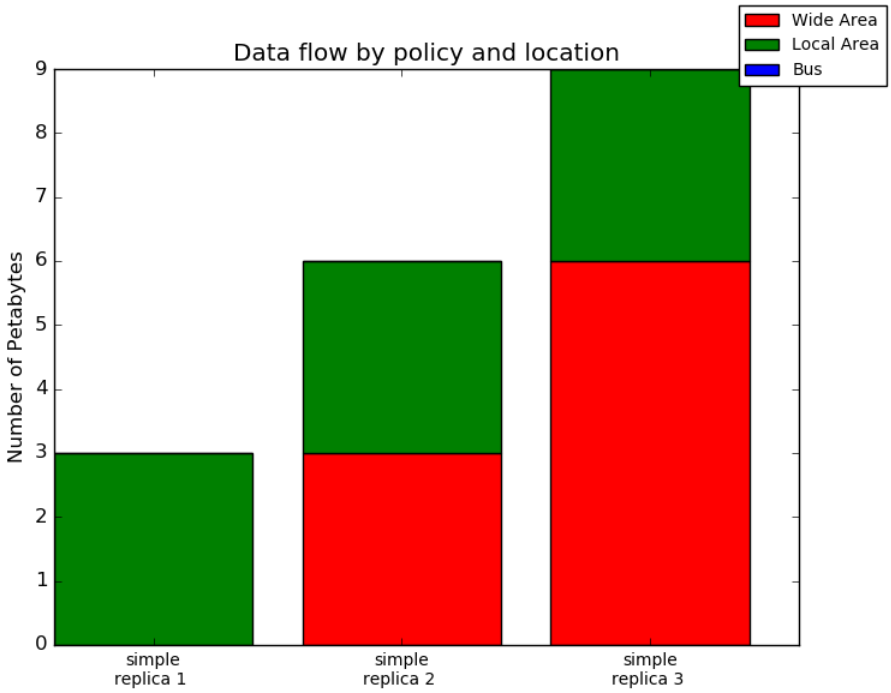


Figure 74. Simple policy's data flow on the CFDR age model.

area traffic by executing a replication to another site. The 3-replica policy follows suit performing one more additional inter-site replication bringing the total data flow to nearly nine petabytes.

The mean wide area traffic of the 2 and 3-replica policies both fall short of their expected targets of six and nine petabytes due to data loss following the initial upload. Typically, this occurs to 0.17 data objects per simulations with a maximum of 2 data objects being witnessed in any run for the 2-replica policy. Somewhat counterintuitively, the 3-replica appears to suffer a greater loss before being able to achieve full replication status. There is a mean loss of 1.1 data objects in any such run and a maximum early loss of five data objects ever experienced. The higher bandwidth requirements for generating three replicas may be responsible for the increase in data loss. As certain network links become saturated with traffic, the duration between the initial upload and subsequent replications lengthens making the data objects more susceptible to data loss.

Reactive

The results of the various data flows of the reactive policy configurations are presented in Figure 75. The reactive policies have an almost identical local traffic flow as the simple policies with the 1 and 2-replica reactive both generating three petabytes of local area traffic for the initial upload of data. The 3-replica

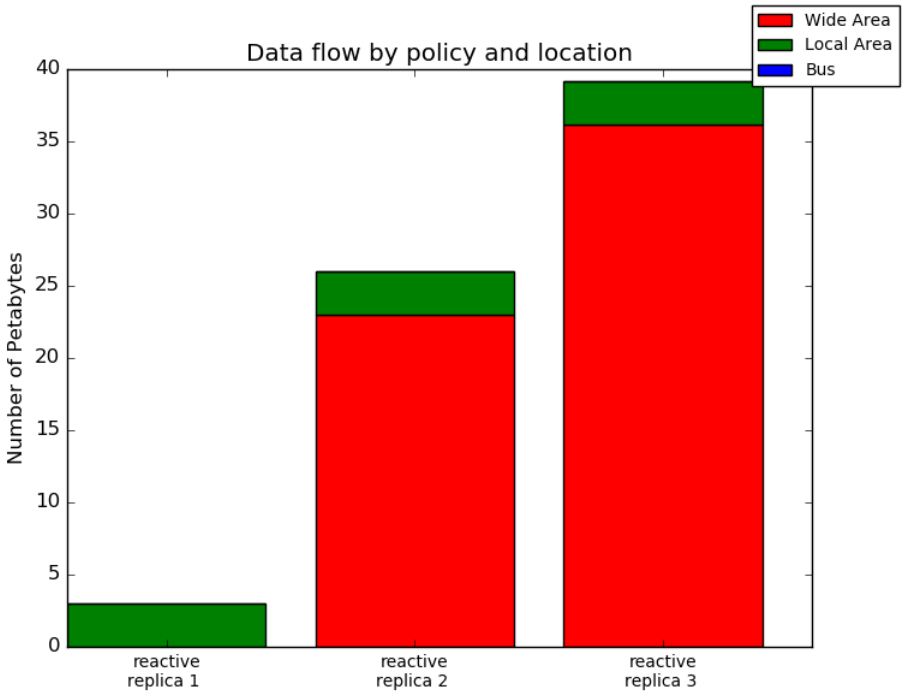


Figure 75. Reactive policy's data flow on the CFDR age model.

reactive policy fails to be able to restrict itself in several simulation instances to only the local area for the data injection. With three required replications per data object and replication repair creating storage imbalances, several of the sites exhausted all available storage near the end of the populate phase. For each simulation, this occurs between 0 to an extreme of 3,285 data objects with an average around 43 data objects.

The appreciable difference between the 2 and 3-replica reactive and the simple policies is the amount of wide area traffic. The 2 and 3-replica reactive policies use the same three and six petabytes for the initial replication but then generate an additional 20 and 30 petabytes of traffic over the wide area for replica recovery. This works out as 20 and 30 million replication operations to attempt to maintain 3 million data objects over the course of 100 years.

There is no bus-only data movement in any of the simulations since the reactive policies only react to failure and have to generate replacements from other sites.

Disk Age Aware Proactive

The data flows for eight of the nine proactive policies is provided below in Figure 76. Unfortunately, the data flow information for the 1-replica, 0.99999-reliability policy was unavailable due to errors with the early checkpointing implementation used for the associated simulations. Since bus-only traffic

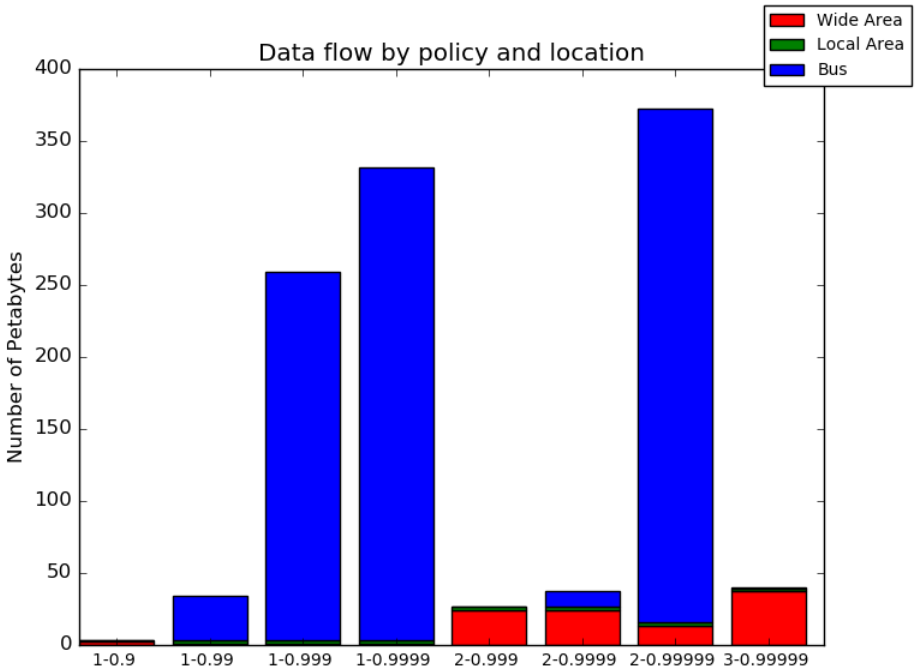


Figure 76. Data flows for the proactive policy for the CFDR age model.

dominates the bar graph in Figure 76, a secondary graph is presented in Figure 77 that excludes the bus traffic to better illustrate the other data flow patterns.

For the 1-replica policies, several trends emerge as the minimum reliability requirement rises. For all four policies, the total of the local and wide area traffic are nearly identical at three petabytes suggesting that the policy used all of the local and wide area traffic recorded for the initial upload. All of the bus traffic witnessed was for maintaining the reliability of the data objects with the policies not having to resort to the local or wide area for data migration. This does appear to begin to break down however for the final 1-replica policy generating an extra 558.2 gigabytes (558 replications) in local traffic for data migration. As the reliability increases, the ratio of local to wide area increases from 0.12 to 1.72 then to 2.74 and finally 3.28. Unlike the reactive policy, not all the initial uploads occur at the originating site.

There is a drastic rise in bus-only traffic for the 1-replica proactive policies corresponding to their reliability setting from 0 petabytes for 0.9-reliability, 31.4 petabytes for 0.99-reliability, 255.8 petabytes for 0.999-reliability and finally to 328.4 petabytes for the 0.9999-reliability policy. With only the bus traffic increasing, these proactive policy configurations are placing the onerous of maintaining reliability solely on intra-depot copying from disk-to-disk. Since roughly 2/3 of the total storage space is unoccupied, there is plenty of free

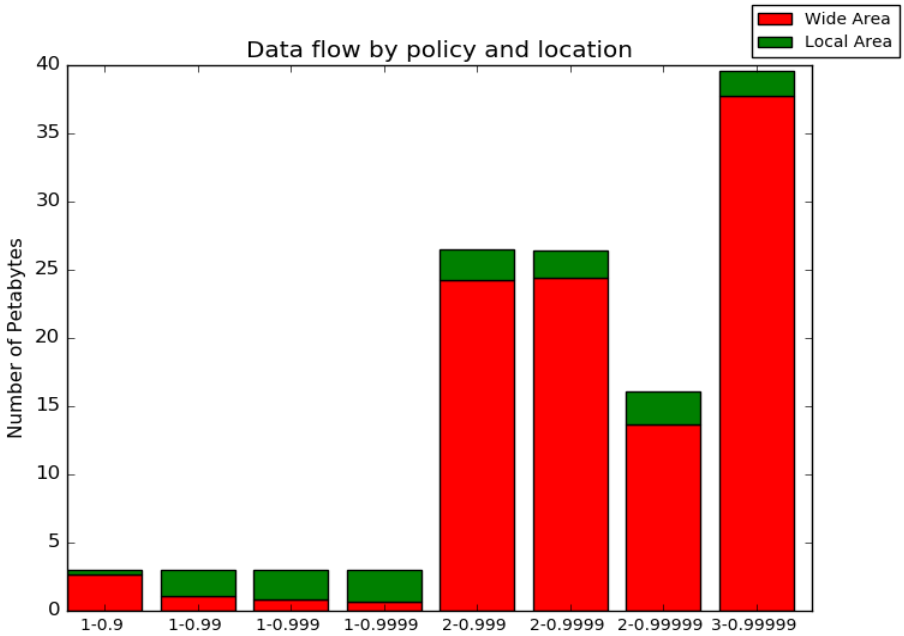


Figure 77. Local and wide area data flows for the proactive policies.

storage at each depot for data migration, so the policy is rarely forced to look beyond the other disks on the depot where the data object resides.

With the reasonable assumption that all bus traffic for the 1-replica policies is from reliability driven migrations and replications, then computing the number of replications is simple. The 0.9-reliability policy performs 0 data replication operations which makes sense since the reliability of an individual disk under the CFDR Age model never drops that low. Proceeding to the 0.99-reliability policy, the number of replication operations for data migration rises to an average of 31,447,634 before escalating to 255,776,237 for 0.999-reliability and lastly to 328,351,107 for the 0.9999-reliability. If every data object survived, then this would mean that the 0.9999-reliability, migrated or replicated every data object an average of 109 times over the course of the simulation.

The precise behavior of the 2 and 3-replica proactive policies is harder to decipher but consider the following information. Except for the 2-replica, 0.99999-reliability policy all the other policies have significantly less bus-only traffic than the 1-replica policies: 2.4 petabytes for the 2-replica, 0.999-reliabilty; 11.4 petabytes for the 2-replica, 0.9999-reliability and only 0.11 petabytes for the 3-replica, 0.99999-reliability policy. There is very little variability in the quantity of local area traffic for these policies ranging from 1.8 to 2.4 petabytes. Both the 2-replica, 0.999 and 0.9999-reliabilty have nearly the same wide area traffic with 24.3 and 24.4 petabytes, respectively. The 3-replica, 0.99999-reliability policy has the highest wide area traffic with 37.7 petabytes. 2-0.99999 has a significant

drop in wide area traffic, only 13.6 petabytes, compared to other 2-replica policies while correspondingly having a large uptick in bus-only traffic. From this information, it could be concluded that only the 2-replica, 0.99999-reliability policy has a high enough reliability metric to trip data migration and duplication at a local site before data loss occurs at a site. This was starting to be the case, to a much smaller degree, in the 2-replica, 0.9999-reliability policy with its small bus usage uptick. With 2 to 3 replicas, the aggregate reliability of the multiple disks is sufficiently high enough that the two other policies do not perform much pre-emptive data movement; instead, they are mostly waiting for failure to do replacement from across the wide area like the reactive policy.

The 2 and 3-replica proactive tend to create only 60 - 80% of the local area traffic compared to the corresponding reactive policies; however, apart from the 2-replica, 0.99999-reliability policy, they generate up to 1.1 times more wide area traffic, a 1.5 petabyte increase. Like the 1-replica proactive, these policies appear to use the wide area for handling the initial upload more, explaining some of the growth in wide area traffic. The similarity in traffic pattern between the proactive and reactive policies adds credence to the notion that the 3-replica and the first two 2-replica proactive policies behave more reactively. An important exception is the 2-replica, 0.99999-reliability policy generating only 40 – 60% of the wide area traffic, but due to the increase in bus-only traffic, it has 14.3 and 9.5 times the total traffic.

