A Practical Approach to Reconciling Availability, Performance, and Capacity in Provisioning Extreme-scale Storage Systems

Lipeng Wan* Feiyi Wang† Sarp Oral† Devesh Tiwari† Sudharshan S. Vazhkudai† Qing Cao*

*University of Tennessee, Knoxville
{lwan1, cao}@utk.edu

†Oak Ridge National Laboratory
{fwang2, oralhs, tiwari, vazhkudaiss}@ornl.gov

ABSTRACT

The increasing data demands from high-performance computing applications significantly accelerate the capacity, capability and reliability requirements of storage systems. As systems scale, component failures and repair times increase, significantly impacting data availability. A wide array of decision points must be balanced in designing such systems.

We propose a systematic approach that balances and optimizes both initial and continuous spare provisioning based on a detailed investigation of the anatomy and field failure data analysis of extreme-scale storage systems. We consider the component failure characteristics and its cost and impact at the system level simultaneously. We build a tool to evaluate different provisioning schemes, and the results demonstrate that our optimized provisioning can reduce the duration of data unavailability by as much as 52% under a fixed budget. We also observe that non-disk components have much higher failure rates than disks, and warrant careful considerations in the overall provisioning process.

1. INTRODUCTION

Provisioning an extreme-scale storage system needs to factor in a variety of goals such as capacity, performance and availability, while adhering to a fixed price point. While each individual target is important, what is even more critical is how they are reconciled with each other towards a practical solution. Some examples of extreme-scale deployments include the Oak Ridge Leadership Computing Facility's (OLCF) Spider I and II storage systems, Livermore Computing Center's Sequoia storage system [2] and Riken Advanced Institute for Computational Science's K-Computer storage system [25]. Let us consider OLCF's Spider I and II Lustre-based parallel file systems, which were among the world's fastest Lustre storage systems at the time of their deployments, and intended for the Jaguar (No. 1 on the June 2010 Top500 list [31]) and Titan (No. 2 on the current Top500 list [31]) supercomputers, respectively. Spider I offered 10 PB of capacity, using 13,440 1 TB drives, organized in RAID 6 arrays, delivering 240 GB/s; Spider II offers 40 PB of capacity, using 20,160 2 TB drives, using RAID 6, delivering 1 TB/s aggregate throughput. It is seldom the case that such configurations are readily arrived at by either the customer or the vendor. In fact, storage system designers negotiate such terms with vendors during the procurement process, based on broad targets derived from user requirements and a desire to provide a certain quality of service to users. Provisioning is therefore a complex reconciliation process between often competing goals.

SC '15, November 15-20, 2015, Austin, TX, USA
© 2015 ACM. ISBN 978-1-4503-3723-6/15/11...$15.00
DOI: http://dx.doi.org/10.1145/2807591.2807615

Large-scale storage systems are often built with *scalable system units* (SSU). These basic building blocks are replicated to meet the required capacity, performance and availability targets. Oftentimes, during the initial provisioning stage, system designers need to optimize the configuration under a fixed budget. These configuration options include (but not limited to) optimizing the number of SSUs for capacity and performance; selecting a drive type and/or size while balancing capacity, performance and rebuild times; eliminating single points of failures, and increasing system redundancy.

From a day-to-day operations view point, the focus shifts to maintaining system health and minimizing data unavailability and loss incidents. One effective way to increase both system and data availability is to reduce the time spent in repairing and replacing failed components. This can be achieved by having an on-site spare parts pool. However, uniformly provisioning spare parts for all hardware components is neither cost effective nor practical. Therefore, a critical concern in extreme-scale storage system deployment and management is designing and implementing an effective spare provisioning policy.

These are but a few design and operational constraints to be considered. However, system designers are left with back of the envelope calculations and rules of thumb when it comes to such provisioning decisions. There are no models, simulations or tools that designers can use to plug in parameters, and answer such *what-if* scenarios.

Contributions: In this paper, we address these provisioning challenges that storage system designers face by building tools that can help reconcile key figures of merit, and answer *what-if* scenarios. Our study and the tools are primarily intended for storage system architects, administrators and procurement teams, for their provisioning needs. We present a quantitative analysis on provisioning a large-scale HPC storage system. Our results shed light on the following: the components that affect cost, performance and capacity; the components that are most important in building a performance-oriented storage system; and the effect redundancy among components has on the reliability of the overall system and its cost-effectiveness. In particular, we show that the I/O controller and disk enclosures play a more dominant role than disk drives in building a high-performance, cost-effective HPC storage system. We also show that it is more cost-efficient to saturate the I/O controllers of one SSU before scaling out. Scaling up SSUs without saturating the controllers may potentially save disk drive costs, but increases the overall cost significantly while decreasing the system reliability. In order to ensure a highly available operational experience, we also build a dynamic optimization model for continuous spare provisioning. Our results demonstrate that the model is not only able to minimize the number of unavailability events, but also reduce the window of data unavailability.