Based on the traffic numbers, the proactive policy performs the following number of replications per configuration: 26,529,304 for the 2-replica, 0.999-reliability, 37,801,298 for the 2-replica, 0.9999-reliability, 39,674,421 for the 3-replica, 0.99999-reliability and 372,771,673 for the 2-replica, 0.99999-reliability. For comparison, the reactive configurations for the 2 and 3-replica were 25,985,910 and 39,171,316.

Elerath Vintage

This final section reviews the results and provides an analysis of differing policy configurations using the Manufacturer, Model, Vintage failure model created from Elerath's research. As with the other the failure models, three replication variations of the simple and reactive policies were first conducted to determine how the current approaches would fare and to provide a baseline for comparison for the model specific proactive policy. For the proactive policy eight different configurations were simulated consisting of the following settings for minimum number of replicas and 2-month, look ahead conditional reliability: 1, 0.99; 1, 0.999; 1, 0.9999; 1, 0.99999; 2, 0.9999; 2, 0.99999; 2, 0.999999 and 2, 0.9999999. The results for these fourteen configurations are examined in regards to the following aspects: data survival, storage utilization, replica requirements, bandwidth utilization and the NOMDL metric with the analysis covered in the following four subsections.

Data Survival

The results for the Elerath vintage model and policy begins with an analysis of the data survival and reliability of each policy grouped by policy type.

Simple

The collective reliability of the data objects under the care of the simple policy is presented in Figure 78 for all three replica settings. As expected, each of the policies suffer massive losses with mean losses being 2,746,600 for the 1-replica; 2,514,855 for the 2-replica and 2,302,021 for the 3-replica making the effective long-term survivorship 8.45, 16.17 and 23.27%. For every additional initial replication performed by the policy, the reliability, while still poor, was improved with over 200,000 data objects preserved for each replication. The reliability results are fairly consistent as shown by the boxplot possessing a small interquartile range and no overlapping between the different curves' whiskers. For the simulations performed the full loss range for the 1, 2 and 3-replica was from 2,710,710 to 2,787,784; 2,463,331 to 2,571,080 and 2,225,176 to 2,371,010 respectively.

Regression analysis reveals that the data object survivorship follows an exponential decay rate for all three configurations reviewed with their decay rate varying by their respective replication levels. With R^2 values of 0.96896, 0.99158

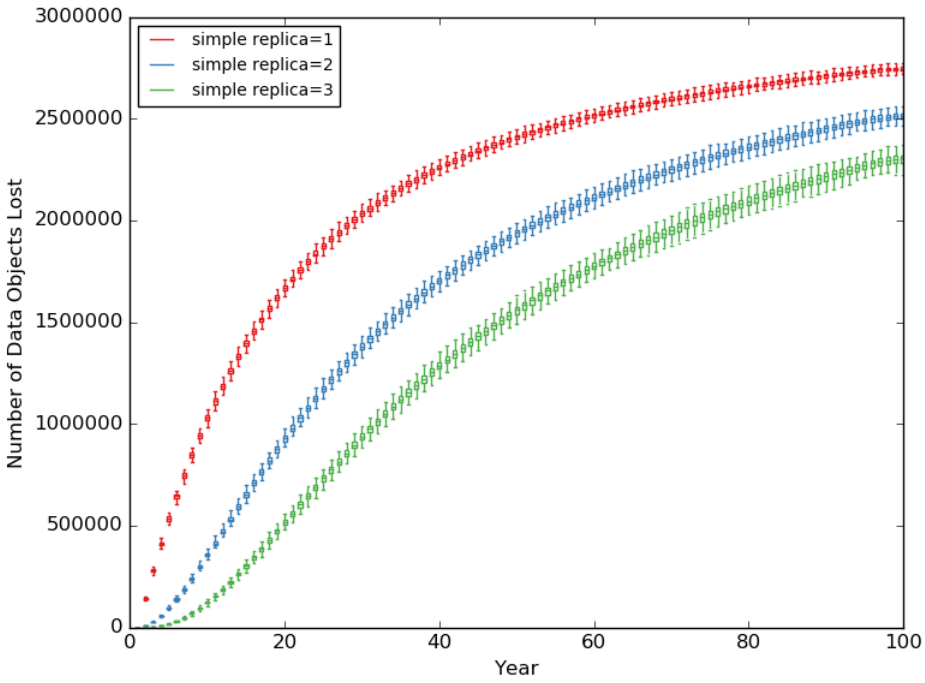


Figure 78. Reliability boxplots for the simple policy against the vintage failure model.

and 0.99705, the decays rates were -0.024, -0.019 and -0.016 per year. As the replication level rises the curves start flattening at the onset forming the “S” shape of a logistic sigmoid curve.

Reactive

Moving on to the reactive policy, the reliability results for the three configurations are shown below in Figure 79. The graph on the left of the figure provides the distribution all of three curves while the graph on the right has a close up of just the 2 and 3-replica reactive policies. Readily apparent is the poor reliability performance of the 1-replica reactive in comparison to the reactive policies. It exhibits an exponential decay rate of -0.024 per year matching the decay rate of the first simple policy and a 100-year reliability of 8.42%

The other reactive policy configurations’ performance is significantly improved, transitioning from an exponential decay distribution to a linear distribution losing 22.62 data objects a year for the 2-replica and 0.0093 data objects a year for the 3-replica. The 2-replica reactive loses between 1,409 to 5,940 data objects for all of its runs. The 3-replica reactive ranges between 0 to 6 data objects lost per run with a mean loss of only 0.928.

As the figure reveals, the reliability of the of the 2-replica reactive is even better when analyzing the box plots. The median number of data objects lost is half thousand less than the statistical average of 2,170. There is a large standard deviation of 901 data objects lost and a severe upper skew caused by a few simulations. The reliability of the 2-replica policy works out to 99.928% and the 3-replica to 99.99997% over the 100-year period.

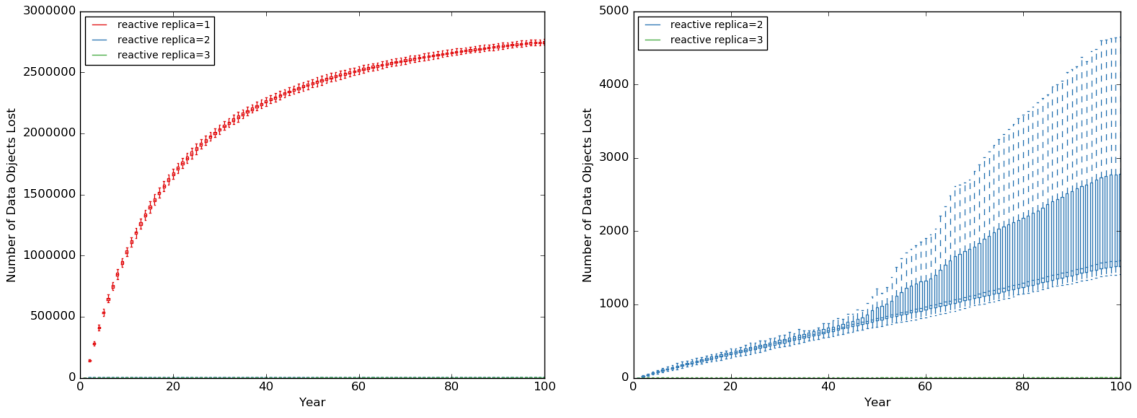


Figure 79. Reliability boxplots for the reactive policy against the vintage failure model. The graph of the left provides all the reliability distributions for the reactive policy, while on the right the graph provides a close-up of just the 2 and 3-replica reactive policies.

Vintage Aware Proactive

With an understanding of the reliability results for the simple and reactive policies for the vintage failure model, the analysis moves on to the eight vintage aware, proactive policies starting with the two worst performers: the 1-replica, 0.99-reliability and the 2-replica, 0.9999999-reliability policies. As shown in Figure 80, while these policies outperform the 1-replica reactive policy's reliability, they cannot match the reliability of the 2-replica reactive policy. The 1-replica, 0.99-reliability policy's reliability distribution is still best described using an exponential function with a decay rate of -0.014. While this decay rate is better than all of the rates of the simple policies and the 1-replica reactive policy, the fact this proactive policy is still exponential helps to exemplify its poor performance. Its 100-year reliability is 24.76% having lost a mean of 2,257,287 data objects over 1,000 times worse than the 2-replica reactive.

The final proactive policy, the 2-replica, 0.9999999-reliability, has a reliability of 90.74% with 277,718 data objects lost. While still worse than the 2-replica reactive, it is an interesting policy case because almost all of data loss for the policy occurs in the first year with an average of 277,483 and only 235 data objects hence. The data loss is caused by a lack of storage capacity to satisfy the required reliability since in the first year the disks have the greatest failure rate. Unable to provide allocations to incoming data objects, LoDN is forced to drop data objects. Ignoring the first year, the data loss is linear with 2.61 data objects per year lost.

The reliability of the other configurations drastically improves. Figure 81 provides a comparison of the reliabilities of the 1-replica, 0.999 and 0.9999-reliability policies versus the 2-replica reactive. Both policies overlap significantly with the 2-replica reactive policy, but their mean and median cases do edge out the reliability of the 2-replica reactive and both have a number of simulation outliers with no data loss. The average data loss suffered by these policies is

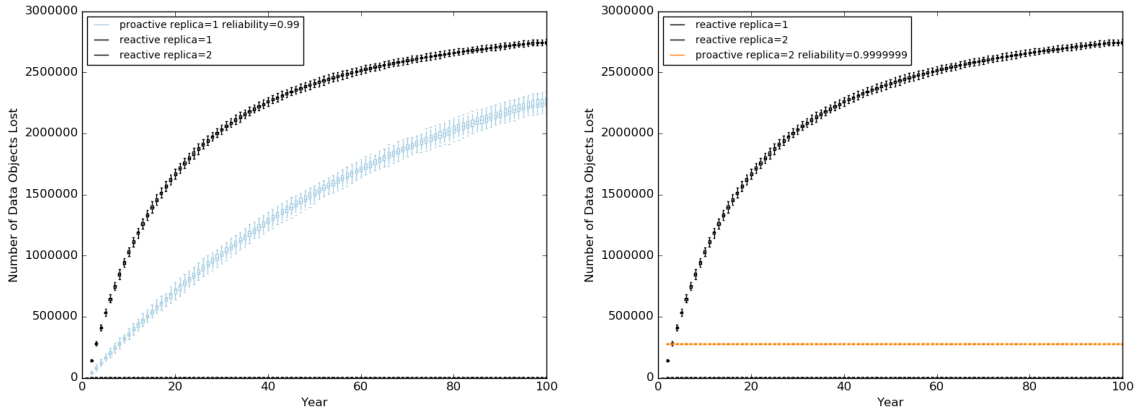


Figure 80. 1-replica, 0.99-reliability and 2-replica, 0.9999999-reliability proactive policies vs. 1 and 2-replica reactive policies.

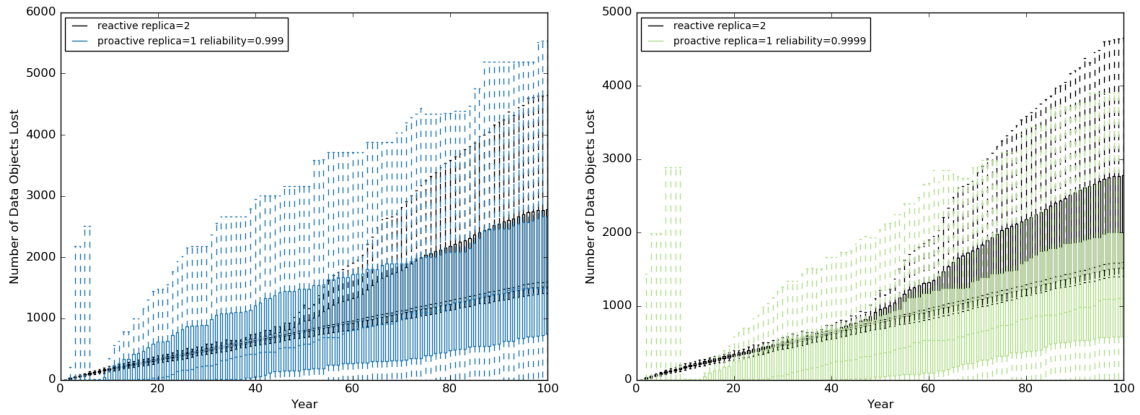


Figure 81. 1-replica 0.999 and 0.9999-relability proactive policies vs. 2-replica reactive Policy on the vintage model.

roughly 89% to 90% of the reactive policy. The average data object count loss is 1,945 and 1,936 affording them a reliability of 99.935%. Their loss rate is consistently linear at -19.205 and -19.667 data objects per year.

The next policy with 1-replica, 0.99999-reliability settings, in Figure 82, continues the increasing reliability trend, trumping the reliability of the 2-replica reactive and aside for several outliers becomes competitive with the 3-replica reactive. The policy loses a mean of 323 data objects, but a median of only four which is just three data objects more than the 3-replica reactive policy. Additionally, 46% of the simulations for the policy end with fewer than two lost data objects. Staying with the mean value, though, the reliability is 99.989% and is six times better than the 2-replica reactive policy but over 300 times worse than the 3-replica reactive policy. Linear regression computes a data loss slope of -1.6819 data objects per year which again indicates that this policy does do better than what the mean suggests.

Moving along to the 2-replica proactive policies, the next configuration for analysis is the 2-replica, 0.9999-reliability in Figure 83. From the graphs, it is apparent that this policy's reliability is very similar to the 1-replica, 0.9999-reliability policy. While the policy's mean reliability is better than the 2-reactive policy's having lost 0.76 times fewer data objects, the policy still has overlapping simulation results with the 2-reactive. The policy with a mean data object loss of 1,640 objects is over 1,700 times worse than the 3-replica reactive policy. With the same reliability setting as the 1-replica configuration, the addition of the extra mandatory replica improves overall reliability and stability. Almost 300 fewer data objects are lost per run, and the standard deviation also drops by 46%. All in all, this policy has a 100-year reliability of 99.945% with a linear loss rate of 14.392 data objects per year.

The reliability of the 2-replica proactive policies continues to improve as the conditional reliability increases to 0.99999. The mean data object loss falls to

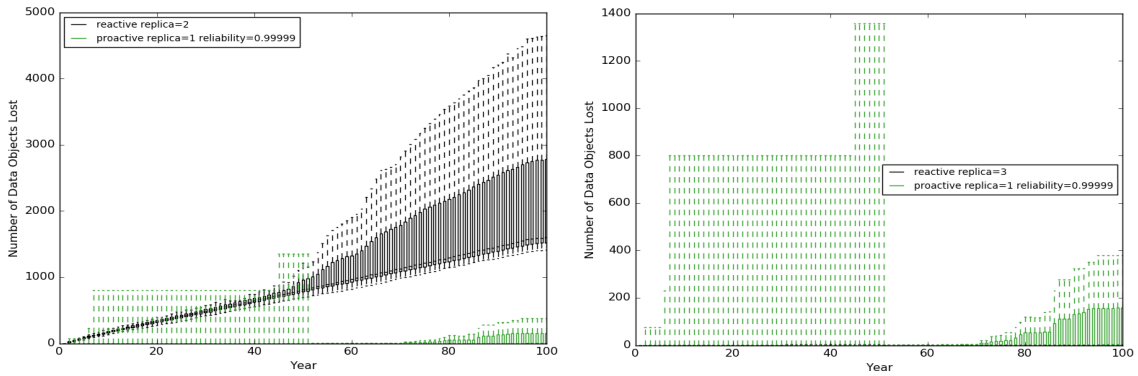


Figure 82. 1-replica 0.99999-reliability proactive policy vs 2 and 3-reactive policies on the vintage model.

274 and a median loss of only 28. The policy suffers only 0.126 times the loss of the 2-replica reactive. Clearly, as shown in Figure 84, the policy while still underperforming the reliability of the 3-replica reactive is approaching it with 28% of the simulations having one or fewer loss events. The reliability of the policy is 99.9901% with a linear loss rate of -1.6937 data objects per year. These results demonstrate that there is very little performance difference between the 1 and 2-replica, 0.99999-reliability policies having nearly identical reliabilities and loss rates. Even the standard deviation of these two policies possess less than a 6% difference.

The most resilient proactive policy configuration was the 2-replica, 0.999999-reliability. While still failing to meet or exceed the performance of the 3-replica reactive policy, it has a mean data loss of 89 and median of 58 objects. As such, it achieves a 100-year reliability of 99.997%. The general loss rate is -

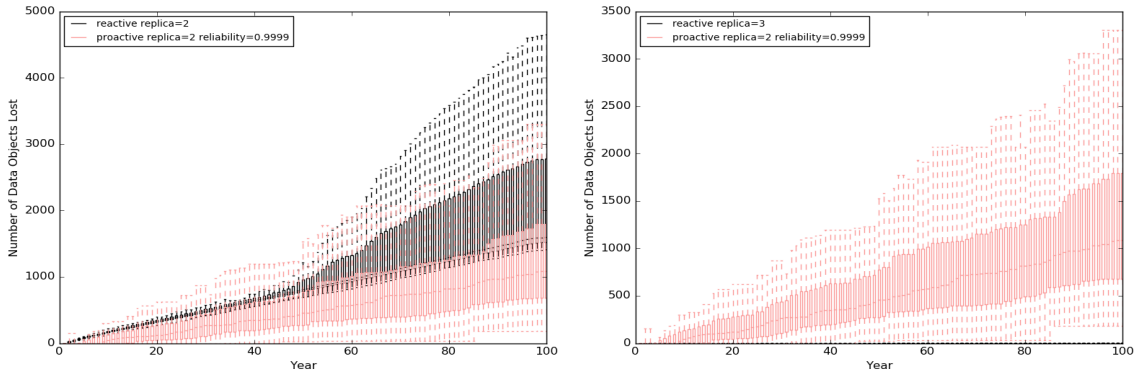


Figure 83. 2-replica, 0.9999-reliability proactive policy vs. 2 and 3-reactive on the vintage model.

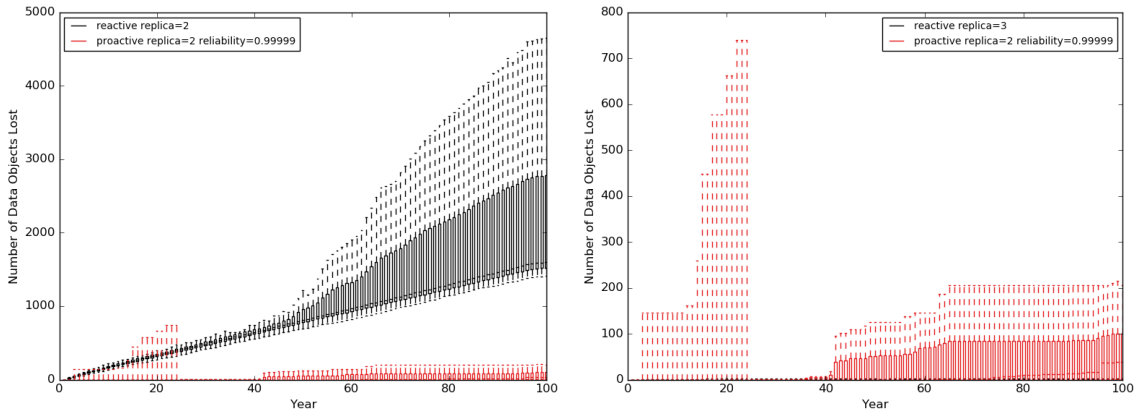


Figure 84: 2-replica, 0.99999-reliability proactive policy vs. 2 and 3-reactive policies on the vintage model.

1.2108 data objects per year. Figure 85 provides a comparison of the configuration against the 2 and 3-replica reactive policies.

Storage Used & Copies Used

Having covered the reliability of the policies, next is an examination of the storage and replicas counts required to achieve those reliability targets. This subsection starts with an analysis of these properties for the simple and reactive policy before comparing them to the proactive policy.

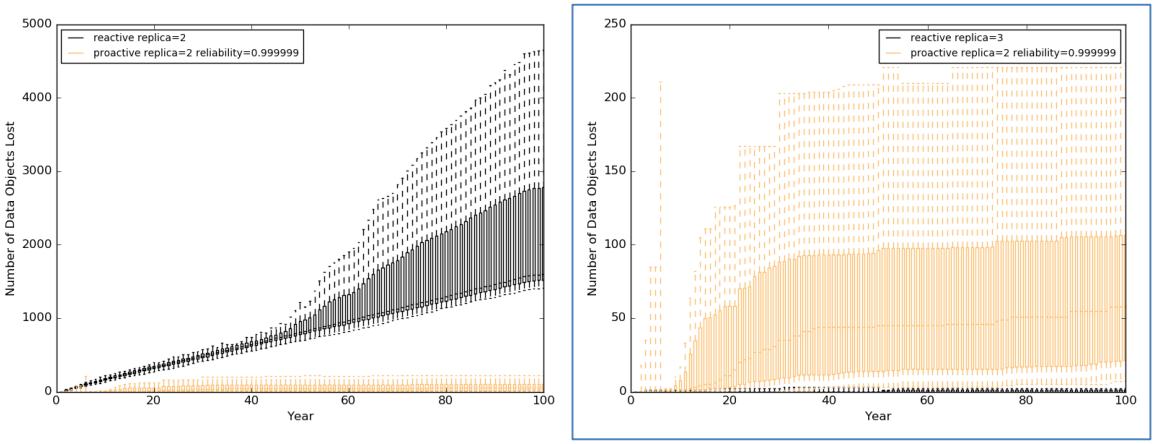


Figure 85. 2-replica, 0.999999-reliability proactive policy vs. 2 and 3-reactive policies on the vintage model.

Simple

Figure 86 provides the storage utilization (top graph) and the number of copies for the surviving data objects (bottom graph) for the three simple policy configurations. All of the simple policy configurations produce the same general storage utilization distribution shape varying only by their total storage usage peaks. The distributions can be divided into two distinct phases. A populate phase when the simulations are being initial seeded with data objects from the clients and the post-populate phase when seeding as stopped and all policy actions being performed are for the maintenance of the data objects. Of course, for the simple policy, there are no actions undertaken after the seeding phase.

Regression analysis reveals that the storage utilization of the three policies during the populate phase to be rising linearly. The 1-replica policy increases at a rate 0.45 petabytes per year, and the 2 and 3-replica have double and triple the rate with slopes of 0.90 and 1.35 petabytes per year. 0.526 petabytes of data are added per year to the simulation during the populate phase so even during this phase, the data loss is apparent in storage as the slopes should be multiples of that quantity. The maximum storage utilization for the three policies comes at 5.58 years as the populate phase ends. The three policies consume 2.60, 5.20 and 7.80 petabytes at this point well short of their possible maximum of 3, 6 and 9 petabytes. This suggests that roughly 13% of storage has failed for each of the policies after initial use.

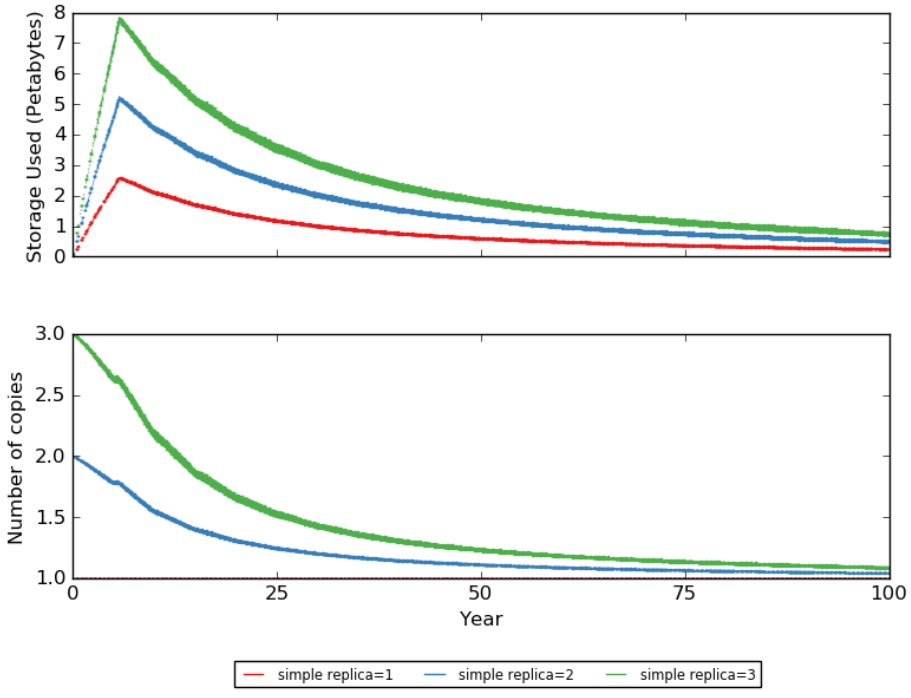


Figure 86. Storage and copies used over time for the simple policies.

In the post-populate phase, each policy displays the same exponential decay rate of -0.0232 per year through the rest of the simulations. By the end, the policies are only using 0.25, 0.50 and 0.76 petabytes for the remaining data objects.

The number of replicas per data object starts at the corresponding replication level for each policy but then begins an immediate descent. Not even additional replicas for newly arriving data objects affects the trend. The 1-replica policy remains at one copy for the duration of the simulation since the instant a data object drops beneath that count, it no longer exists. The replica counts for the 2 and 3-replica policies logarithmically approach that value with rates of -0.221 and -0.46, respectively. The mean ending replica counts for these two policies are 1.04 and 1.09.

Reactive

The storage utilization and replica count for the reactive policies are provided below in Figure 87. Lacking any foresight and with no additional replicas to engage in repair, the distribution of storage usage for the 1-replica reactive policy is very like the 1-replica simple policy. During the populate phase, the storage consumption increases linearly at 0.45 petabytes per year, peaking at 2.6 petabytes at 5.58 years. The storage utilization in the post-populate phase follows an exponential decay rate -0.0232 per year, finishing around 0.25 petabytes.

Starting with the 2-replica reactive policy, the post-populate distributions shift to a nearly flat linear slope. The 2 and 3-replica policies still have their storage curves rise linearly as before in the populate phase; however, with the ability to replace lost replicas the rates are higher with a 1.05 and 1.58 petabyte increase per year. These consumption rates are almost exactly what would be expected for creating 2 and 3 replicas of the 0.53 petabytes being annually seeded into the simulation. In the post-populate phase, the two policies' storage utilizations hover close to the expected usage targets consistent with full replication status. The 2-replica reactive policy fluctuates between 5.963 and 5.994 petabytes, and the 3-replica policy between 8.941 and 8.998 petabytes. Both policies also exhibit a very slight upward trend of 248 and 444 gigabytes per year despite suffering data loss. This would appear to be caused by sampling effects and disk remap events that occur every five years. There is also the possibility that replicas migrate over time to the more reliable vintages through natural selection, i.e. replicas on less reliable disks get replaced and eventually migrate to more reliable drives. Fewer replica absences at data collection would give the appearance of rising storage usage.

The reactive policies maintain their respective replica counts for the surviving data objects with only minor variation due to interim losses between repair operations, 1.99635 to 1.99998 replicas for the 2-replica policy and 2.99148 to 2.99995 replicas for the 3-replica policy. This information helps to

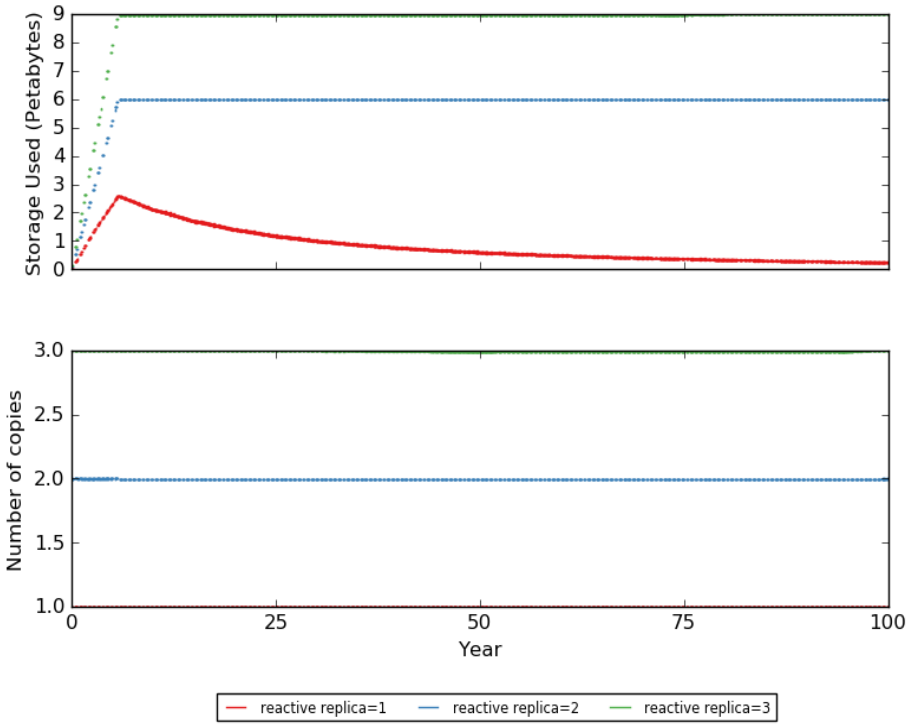


Figure 87. Storage and copies used over time for the reactive policies.

verify the behavior of the reactive policy to ensure a fair comparison with the proactive policy devised for this failure model.

Vintage Aware Proactive

The following is an analysis of the vintage aware proactive policies with an eye to how the storage utilization and counts relate to the reactive policies. The next two figures illustrate the storage usage and replica count for the vintage aware proactive policies. To improve readability, the eight proactive policies are partitioned based on their minimum required replication level. The four 1-replica policies are in the first figure, Figure 88, and the four 2-replica policies are in the second, Figure 89.

The 1-replica, 0.99-reliability policy's storage utilization is similar to the simple and 1-replica policies. When the data objects are being seeded, the storage utilization exhibits a linear rise of 0.505 petabytes annually, reaching a maximum consumption of 2.87 petabytes at 5.67 years. In the post-populate phase, the storage utilization suffers a slight exponential decay of -0.014 per year. While this policy does not have the reliability of the 2-replica reactive, it is an improvement upon all three simple policies and the 1-replica reactive while needing only one replica per data object for the duration of the simulations.

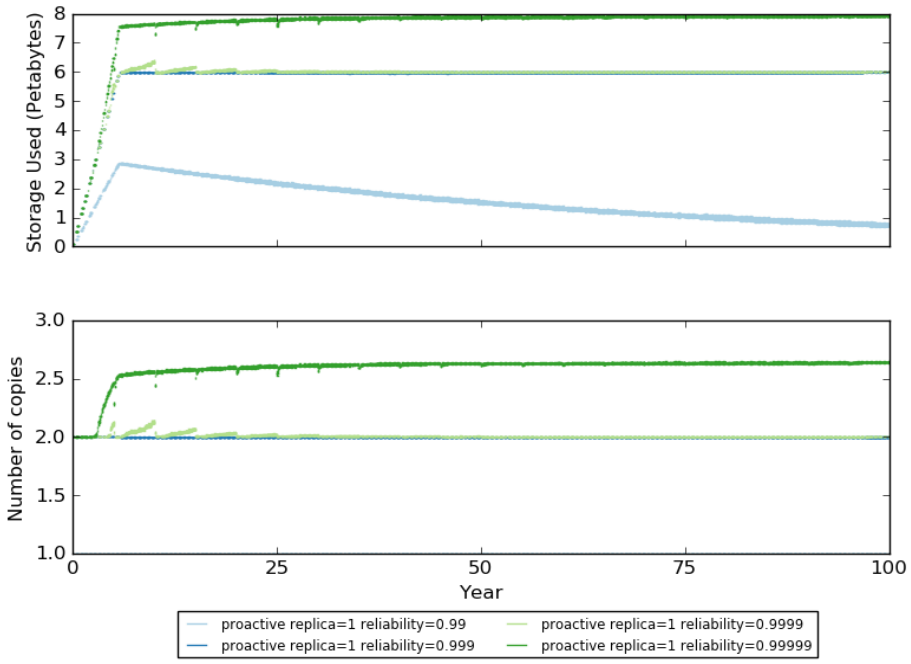


Figure 88. Storage and copies used over time for the 1-replica proactive policies on the vintage model.

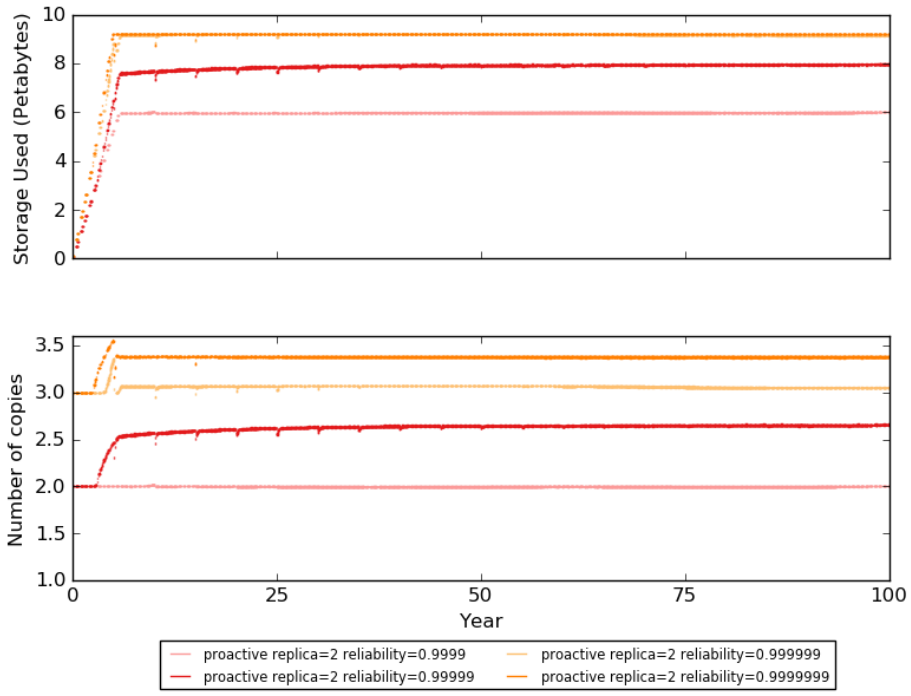


Figure 89. Storage and copies used over time for the 2-replica proactive policies on the vintage model.

Beginning with the 1-replica, 0.999-reliability policy, none of the remaining policies ever drop far below needing to maintain a mean replica count of 2. The 1-replica, 0.999-reliability policy fluctuates between 1.99755 and 1.99997 replicas with essentially a zero-slope change in replicas over the 100-year period. During the populate phase, the storage utilization climbs at 1.05 petabytes a year before transitioning to a gradual increase of 0.2 terabytes per year. The maximum storage utilization comes at 99.92 years with 5.996 petabytes. While not a blowout this policy does provide marginally better reliability for less storage than the 2-replica reactive.

The reliability versus storage utilization gets even more complicated for the final two 1-replica proactive policies. Both the 0.9999 and 0.99999-reliabiliy policies have greater reliability than the 2-replica reactive, but they also have storage peaks exceeding that of the 2-replica reactive policy. The 1-replica, 0.9999-reliability policy lowest mean replica observed is 1.99997, and the single greatest peak is 2.1284 replicas with 6.4 petabytes of storage used around the 9.9 year mark. For most the policy's run, the number of replicas is 2, but early on both the replica count and storage utilization curves feature a "saw-tooth" pattern of usage peaks. These peaks ebb with time as the policy adjusts the arrangement of the replicas into a more stable fit. By year 25, this pattern has completely abated into a flat line. If only this post period was considered, then clearly this policy does better than 2-replica reactive for the same replica count.

This issue is further exacerbated by the results of the 1-replica, 0.99999-reliability policy. This policy's reliability is nearly as good as the 3-replica reactive, but its replication needs range from 1.9998 to 2.6440 copies with around 2.6 being the norm. Looking at the storage usage and replica counts of this policy, there are two interesting quirks. The first is that before year 4, the policy was only requiring two replicas for every data object but then in years 4 and 5 at the end of the seeding phase for the simulation, the replica count jumps to 2.54 and eventually settles around 2.6. This behavior suggests that for the first 4 years despite shifting vintage reliabilities, there is enough availability on the better disks for the policy's data migration to require only two replicas. Then as the free storage tightens up with the arrival of more data objects, the policy has no choice but to seek additional permanent replicas to meet the reliability target. The second point of interest is the inverse "saw-tooth" where the mean replica counts drops periodically coinciding with the 5-year disk replacements. The fresh disks have better reliability that permits the policy to drop the replica count for small intervals. The policy's behavior suggests that the reliability setting is approaching the edge of the policy's capabilities for the storage provided.

In terms of storage utilization, the 1-replica, 0.9999-reliability policy has an increasing rate of 1.07 petabytes per year in the first phase and then with the exception of the "saw-tooth" gradually relinquishes storage at -797 gigabytes annually. It settles into the same pattern as the previous policy, the 1-replica, 0.999-reliability policy. After a steady rise of storage consumption of 1.33 petabytes per year, the 1-replica, 0.99999-reliability policy continues to consume

additional storage at an average rate of 2.60 terabytes yearly. Its maximum storage utilization comes around 99.92 years with 7.932 petabytes.

The first two 2-replica policies with 0.9999 and 0.99999-relability parameters are very like the corresponding 1-replica proactive policies in their results and the behavior of their curves. The 2-replica, 0.9999-reliability does not display the same “saw-tooth” pattern owing to early arriving data objects already mandatorily distributed to multiple disks. They were already blended among the different disk vintages as a result. The policy ranges between 1.9974 to 2.0176 replicas and has a maximum storage utilization of 6.02 petabytes at 9.92 years. These maximums are less than that found in the 1-replica policy due to the lack of the “saw-tooth.” The policy increase storage usage by 1.05 petabytes per year during the populate phase and then mostly flatlines with an increase of only 0.237 terabytes per year. While the storage rises, the replica count diminishes at -0.0000221 replicas per year.

The 2-replica, 0.99999-reliability policy is even more similar to its 1-replica variant, possessing almost the same exact values. Its storage consumption rises at the same rate of 1.33 petabytes annually and then settles out to 2.64 terabytes per year; the maximum value witnessed was 7.967 petabytes. A difference in the post-populate rate of just 0.04 terabytes per year and a maximum difference of 35 terabytes for the two policies. The replica count varies from 1.9993 to 2.6556 versus the 1.9998 to 2.644 seen in the 1-replica policy; in fact, ever the replica change rate matches between the policies. With both 2-replica proactive policies exhibiting such strong similarities with their predecessors, the same problem arises when attempting to rank them. They are more reliable than the 2-replica reactive but tend to require more storage.

The penultimate vintage aware proactive policy, the 2-replica, 0.999999-reliability policy, requires even more storage to achieve its reliability requirements, operating on the fringe of available storage space. The maximum mean storage usage observed comes at 32 years with 9.21 petabytes, but with a post-populate storage utilization that diminishes at -0.596 terabytes per year, it remains close to the limit throughout its lifetime. The number of replicas varies from 2.951 to 3.071 with the replica count being consistently 3.071 for the majority of the run with a nearly flat slope of -0.00009 replicas per year. It has a single replica peak between from year 4 to 5 that reaches 3.348 before a disk replacement cycle provides it enough flexibility to find a better storage arrangement for the data objects.

The 2-replica, 0.9999999-reliability policy has the distinction of being the only proactive policy studied that has less reliability despite having higher reliability settings. Examining its storage and replica curves provides insight into why this occurs. The storage utilization during the seeding phase rises at 1.80 petabytes per year which means that by the end of the phase, the policy would require 10.2 petabytes. A full petabyte more than is offered by the collective storage sites. Starting at year 3, the policy must start creating more replicas for the data objects to meet the immediate reliability needs resulting in a spike of

3.55 replicas. With the arrival of more data objects during this period, the policy is unable to meet the demands of all of the data objects having exhausted all available storage. Unable to remove replicas without compromising the reliability of existing data objects, arriving data objects are dropped inducing data loss. The disk refresh at year 5 provides a break for the policy with new disks raising the reliability enough to allow the policy to free some space and accept more data objects until the end of the seeding. During the post-populate phase, the mean number of replicas stays at 3.38 replicas with little to no fluctuations. The storage usage remains at 9.212 petabytes for the simulations with a small decline -0.137 terabytes per year caused by the several data objects that get lost in the subsequent 95 years.

Normalized Magnitude of Data Loss

In the last two subsections, drawing conclusions between the different configurations of the proactive and reactive policy has been difficult due to the amount of overlap and proximity of values for reliability and storage utilization. In order to clarify the results and enable the production of meaningful conclusions, this subsection employs the NOMDL metric based on the previous results. By plugging the values for the number of lost data objects and maximum replica counts into Equation 10 and Equation 11, the NOMDL for each policy is generated below in Table 34.

Table 34, in addition to the NOMDL, also provides a rundown on the theoretical and apparent storage usage and the size of the data loss experienced by each policy. The storage used is the total amount of storage that would be consumed by the policy had no data objects been lost, i.e. the maximum replica count multiplied by the total of all the data objects' size. The free storage is the amount of storage left for other purposes taken from the total storage of the system subtracted by the theoretical amount used. The MDL is the magnitude of total data loss, and D is the user apparent storage capacity of the system based on the total size of data and the remaining free storage available for other purposes.

Regardless of storage usage, all the NOMDL's for the proactive policies are better than 1-replica reactive and the simple policies. In fact, as the replication level rises, the values for the simple policy worsen significantly. The increase in replication out does the modest improvement in reliability for the simple policies. The reactive policies, however, improve tremendously with each additional replica despite the storage utilization rising by two and three times the base. The NOMDL for the 3-replica reactive is essentially 0 having lost only 1 data object on average. Agreeing with the findings in the reliability subsection, none of the other policies match the 3-replica reactive policy even with storage usage taken into consideration.

The one easy comparison previously was the improvement of 1-replica, 0.99-reliability policy over the 1-replica reactive policy. The NOMDL metric has

Table 34. NOMDL for vintage policies.

Configuration Replica	Reliability	Lost Data Objects	Replica Count	Storage Used (PB)	Storage Free (PB)	MDL (PB)	D (PB)	NOMDL
Simple								
1		2,746,600	1.00	3.00	6.22	2.75	9.22	0.298025
2		2,514,855	2.00	6.00	3.22	2.51	6.22	0.404578
3		2,302,021	3.00	9.00	0.216	2.30	3.22	0.715803
Reactive								
1		2,747,376	1.00	3.00	6.22	2.75	9.22	0.298109
2		2,170	2.00	6.00	3.22	0.00217	6.22	0.000349
3		1	3.00	9.00	0.216	.00000093	3.22	0.000000
Proactive								
1	0.99	2,257,287	1.00000	3.00	6.22	2.26	9.22	0.244931
1	0.999	1,945	1.99997	6.00	3.22	0.00194	6.22	0.000313
1	0.9999	1,936	2.12844	6.39	2.83	0.00194	5.83	0.000332
1	0.99999	323	2.64395	7.93	1.28	0.000323	4.28	0.000075
2	0.9999	1,640	2.01764	6.05	3.16	0.00164	6.16	0.000266
2	0.99999	274	2.65558	7.97	1.25	0.000274	4.25	0.000065
2	0.999999	89	3.34807	10.0	-0.828	0.0000891	2.17	0.000041
2	0.9999999	277,718	3.55217	10.7	-1.44	0.278	1.56	0.178084

the same ratio of improvement since both have the same exact storage requirements.

There is almost an order of magnitude difference between the reliabilities of the 1-replica, 0.99-reliability and the 2-replica, 0.9999999-reliability policies, but due to the difference in storage usage, their NOMDL rankings are only 1.38 times apart. The 1-replica, 0.999-reliability policy continues to barely edge out the 2-replica reactive policy under the NOMDL with the former's slightly greater reliability.

The first real performance results resolved by the NOMDL are the comparisons between the 1-replica, 0.9999 and 0.99999-reliability policies versus the 2-replica reactive. The two proactive policies utilize more storage but have better reliability outcomes. Running the numbers reveals that at least according to the NOMDL metric, the two proactive policies are 1.05 and 4.65 times better than the 2-replica reactive, respectively. Similar results are shown for the 2-replica, 0.9999 and 0.99999-reliability policies. Both policies have better reliability than their 1-replica counterpart policies and nearly matching storage requirements, so they outperform them and by extension the 2-replica reactive policy.

The 2-replica, 0.999999-reliability remains the best of the proactive policies, but its lead is diminished due to its higher replication needs. Its 3.1

times reliability lead over the next closest policy, the 2- replica, 0.99999-reliabiity, is reduced by nearly half to a 1.6 times lead for the NOMDL metric because of the jump in storage usage. An important point also arises for this policy in that if all data objects were to survive an additional 0.83 petabytes of storage would be required beyond what the system's configuration. This is made worse in the final policy, the 2-replica, 0.9999999-reliability, that would need at least another 1.44 petabytes, and it is unknown whether the replica count would even top out at this point.

Bandwidth Used

Simple

The bar graph in Figure 90 gives the traffic pattern for the simple policy configurations over the course of the simulations. The total traffic for each configuration is divided into its constituent types of bus-only, local area only and wide area data flow.

Beginning with the 1-replica simple policy, each policy uses three petabytes in the local area to move data from the source clients to depots co-located at the same site, producing the first replica. In the case of the 1-replica policy, this all that is required so this represents the entirety of the data flow it generates. In addition to this traffic, the 2-replica policy generates three

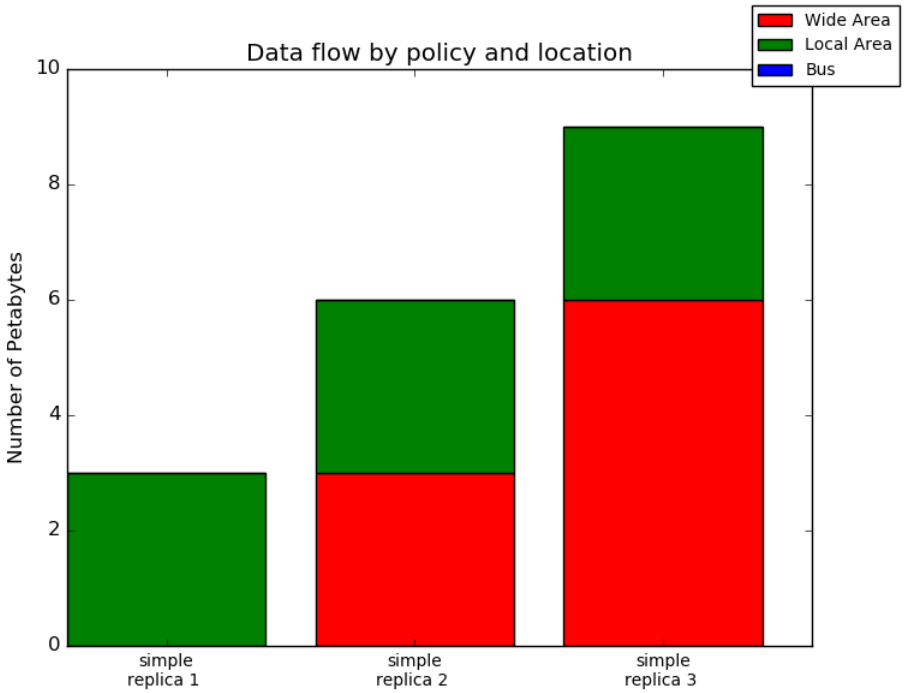


Figure 90. Data flow for the simple policies by category on the vintage model.

petabytes across the wide area to produce a second replica at another site using a total of 6 petabytes. In turn, the 3-replica policy reaches a total of nine petabytes having six petabytes in the wide area used to generate another replica.

For the 2 and 3-replica policies, the amount transferred exceeds the expected three and six petabytes by 2.56 and 4.87 gigabytes in the wide area due to occasional disk loss during the replica creation process and the policies responding with a new replication process. The highest overage witnessed was 11 gigabytes which would translate to 11 replication instances experiencing this effect.

Reactive

Except for the 1-replica reactive configuration, the reactive policy generates far more traffic than the simple policy, principally in the wide area. The 2 and 3-replica reactive policies have total usages at 31.40 and 52.38 petabytes, respectively. Figure 91 provides a breakdown of the reactive traffic flows by configuration and category of traffic.

Each of these policies utilize approximately three petabytes of local area traffic for the initial upload saving the wide area traffic for the additional required and recovery replications. The 3-replica reactive deviates from this slightly falling an average of 107.9 gigabytes short of the expected three petabytes. As the storage system was being utilized to near capacity with very little slack in storage at the end of the populate phase, the policy was forced to perform initial uploads to non-local depots at different sites than the originating data. There were also

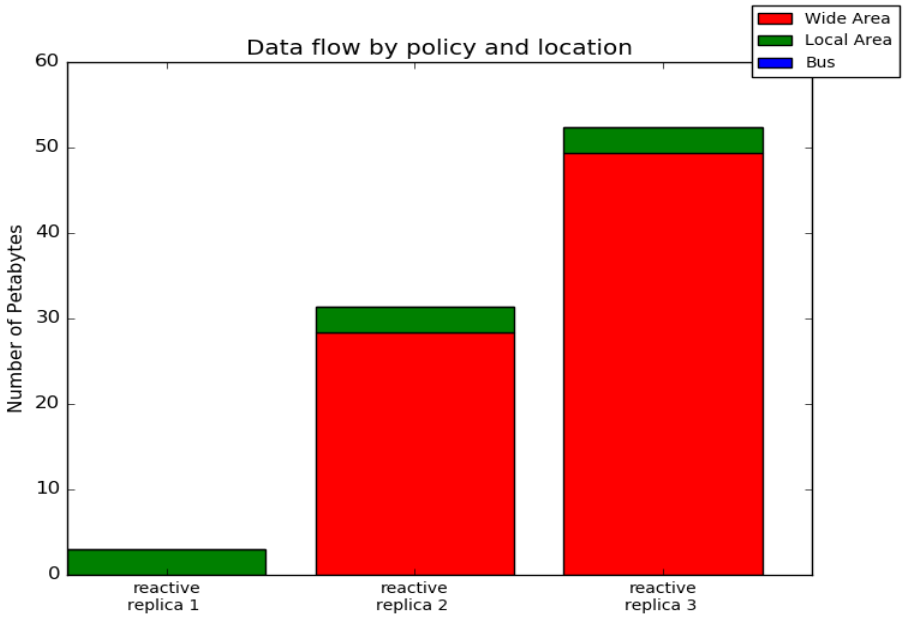


Figure 91. Data flow for the reactive policy on the vintage model.

simulation instances where this behavior did not occur, and all uploads were local. Like the corresponding simple policy, the three petabytes used by the 1-replica reactive policy for the initial data ingestion is all of the traffic generated. The two subsequent policies have a significant uptick in wide traffic with 28.40 and 49.38 petabytes. If all of the data objects achieve full initial replication then there are 25.40 and 46.38 million additional replications used to maintain the data objects over the 100-year timeframe.

There is no bus traffic exhibited by any of the reactive policies meaning all reactive repair operations occurred by pulling data objects over the wide area from other sites.

Vintage Aware Proactive

The results of the data flow analysis of the proactive policy are presented in Figure 92 along with the results for the 2 and 3-replica reactive policy for easy reference.

With the exception of the last two proactive policies, the 2-replica, 0.999999 and 0.9999999-reliability, The proactive policies generate less wide and local area traffic than the 2 and 3-replica reactive policies. However, all but the 1-replica, 0.99 and 0.999-reliability policies cause more overall total traffic with the inclusion of the bus-only traffic as the proactive policies shuffle data objects between disks on the same depot to maintain their reliability criteria.

The 1-replica, 0.99-reliability policy has the same amount of total traffic, three petabytes, as the 1-replica reactive policy and no bus-only traffic. This proactive policy confines 2.74 petabytes to the local area, but resorts to sending 0.25 petabytes across the wide area unlike the 1-replica reactive. This indicates that the policy configuration never performs any further replication or data migration for maintaining reliability but is more selective about initial data placement explaining why it outperforms the 1-replica reactive policy. It is sending local originating data objects to other sites if they have more reliable disks available.

The next policies, 1-replica, 0.999, 0.9999 and 0.99999-reliability, have nearly identically local and wide area traffic patterns of 1.698 and 1.307; 1.700 and 1.335; and 1.680 and 2.036 petabytes, respectively. Similar to the first proactive policy, these three policies are using most of the local and wide area traffic for the initial upload attempting to send new data objects to the most reliable of the remaining available disks. Each of these policies increasingly migrate and replicate data objects within the same depot to achieve the mandatory reliability. The mean replication level for these policies varies from nearly 2 to 2.64 and the spike in bus traffic indicates that all of the replicas for each data object are residing on different disks on the same depot. While the total traffic flows are as high or higher than the 2 and 3-replica reactive policies, nearly all of it is confined to the bus with 24.973 petabytes for the 1-replica, 0.999-reliability; 55.188 petabytes for the 1-replica, 0.999-reliability; and 77.801 petabytes for the 1-replica, 0.999-reliability policies.

The 2-replica, 0.9999-reliability policy has local and wide area traffic levels like the 2-reactive policy with 2.394 and 26.316 petabytes opposite of the 2-replica reactive policy's 3.0 and 28.4 petabytes. Although 15.859 petabytes of bus traffic are generated by the policy for replication and migration, the reliability setting is not high enough with two replicas to prevent a data object being lost at a site and requiring replacement. The combination of the two replicas is sufficient to meet the reliability criteria even if one of the replicas resides on a vulnerable disk. As a result, the policy frequently performs inter-site replications to maintain the two site replica invariant as indicated by the 26.316 petabytes of wide area traffic. The 2-replica, 0.99999-reliability policy improves on this with only 7.618 petabytes in wide area traffic, three petabytes of which is mandatory for the initial double site placement. Additionally, only 1.685 petabytes are created in the local area, so at least another 1.3 petabytes of the wide area traffic are used for the initial seeding. The reliability setting is now high enough to trigger reliability-driven migrations even with two replicas. The policy uses 85.326 petabytes of bus-only traffic to maintain the data objects on two sites.

The final two policies generate more wide area traffic than any of the other policies with 57.528 and 63.600 petabytes for the 2-replica, 0.999999 and 0.9999999-reliaiblity policies. For the 2-replica, 0.999999-reliaibilty policy, this is 2 and 1.2 times more than the 2 and 3-replica reactive policies and 2.2 and 1.3

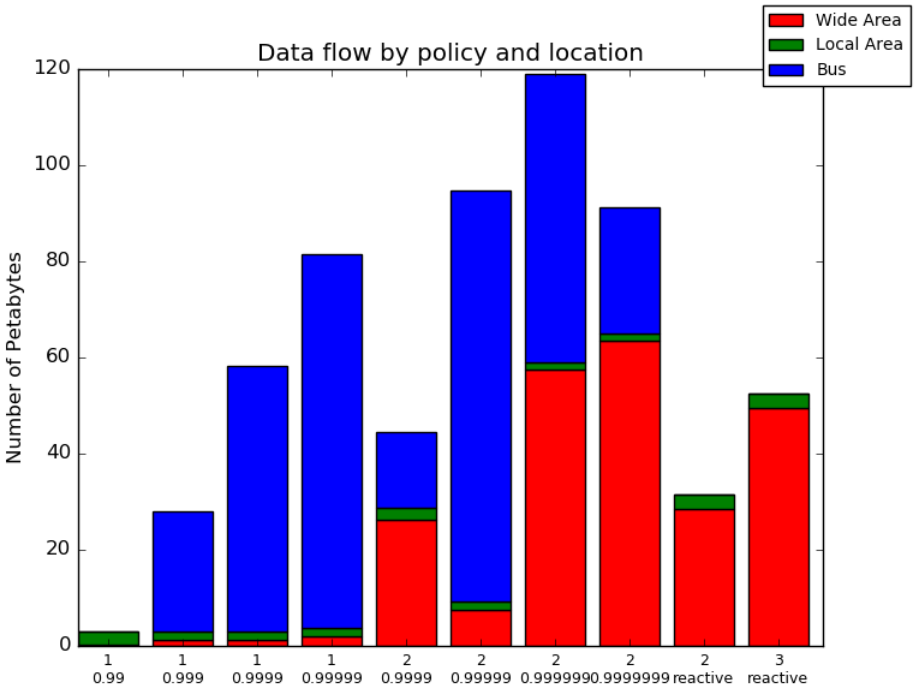


Figure 92. Data flow on the vintage model for the proactive policies by category with the 2 and 3-reactive policies for comparison.

times for the 2-replica, 0.9999999-reliability policy. Both policies also produce 59.892 and 26.129 petabytes of bus-only traffic but little local area traffic at 1.518 and 1.419 petabytes. The high utilization of both bus-only and wide area, while at the same time, low usage of the local area suggests that the policies are struggling to meet their reliability goals. When there is not a suitable disk available at the same depot, the policies are resorting to disks residing at other sites. It would appear that the policies have saturated most of the free space, and shifting reliabilities of the vintages are creating churn for replication and data migration. In total the 2-replica, 0.999999-reliability policy has the highest total traffic at 118.940 petabytes and the 2-replica, 0.9999999-reliability, the third highest at 91.147 petabytes. The total for the final proactive policy would probably have been higher had more data objects survived the initial populate phase.

CHAPTER FIVE

CONCLUSIONS AND RECOMMENDATIONS

This dissertation sought out to investigate if the survivability of data in Logistical Networking can be improved if naive replication policies are replaced with adaptive redundancy heuristics that predict storage failures and if these heuristics could enable Logistical Networking to optimize the usage of less healthy storage elements required to achieve reliability comparable to naive replication, thus reducing hardware and operational costs. The research utilized factors such as hard drive age, SMART errors, and vintage; to attempt to predict where near term disk failures were more likely to occur and proactively replicate at risk data before data loss would occur.

Having examined the literature, it was discovered that very little research had been done in this area for the wide area. Most of the research had concentrated on the use of SMART errors solely within the local area particularly large NAS's and data centers. In such situations the SMART errors are not used so much as a refined health or predictor metric, but just as an indicator of when to substitute them with fresh disks. Various disk management systems will pre-emptively migrate the data or jump to a "hot" spare disk and avoid utilizing the disk. There is no real refinement in the way the different error types are handled and reaction is uniform for all data objects affected by the disk change as a result of the use of RAID. Wide area storage and distribution systems appear to have settled on reactive replication, relying on replication and erasure encoding to safeguard the data, hoping that these schemes can provide enough time for repair or reconstruction before enough errors occur to induce data loss.

The lack of more sophisticated techniques for predicting and handling impending disk failures would appear to be caused by an overall lack of published data sets. Instead researchers have had to rely mostly on more suspect and limiting factors like MTTF numbers, accelerated life tests, and the supposed "bathtub" failure distribution. A wide range of factors have been speculated for years to affect the longevity of disk drives from utilization, temperature, etc. but very little concrete data to support and analyze their impact. In recent years, though, this has begun to change though the diligent research of various groups. Chief among these contributions has been the work and analysis done at Google on SMART errors by Eduardo Pinheiro, Wolf-Dietrich Weber and Luiz Andre´ Barroso; Disk population age and the CFDR by Bianca Schroeder and Garth Gibson and Jon Elerath's research on the importance of disk manufacture, model and vintage at NetApp. Their work enabled the research presented in this dissertation with the disk failure models based directly on their analyses.

To test the dissertation's hypothesis, a simulator was created for Logistical Networking that could simulate the various components of Logistical Networking including the network topology and storage media as well as the Logistical Networking software stack. The simulator was configurable for an array of

options including the failure behavior of the hard drives and the policy mechanism employed by LoDN for the distribution and maintenance of the data objects. The policy framework for data distribution was designed to be as flexible and extensible as possible with callbacks for handling different disk event types including status probes for information such as age and SMART error counters. For the topology and storage distribution, the REDDnet infrastructure served as the underlying model with the storage elements scaled up to 9+ petabytes. In an effort to be as accurate as possible, real life routing and time series bandwidth usage information was collected for as many of the network segments as possible and emulated as competing traffic flows.

From the research gathered, four different disk failure models were created for the purposes of simulation. The first two models were created from the analysis of SMART errors performed by Pinheiro et al. All of the drives fail according to the distribution curves provided by their research. Disk profiles are generated where every year a certain percentage will fail and of those some fail with and without preceding SMART errors. The disk profiles are populated with SMART errors that meet the dispersal pattern and count according to the lethality and false positives of the various SMART error types. The first SMART error failure model is based on the lethalities and survivorship of the SMART error types based on disk age at the time of the error. The second SMART error failure model used the lethalities and survivorship of the SMART error type cluster counts. The third failure model, the CFDR disk age model, was derived from the Bianca Schroeder and Garth Gibson's work on the HPC 1 datasets of the Computer Failure Data Repository using the monthly ARR of the disk population as a proxy for individual disk age failures. The fourth and final model, the Vintage model, is extrapolated from the six different vintages studied by Jon Elerath. By fitting his suggested Weibull distributions to the given data points, a probabilistic failure generator of 5 years for each vintage was created to simulate the disk failures.

Each of the four failure models were initially subjected to the same six policies consisting of 3 different replication levels, 1 to 3, for the simple and reactive policies. These two policies were designed to emulate the repair strategy observed in Logistical Networking and other wide area replication services. The simple policy handles creating an initial storage allocation based on proximity to the data source client. After the client finishes the initial upload to the allocation, the policy handles the creation and distribution of the data object replicas to the other sites. Once the desired replication and site dispersion level has been reached, the policy performs no further actions on behalf of the data object allowing the data object to decay over time. The reactive policy builds on the simple policy by performing replica repairs when disks fail. When the policy becomes aware of a disk failure, the policy evaluates the replication and site localities for all the affected data objects. If the replication level is too low for any data object, the policy will attempt to create new replicas and make every effort to maintain the multi-site invariant. When unable to do so, the policy will fall back

to attempting to create a replica at one of the already occupied sites starting at an unoccupied depot and then finally resorting to any unoccupied disk.

The proactive policies extend the aforementioned policies by utilizing different disk metrics to estimate the health and reliability of each replica and data object and then adjusting the replication and distribution of a data object's replicas accordingly. The principle aim being to better estimate where near-term failure is likely to occur and proactively move and create replicas before failure occurs lessening the chance of data object loss. The secondary goal is to decrease the required number of replicas needed for a given level of reliability over that required by the reactive policies. Finally, to reduce the amount of wide area traffic by performing data migration and replication within the same site where data is most likely to be lost before requiring repair involving different sites. When a disk event occurs, the policy re-evaluates the reliability of disk and any affected data objects. When the reliability of a data object is too low, the policy will respond by attempting to boost replication on the same depot and then the same site before resorting to inter-site replication. If the reliability is too high, then the policy will scan for any replicas that can be removed without violating the multisite replication invariant.

There are three different implementations of the proactive policies for the four failure models. The two SMART error failure models are both covered by the same SMART error-aware policy. In addition to the specified minimum replication level, the policy monitors the health of a data object using the summation of the reliabilities of each disk where the health of the disk is crudely calculated as the inverse power of 2 raised to the number of SMART errors witnessed in the last nine months. The CFDR disk age aware policy handles the CFDR failure model. For every month of a disk's life, the policy re-evaluates the reliability of the disk and the data objects adjusting the replication for the reliability level while always maintaining a minimum replication level. The Vintage aware policy computes the 2-month conditional Weibull reliability of each disk based on the disk's current age and vintage using the data analyzed and produced by Elerath. It maintains a given minimum replica count and reliability per data object where the reliability is the parallel aggregate of the disk reliabilities that the object resides on.

After performing a plethora of simulations and analyzing the results, this research demonstrated a definite improvement in reliability and storage savings provided by the disk aware, proactive policies for the first three failure models over traditional reactive techniques. For each of these failures models; SMART Error age, SMART Error count and CFDR disk age; there was a significant impact on data object resiliency under the proactive policies. For the SMART Error age failure model, the proactive policy was always more reliability than the reactive policy for the same requested replica level. The 2-replica proactive policy configurations suffered 63 to 200 times less data loss than the 2-replica reactive policy with the data loss dropping drastically as the health metric increased. Except for the 2-replica, 1.0-health proactive policy, the 2-replica

policy are also more reliable than even the 3-replica reactive policy. Examining the replica and storage utilization reveals that two of the 2-replica proactive policies require only two replicas; the 2-replica, 2.0-health policy needs 2.13 to 2.2 replicas. For the same amount of storage as the 2-replica reactive, the 2-replica, 1.0 and 1.5-health policies are more reliable having no greater storage usage than the reactive policy. Furthermore, both the 2-replica 1.5 and 2.0-health policies are more reliable and use less storage than the 3-replica reactive. In the case of the 1.5-health policy, this means saving a full replica and in the 2.0-health, 0.8 replicas per data object. Despite the 2-replica, 2.0-health policy's greater storage needs, factoring in its reliability, still, nets it a 188 times better NOMDL score than that of the 2-replica reactive. The 2-replica proactive policies are better than the 3-replica reactive policy according to the NOMDL metric even though the first 2-replica proactive policy scores worse in terms of reliability.

The SMART error count failure model shows similar results under the same proactive policy. For each replication level, this proactive policy is more resilient than the reactive policy. The 2-replica proactive policy configurations outperform the 2-replica proactive with 59 to 200 times less data loss. Like before, all of the 2-replica proactive policies except the 1.0-health are more reliable than the 3-replica reactive. Storage usage is slightly higher for the 1-replica proactive than the 1-replica reactive with the proactive policy using 1.16 replicas per data object. The 2-replica, 1.0 and 1.5-health return to form, though, with both using only two replicas. The final proactive policy; the 2-replica, 2.0-health; has need of more than the two replica minimum maintaining 2.23 replicas. Despite the increase in storage utilization, the increased reliability was enough to ensure that its NOMDL score exceeded the 3-replica reactive's by 3.2 times. While the 2-replica, 1.0-health policy configuration is not as reliability as the 3-replica reactive, it had a better NOMDL score due to its storage savings. Unfortunately, this trend does not remain true for the 1-replica, 1.0-health. The NOMDL score for this policy is slightly worse than 1-replica reactive even with its higher reliability. The rest of the proactive policies continued to have rising NOMDL values as their health metric rose.

The CFDR disk age proactive policy is also more resilient and efficient than the reactive policies although its reliability to storage ratio performance is not as strong as the two preceding SMART error models. Part of the issue lies in some of the proactive policies not having a steady storage utilization in the post-populate phase with some severe peaks early on. The "saw-tooth" pattern for these policies diminish with time, and statistics for both the peak and mean replica counts are used for comparison. The mean replica count is computed from the replica-seconds integral for the lifetime of the policies.

All of the proactive policies do as well as or better than the reactive policies with the same minimum replica count. The first proactive policy, the 1-replica, 0.9-reliability, does as well as the 1-replica reactive with the same storage usage. The subsequent 1-replica proactive policies do better but at higher peak replica counts of 1.6, 1.9997, 2.51 and 2.99994 as the reliability

metric rises. Examining the mean replica counts, though, reveals that the 1-replica, 0.99-reliability proactive policy has the same mean storage usage but with better reliability than the 1-replica reactive. All of the 2-replica proactive policies outperform the reliability the 2-replica reactive by 63 to 250,000 times, and in the case of the 2-replica, 0.999-reliability policy at the same maximum replica count. The 2-replica, 0.9999-reliability policy also does better than the reactive policy at the same lifetime average replica count of 2.0. The 3-replica, 0.99999-reliability policy does 1,300 times better than just the 3-replica reactive with a mean loss of 0.289855 data objects per simulation.

Continuing to compare the CFDR proactive policy configurations revealed that they tend to outperform the reactive policies at lower replication levels especially when using the mean lifetime replica counts. The 1-replica, 0.9999 and 0.99999-replica policies are more reliable than the 2-replica reactive by 58 and 156 times. The peak replica counts for these two policies are significantly higher than the 2-replica reactive, but their mean replica counts are only 1.70 and 1.74. Additionally, the 1-replica, 0.99999-replica is consistently better than even the 3-replica reactive policy for the first 95 years and may only lose out in the end due to a statistical fluke given the small sample size. While the 2-replica, 0.99999-reliability policy has several severe outliers that cause its mean reliability to be worse than the 3-replica reactive, both its median score and box plots place it as being nearly twice as reliable. The peak utilization is slightly less than three replicas with a lifetime mean replica count of just 2.14.

The NOMDL score improves the results of the CFDR proactive policy especially when adjusted to use the average replica counts instead of the maximum values. For instance, under the NOMDL all of the 1-replica proactive configurations except for the 0.99-reliability surpass 1-replica reactive and once adjusted even the 0.99-reliability does as well. The 1-replica, 0.9999 and 0.99999-reliability outdo the 2-replica reactive by 43 and 80 times which climbs further to 66 and 187 times once adjusted. The 1-replica, 0.99999-reliability overtakes the 3-replica reactive policy by a factor of 2 with the mean lifetime replica count. All of the 2-replica proactive policies are better than their 2-replica reactive counterpart by 62 times. This margin increases to between 62 to 112 times for the mean replica values. The adjusted NOMDL for the 2-replica, 0.99999-reliability achieves 1.2 times the score of the 3-replica reactive policy, and the regular NOMDL 3-replica 0.99999-reliability is a factor of 1,333 improved.

Unfortunately, the Manufacturer, model and vintage failure model did not demonstrate as positive a response with its proactive policy as the previous models. There are fewer instances of lower replicated proactive policies besting higher replicated reactive policies, though the proactive policy configurations did exceed similarly replicated reactive policies. The 1-replica, 0.99-reliability policy is more resilient than the 1-replica reactive policy while still only consistently requiring one replica. The 2-replica, 0.9999-reliability policy is improved over the 2-replica with 0.76 times fewer data objects lost with nearly the same replication

level. In terms of proactive policies with lower replication level meeting or exceeding higher replicated reactive policies, the 1-replica, 0.999-reliability policy has better reliability than the 2-replica (89% of the data loss suffered) with slightly less replication levels. The 1-replica, 0.99999-reliability configuration is competitive with the 3-replica reactive requiring only 2.6 replicas. The 2-replica, 0.9999999-reliability configuration is less reliable than lower specified reliability configurations. It exhausts all available storage by needing 3.5 replicas per data object preventing the policy from being able to service all of the data objects as a result.

Examining the NOMDL scores improves the comparison between the proactive policy and the reactive policy somewhat, but still, none of the proactive policies can compete with the 3-replica reactive even with the storage to reliability ratio taken into consideration. The 1-replica, 0.99-reliability and 0.999-reliability policies continue to surpass the 1-replica reactive and the 2-replica reactive policies, respectively. Under the NOMDL, the 1-replica, 0.9999 and 0.99999-reliability policies are better than the 2-replica despite needing more storage by factors of 1.05 and 4.65 times. All of the 2-replica proactive policies now surpass the 2-replica reactive policy.

The proactive policies tended to reduce the amount of wide area traffic in comparison to the reactive policies by enabling the use of local intra-depot replication and data migration when the health and reliability of a data object dropped below specified levels. The reactive policies very infrequently exploited bus traffic except when storage was nearing exhaustion. Instead, the reactive policies would have to wait until a replica was lost before issuing a replacement requiring inter-site replication to maintain the multi-site invariant. The proactive policies would typically have greater total traffic flow than the 2 and 3-replica reactive policies due to increased bus-only traffic generated by the intra-depot data migrations. The advantages of not going over the wide area are greater transfer reliability, greater transfer bandwidth, less latency and not having to rely on the behavior of other sites or competing with other individuals' traffic. Being able to have greater bandwidth and lower latency also feeds back to improve reliability by increasing the likelihood of successful replication before data loss.

The 2-replica proactive policies for the two SMART error models policies generate 1.0 to 1.2 times the amount of total traffic as the 2-replica reactive, but they conserve up to 25 petabytes in comparison to the 3-replica reactive (20% less traffic). This is significant because even though they are more reliable than the 3-replica reactive policy, they end up producing both less total and wide area traffic (as low as 40% of the 3-replica reactive policy's wide area traffic). As their health metric increases and the bus-only traffic rises in response to handle the data migration, the wide area traffic drops.

The CFDR disk age policy generates large quantities of bus-only traffic; in fact, several of them create in the order of 100's of petabytes of bus-only traffic. Despite this, though, their local-only and wide area traffic patterns are largely the same as the equal replica count reactive policy. The better protection the data

objects are afforded by this policy comes solely at the cost of increased bus-only traffic. The 2-replica, 0.99999-reliability policy only generates 40 to 80% of the wide area traffic as the 3-replica reactive policy but provides slightly better more reliability. Traffic analysis also reveals why the first two 2-replica CFDR proactive policies did not perform more reliably. Their specified reliability is too low to be tripped with the number of replicas that they are required to keep. Thus, their behavior is more like the reactive policies having to wait for failure before acting.

For the Manufacturer, Model and Vintage policy, all of the proactive policies except the last two proactive policies, 2-replica, 0.999999 and 0.9999999-reliability, generate less total, wide and local area traffic than the 2 and 3-replica reactive policies; albeit with an upsurge in bus traffic. For instance, the 1-replica, 0.999 and 0.9999-reliability policies are more reliable than the 2-replica reactive policy, but only transmit 3.0 to 3.7 petabytes across the local and wide area, contrast that to the 31.4 petabytes transmitted by the reactive policy. The best traffic for the 2-replica proactive policy is the 2-replica, 0.99999-policy with only 9.3 petabytes worth of non-bus-only traffic flows and far more reliability. As a general rule, as the specified reliability increased for a given replication level so did the amount of the bus-only traffic.

The work presented in this dissertation is unique and of vital importance to the future of not just Logistical Networking but all wide area replication services. Having completed a survey of all the methodologies and techniques employed for replication services it would appear that no one had studied this before. Several researchers have mentioned the possibility and the desire; in fact in [39], the authors directly acknowledge this, but the lack of disk reliability information and resources have so far stifled any such work.

Increasingly there will be a need for greater redundancy as data and storage capacity outpace disk reliability and IO bandwidth driving up repair time. Already local area storage systems and NAS's are transitioning from 1 to 2 level redundancy to 3 and above due to the long repair times. This problem is only exacerbated further in the wide area with higher latencies and lower available bandwidth. Additionally, all the management and transient errors in hardware and software only serve to further demonstrate the importance of proactive techniques. This dissertation has shown the feasibility of using proactive disk metric aware policies to curb increasing redundancy. At the same time, these policies have also been demonstrated to drastically decrease the amount of wide area traffic in most cases.

The results obtained from this research should be directly applicable in the real world. The simulator is detailed and accurate enough that Logistical Networking would experience real benefits from implementing a proactive policy framework similar to this one. While the research and results of this dissertation were tailored to Logistical Networking, because the underlying IBP protocol is very generic and simple, it should not be too onerous to port it to other services. The limiting factor is mostly having accurate failure metrics for the disk populations in question.

This dissertation and the underlying research for it lays out a solid, fundamental groundwork for further studies in the future. While this body of work aimed to be as comprehensive as possible on the subject, there are several avenues of potential research and resources that were not available at the outset that might warrant further investigation.

The most important of these is obtaining more real world disk metrics from a variety of sources. The statistics used for the failure models had to be reconstructed from data tables and figures presented in published work. If the original datasets could be obtained and analyzed, better and more accurate failure models and policies could be devised. The best source of disk metrics would probably come from vendors, but as mentioned by others, they are typically hesitant to share such information for fear of jeopardizing market position. Recently though more potential sources have become available. For instance, since 2015 BackBlaze [86], a cloud-based storage company, has started providing not just failure analyses based on factors like SMART errors, age, manufacturer and model, but have also published the raw underlying disk datasets for independent analysis. G.F. Hughes has made another SMART error dataset available at [87] for review.

This research concentrated solely on the use of SMART errors, disk age and vintage, but there are more metrics worth consideration such as utilization and temperature. While such metrics might be difficult to quantify initially, they could enhance reliability ratings. Further refinement and improvement could also come from utilizing multiple metrics simultaneously, but this would require obtaining, at a minimum, datasets from the published work already used or new datasets assuming that the correct information is available. Somewhat related to vintage, is the phenomena of correlated failure where disks of the same vintage have the tendency to experience similar failure patterns. While not strictly predictive in nature, proactive policies could be created to avoid distributing replicas on the same vintage of disks of the same age. This could alleviate problems suffered in other replication and erasure encoding schemes.

Hard drives were the only storage devices studied due to them being the principle storage medium in Logistical Networking and most other wide area storage systems and the lack of failure characteristics for other storage mediums. However, recently information as become available for newer storage technologies like flash based SSDs. The results of the first large-scale, multi-year study on SSDs was just published using data on storage servers employed by Facebook that examined attributes such as utilization, reads, writes, erase level-wearing behavior, temperature, bus power and platform [12]. Also during a preliminary presentation at Oak Ridge National Laboratory of the results for SMART errors policy, there was significant interest shown in using Logistical Networking in a hard drive and magnetic tape hybrid archival system with the ability to be proactive for both disk and tapes. There was the possibility of obtaining failure metrics for magnetic tape as well as hard drives used in ORNL's archival system.

The simulator so far relied upon full replication to boost redundancy due to the inherent latency and increased bandwidth required by erasure encoding and raid. In the future, though, it might be worthwhile to experiment with using erasure encoding methods instead of naive copying in a limited fashion. A possible scenario would be using replication for fast initial distribution and repair, then slowly replacing replicas over time with erasure-encoded blocks in the background. Under this scheme a small number of complete replicas would be kept for quick access and recovery, and then further redundancy would be added from the coding blocks. Another possibility would be confining erasure encoding at a per site level. When additional replicas are needed at a particular site, instead of generating more replicas, boost the local replica count with coding blocks. Both schemes would allow for fast repair but offer the storage savings of erasure encoding. If the simulator is enhanced with erasure encoding, then estimated processor utilization might be another resource metric to track and attempt to minimize since coding is a processor intensive process.

In addition to collecting more metrics for hard drives and other storage mediums, there are a number of improvements that could enhance the existing policies as well. For instance, instead of relying upon the Haversine distance equation to determine proximity between sites, the policies could utilize a network weather forecast system to determine where best to move and shuffle data. As seen with the 2 and 3-replica proactive CFDR disk age policies, a minimum global reliability specification many not always be enough to lower traffic over the wide area. In such cases, the policies may need to be extended to include per site and even per depot reliability sensitivity to force replication over the bus and local area network when local data is endangered.

Thus, far the policies only compared up to 3 replicas, the next wave of research might consider going to 4 replicas as +3 redundancy is becoming more common.

There are enhancements to the individual policy implementations that could be of value. For the SMART error policy, the health metric could be better defined to capture the near-term reliability of a data object and differentiate between the various SMART error types. Instead of first-fit deletion of replicas to make more storage available, implementing a best-fit replica deletion algorithm that provides enough reliability for the requesting data object but leaves the other data objects as health as possible. For the CFDR policy, some of the traffic and reliability issues could be addressed by having an algorithm that periodically rebalances data objects to maximize the system-wide health of all the data objects. Over time, a wide assortment of reliabilities was seen between the data objects that if better managed would improve the reliability of the population. Also, maintaining a pool of quick storage in the event of sudden need would lessen the response time of the policy to adjust to shifts in reliability by not having to wait on deletion operations to free space. The vintage policy could be enhanced by finding an optimal conditional reliability time period instead of the assumed two months. The period of two months was chosen based on

calculations of the reasonable amount of time needed to shuffle data objects without generating too much thrashing. In the case of the last vintage policy configuration, data objects were dropped during the populate phase because there was no storage space available and no storage could be freed due to the required reliability of each data object. Allowing the policy to voluntarily degrade reliability for short periods to free space would have prevented that from occurring and may have enabled the policy to best the performance of the 3-replica reactive policy.

LIST OF REFERENCES

[1] D. Reinsel, C. Chute, W. Schlichting, and J. McArthur, “The Expanding Digital Universe,” *White paper*, 2007.
[2] J. Gantz, “The Diverse and Exploding Digital Universe.”
[3] J. Gantz, *The Digital Universe Decade, Are You Ready?* IDC, 2010.
[4] A. Leventhal, “Triple-parity RAID and beyond,” *Commun. ACM*, pp. 58–63, Jan. 2010.
[5] M. Beck and T. Moore, “Logistical networking: a global storage network,” *J. Phys.: Conf. Ser.*, vol. 16, no. 1, pp. 531–535, 2005.
[6] *REDDnet Map*. http://www.reddnet.org/mwiki/images/1/12/Reddnetmap.gif.
[7] C. Walter, “Kryder's law.,” *Sci Am*, vol. 293, no. 2, pp. 32–33, Aug. 2005.
[8] M. H. Kryder and Chang Soo Kim, “After Hard Drives—What Comes Next?,” *IEEE Trans. Magn.*, vol. 45, no. 10, pp. 3406–3413, 2009.
[9] W. Jiang, C. Hu, Y. Zhou, and A. Kanevsky, “Are disks the dominant contributor for storage failures?,” *Trans. Storage*, vol. 4, no. 3, pp. 1–25, Nov. 2008.
[10] B. Schroeder and G. A. Gibson, “Disk failures in the real world: What does an MTTF of 1, 000, 000 hours mean to you?,” *FAST*, 2007.
[11] K. V. Vishwanath and N. Nagappan, “Characterizing cloud computing hardware reliability,” presented at the the 1st ACM symposium, New York, New York, USA, 2010, pp. 193–204.
[12] J. Meza, Q. Wu, S. Kumar, and O. Mutlu, “A Large-Scale Study of Flash Memory Failures in the Field,” presented at the SIGMETRICS '15: Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, New York, New York, USA, 2015, pp. 177–190.
[13] D. A. Patterson, G. Gibson, and R. H. Katz, “A case for redundant arrays of inexpensive disks (RAID),” presented at the the 1988 ACM SIGMOD international conference, New York, New York, USA, 1988, pp. 109–116.
[14] I. S. Reed and G. Solomon, “Polynomial codes over certain finite fields,” *Journal of the society for industrial and applied …*, 1960.
[15] J. Blömer, M. Kalfane, R. Karp, M. Karpinski, and M. Luby, “An XOR-based erasure-resilient coding scheme,” 1995.
[16] P. Corbett, B. English, A. Goel, and T. Grcanac, “Row-diagonal parity for double disk failure correction,” *FAST-2004: 3rd Usenix …*, 2004.
[17] M. Blaum, J. Brady, J. Bruck, and Jai Menon, “EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures,” *IEEE Trans. Comput.*, vol. 44, no. 2, pp. 192–202, 1995.
[18] P. Maniatis, M. Roussopoulos, T. J. Giuli, D. S. H. Rosenthal, and M. Baker, “The LOCKSS peer-to-peer digital preservation system,” *ACM Transactions on Computer Systems (TOCS)*, vol. 23, no. 1, pp. 2–50, Feb. 2005.
[19] A. Rowstron, P. Druschel, and A. Rowstron, *Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility*, vol.

35, no. 5. ACM, 2001, pp. 188–201.
[20] A. Rowstron and P. Druschel, “Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems,” in *Research and Advanced Technology for Digital Libraries*, vol. 2218, no. 18, Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 329–350.
[21] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, “Wide-area cooperative storage with CFS,” presented at the SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles, 2001, vol. 35, no. 5.
[22] D. M. Geels, “Data Replication in OceanStore,” Dec. 2002.
[23] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, “Tapestry: A Resilient Global-Scale Overlay for Service Deployment,” *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 41–53, Jan. 2004.
[24] M. W. Storer, K. M. Greenan, E. L. Miller, and K. Voruganti, “POTSHARDS—a secure, recoverable, long-term archival storage system,” *ACM Transactions on Storage (TOS)*, vol. 5, no. 2, Jun. 2009.
[25] J. J. Wylie, M. W. Bigrigg, J. D. Strunk, G. R. Ganger, H. Kiliccote, and P. K. Khosla, “Survivable information storage systems,” *Computer*, vol. 33, no. 8, pp. 61–+, Aug. 2000.
[26] M. Selimi and F. Freitag, “Tahoe-LAFS Distributed Storage Service in Community Network Clouds,” presented at the 2014 IEEE International Conference on Big Data and Cloud Computing (BdCloud), 2014, pp. 17–24.
[27] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn, *Ceph: a scalable, high-performance distributed file system*. USENIX Association, 2006, pp. 307–320.
[28] S. A. Weil, S. A. Brandt, E. L. Miller, and C. Maltzahn, “CRUSH: controlled, scalable, decentralized placement of replicated data,” presented at the SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing, New York, New York, USA, 2006, p. 122.
[29] A. Haeberlen, A. Mislove, and P. Druschel, “Glacier: highly durable, decentralized storage despite massive correlated failures,” presented at the NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2, 2005, vol. 2, pp. 143–158.
[30] E. Sit, A. Haeberlen, F. Dabek, and B. Chun, “Proactive replication for data durability,” 2006.
[31] R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G. M. Voelker, “Total Recall: System Support for Automated Availability Management.,” 2004.
[32] Z. Yang, J. Tian, B. Y. Zhao, W. Chen, and Y. Dai, “Protector: A Probabilistic Failure Detector for Cost-Effective Peer-to-Peer Storage,” *IEEE Trans. Parallel Distrib. Syst.*, Nov. 2010.
[33] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R.

Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer, “Farsite: federated, available, and reliable storage for an incompletely trusted environment,” *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 1–14, Dec. 2002.

[34] F. Junqueira, R. Bhagwan, A. Hevia, K. Marzullo, and G. M. Voelker, “Surviving internet catastrophes,” presented at the ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference, 2005.

[35] F. Junqueira, R. Bhagwan, K. Marzullo, S. Savage, and G. M. Voelker, “The phoenix recovery system: rebuilding from the ashes of an internet catastrophe,” presented at the HOTOS'03: Proceedings of the 9th conference on Hot Topics in Operating Systems - Volume 9, 2003, vol. 9, pp. 13–13.

[36] S. P. Marketing, “Get S.M.A.R.T. for Reliability,” Jul. 1999.

[37] G. F. Hughes, J. F. Murray, K. Kreutz-Delgado, and C. Elkan, “Improved disk-drive failure warnings,” vol. 51, no. 3, pp. 350–357, 2002.

[38] J. Murray and G. Hughes, “Hard drive failure prediction using non-parametric statistical methods,” presented at the Proceedings of ICANN/ …, 2003.

[39] E. Pinheiro, E. Pinheiro, W. D. Weber, W.-D. Weber, L. A. Barroso, and L. A. Barroso, “Failure Trends in a Large Disk Drive Population.,” *FAST*, pp. 17–28, 2007.

[40] G. Hamerly, “Bayesian approaches to failure prediction for disk drives,” 2001.

[41] V. Agarwal, C. Bhattacharyya, T. Niranjan, and S. Susarla, “Discovering Rules from Disk Events for Predicting Hard Drive Failures,” presented at the 2009 International Conference on Machine Learning and Applications (ICMLA), 2009, pp. 782–786.

[42] W. Featherstun, “Using syslog message sequences for predicting disk failures,” *… of the 24th international conference on Large …*, 2010.

[43] B. Zhu, G. Wang, X. Liu, D. Hu, S. Lin, and J. Ma, “Proactive drive failure prediction for large scale storage systems,” presented at the 2013 IEEE 29th Symposium on Mass Storage Systems and Technologies (MSST), 2013, pp. 1–5.

[44] Y. Zhao, X. Liu, S. Gan, and W. Zheng, “Predicting Disk Failures with HMM- and HSMM-Based Approaches,” in *Research and Advanced Technology for Digital Libraries*, vol. 6171, no. 30, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 390–404.

[45] J. Li, X. Ji, Y. Jia, B. Zhu, G. Wang, Z. Li, and X. Liu, “Hard Drive Failure Prediction Using Classification and Regression Trees,” presented at the 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2014, pp. 383–394.

[46] W. Yang, D. Hu, Y. Liu, S. Wang, and T. Jiang, “Hard Drive Failure Prediction Using Big Data,” presented at the 2015 IEEE 34th Symposium

on Reliable Distributed Systems Workshop (SRDSW), 2015, pp. 13–18.
[47] T. Pitakrat, A. van Hoorn, and L. Grunske, “A comparison of machine learning algorithms for proactive hard disk drive failure detection,” presented at the the 4th international ACM Sigsoft symposium, New York, New York, USA, 2013, pp. 1–10.
[48] B. D. Strom, SungChang Lee, G. W. Tyndall, and A. Khurshudov, “Hard Disk Drive Reliability Modeling and Failure Prediction,” *IEEE Trans. Magn.*, vol. 43, no. 9, pp. 3676–3684, 2007.
[49] W. Tepin and Y. Kidjaidure, “Customer Failure Modes prediction for Hard Disk Drive using Neural Networks Rank-Level Fusion,” presented at the 2011 8th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON 2011), pp. 476–479.
[50] T. Suchatpong and K. Bhumkittipich, “Hard Disk Drive failure mode prediction based on industrial standard using decision tree learning,” presented at the 2014 11th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2014, pp. 1–4.
[51] J. Yang and Feng-Bin Sun, “A comprehensive review of hard-disk drive reliability,” presented at the Annual Reliability and Maintainability. Symposium. 1999 Proceedings (Cat. No.99CH36283), 1999, pp. 403–409.
[52] J. G. Elerath and Sandeep Shah, “Annual Symposium Reliability and Maintainability, 2004 - RAMS,” presented at the Annual Symposium Reliability and Maintainability, 2004 - RAMS, 2004, pp. 151–156.
[53] Sandeep Shah and J. G. Elerath, “Disk drive vintage and its effect on reliability,” presented at the Annual Symposium Reliability and Maintainability, 2004 - RAMS, 2004, pp. 163–167.
[54] “Reliability analysis of disk drive failure mechanisms,” pp. 226–231, Jan. 2027.
[55] J. G. Elerath and S. Shah, “Disk drive reliability case study: dependence upon head fly-height and quantity of heads,” *Reliability and Maintainability Symposium, 2003. Annual*, pp. 608–612, 2003.
[56] J. Elerath, “Hard-disk drives: the good, the bad, and the ugly,” *Commun. ACM*, vol. 52, no. 6, p. 38, Jun. 2009.
[57] K. M. Greenan, J. S. Plank, and J. J. Wylie, “Mean time to meaningless: MTTDL, Markov models, and storage system reliability,” presented at the HotStorage'10: Proceedings of the 2nd USENIX conference on Hot topics in storage and file systems, 2010.
[58] J. F. Paris, T. Schwarz, and D. Long, “When MTTDLs are not good enough: Providing better estimates of disk array reliability,” *Proc 7th International …*, 2008.
[59] B. Eckart, X. Chen, X. He, and S. L. Scott, “2008 IEEE International Symposium on Modeling, Analysis and Simulation of Computers and

Telecommunication Systems," presented at the Telecommunication Systems (MASCOTS), 2008.
[60] I. Koren and C. M. Krishna, *Fault-Tolerant Systems*. Morgan Kaufmann, 2010.
[61] B. K. Choi and D. Kang, *Modeling and Simulation of Discrete Event Systems*. 2013.
[62] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, 2nd ed. Chichester, UK: John Wiley and Sons Ltd., 2002.
[63] B. Schroeder and G. A. Gibson, "The computer failure data repository (CFDR)," *Workshop on Reliability Analysis of System Failure Data (RAF'07), MSR Cambridge, UK*, 2007.
[64] UNIPHIZ Lab Software, "FindGraph."
[65] M. Xie and C. D. Lai, "Reliability analysis using an additive Weibull model with bathtub-shaped failure rate function," *Reliability Engineering & System Safety*, vol. 52, no. 1, pp. 87–93, 1996.
[66] MATLAB, "MATLAB." The MathWorks Inc., Natick, Massachusetts.
[67] "noc.net.internet2.edu," *noc.net.internet2.edu*. [Online]. Available: https://noc.net.internet2.edu. [Accessed: 18-Nov-2016].
[68] "configs.html," *vn.grnoc.iu.edu*. [Online]. Available: http://vn.grnoc.iu.edu/Internet2/configs/configs.html. [Accessed: 25-Aug-2016].
[69] "cricket.nlr.net," *cricket.nlr.net*. [Online]. Available: http://cricket.nlr.net. [Accessed: 25-Aug-2016].
[70] "cricket.cenic.org," *cricket.cenic.org*. [Online]. Available: http://cricket.cenic.org/. [Accessed: 25-Aug-2016].
[71] "intermapper.engineering.cenic.org," *intermapper.engineering.cenic.org*. [Online]. Available: https://intermapper.engineering.cenic.org/. [Accessed: 25-Aug-2016].
[72] "repository.uslhcnet.org," *repository.uslhcnet.org*. [Online]. Available: http://repository.uslhcnet.org/. [Accessed: 25-Aug-2016].
[73] "futuregrid," *snapp2.bldc.grnoc.iu.edu*. [Online]. Available: http://snapp2.bldc.grnoc.iu.edu/futuregrid. [Accessed: 25-Aug-2016].
[74] "pacificwave.net," *pacificwave.net*. [Online]. Available: http://pacificwave.net/. [Accessed: 25-Aug-2016].
[75] "network-infrastructurearchitecture," *sox.net*. [Online]. Available: http://www.sox.net/wp/services/network-infrastructurearchitecture/. [Accessed: 25-Aug-2016].
[76] "www.gigapop.gen.tx.us," *gigapop.gen.tx.us*. [Online]. Available: http://www.gigapop.gen.tx.us/. [Accessed: 25-Aug-2016].
[77] "routers2.cgi," *wan.tamu.edu*. [Online]. Available: http://wan.tamu.edu/cgi-bin/routers2.cgi. [Accessed: 25-Aug-2016].
[78] "graph.php?local_graph_id=4981&rra_id=all," *stats.dfw.tx-learn.net*. [Online]. Available: https://stats.dfw.tx-

learn.net/graph.php?local_graph_id=4981&rra_id=all. [Accessed: 25-Aug-2016].
[79] “nms.tx-learn.net,” *nms.tx-learn.net*. [Online]. Available: http://nms.tx-learn.net/. [Accessed: 25-Aug-2016].
[80] “www.flrnet.org,” *flrnet.org*. [Online]. Available: http://www.flrnet.org/. [Accessed: 25-Aug-2016].
[81] “tengigabitethernet3_1?view=octets64;ranges=m%3Ay,” *noc.ucsb.edu*. [Online]. Available: http://noc.ucsb.edu/cricket/cisco/border/r1/tengigabitethernet3_1?view=octets64;ranges=m%3Ay. [Accessed: 25-Aug-2016].
[82] “nms.ns.ufl.edu,” *nms.ns.ufl.edu*. [Online]. Available: https://nms.ns.ufl.edu. [Accessed: 25-Aug-2016].
[83] E. Jones, T. Oliphant, and P. Peterson, “SciPy: Open source scientific tools for Python, 2001.” URL http://www. scipy. org, 2015.
[84] J. D. Hunter, “Matplotlib: A 2D graphics environment,” *Computing In Science \& Engineering*, vol. 9, no. 3. IEEE COMPUTER SOC, pp. 90–95, 2007.
[85] “XLWT,” *python-excel.org*.
[86] “Hard Drive Test Data - Determining Failure Rates and More,” *backblaze.com*. [Online]. Available: https://www.backblaze.com/b2/hard-drive-test-data.html. [Accessed: 11-Sep-2016].
[87] G. F. Hughes, “S.M.A.R.T. Dataset.” 01-Feb-2013.

VITA

Christopher D. Brumgard was born on September 5, 1979. He attended the University of Tennessee at Knoxville to obtain a BS in 2003, an M.S. in 2005 and a Ph.D. in 2016. He was a graduate student under the direction of Dr. Micah Beck at LoCI since 2005 and has worked in several HPC environments including Oak Ridge National Laboratory under the XSEDE and DataOne projects. From 2010 to 2012, he was a researcher and developer at ACCRE at Vanderbilt University. He has collaborated with large-scale, data-intensive organizations like the LHC project and AmericaView.

Upon completion of his Ph.D., he is scheduled to start work as postdoctoral researcher at Oak Ridge National Laboratory.