2. THE GENERAL APPROACH

The design and procurement of extreme-scale storage systems are complex in nature. When faced with multi-faceted considerations, system designers usually cope with the challenges by adopting an ad hoc process that is a combination of back of the envelope calculations and the reliance on past experiences. The end result may *make sense*, but they are difficult to reason, with little or no quantifiable justification. In this paper, we take a more systematic approach by focusing on three key issues in designing such a system, namely availability, capability (performance) and capacity, under a fixed cost constraint. In particular, we divide the provisioning process into two phases, namely **initial provisioning** and **continuous provisioning**. Our operational experience suggests that designing for the second phase should receive equal, if not more attention since the shelf life of an extreme-scale storage system tends to be five years or even longer. As can be seen in Table 1, the two phases also place different emphasis regarding the aforementioned key characteristics. Note that in both cases, we consider the total cost of ownership to be fixed.

	Performance	Capacity	Availability	Spare Parts
Initial Provisioning	✓	✓	✓	Fixed
Continuous Provisioning	Fixed	Fixed	✓	✓

Table 1: Provisioning approaches. Check marks indicate the performance metrics that will be optimized in each phase, while "fixed" indicates the constraints during each phase.

The provisioning of the initial system deployment is primarily based on the understanding of the trade-offs among cost, performance and capacity. While cost remains the primary constraint, it is not the case that simply buying faster disks will yield the best performance for a given budget. This is because of the complex building structures of an HPC storage system, as well as how different components affect the performance, cost and reliability of the whole system. We quantify these trade-offs in Section 4.

Since the component characteristics in storage systems change over time, achieving high data availability requires continuous provisioning and deployment. For instance, when an extreme-scale storage system is initially deployed, all components are new, but as time goes by, some components fail and get replaced, which changes their performance or reliability characteristics. If spare parts have been provisioned, and readily available before the failure, the replacement and repair of these faulty parts could be completed quickly, significantly reducing the possibility of data unavailability. Moreover, as failed components are replaced by spare parts, the system contains both new and aging components. Thus, the reliability status of the system during operations is different from the one at the time of initial deployment. Therefore, the spare provisioning policies for continuous operations should also be different.

For both initial and continuous provisioning, a series of *what-if* questions based on system capacity, capability and component reliability characteristics need to be answered. In the following sections, we will discuss three essential pieces to support our study in answering these questions. First, we present an anatomy of a large-scale storage system that serves both as a background and a review of current practice. Second, we present a thorough analysis of field failure data collected from a production storage system. Our analysis shows that the failure trend over time provides more insight than vendor-supplied reliability metrics. Third, we present a generic provisioning tool for storage systems, which is used as a foundation for studying and evaluating our initial and continuous provisioning models.

3. THE BUILDING BLOCKS

Building an effective provisioning mechanism requires a clear understanding of the past and present characteristics, as well as the expected future state of a given system. In this section, we describe the foundational building blocks that enable us to explore, model and experiment with the proposed *initial* and *continuous provisioning* methods. First, we dissect an extreme-scale storage system, provide a perspective on how large-scale storage systems are built from scratch, and discuss the lessons learned in the process. We then analyze a multi-year field failure dataset from an extreme-scale storage system. This dataset provides not only a detailed view on the storage system's reliability and availability characteristics, but also insights and parametric inputs to the follow-on provisioning tool and experimental design. Finally, we describe the design and implementation of the provisioning tool, and analyze how to leverage the tool to answer several provisioning questions in sections 4 and 5.

3.1 Anatomy of a Large-Scale Storage System

Extreme-scale storage systems are built using SSUs for ease of design, procurement, deployment, management and maintenance. An SSU consists of all required components to build a stand alone file system. In order to reach the design targets, multiple SSUs are acquired and deployed. SSU examples include block-level storage systems (e.g. DDN SFA series [5], IBM DS series [14], NetApp FAS series [18]) or file-system level appliances (e.g. Seagate ClusterStor9000 [29] or Panasas ActiveStor [19]).

As an example of an extreme-scale storage system, we present the architecture of OLCF's Spider I. The design targets and specifications of Spider I are well documented [30]. Spider I was deployed in 2008, and remained operational until 2013, serving the Jaguar supercomputer that was No. 1 on the Top500 list of machines in June 2010. At the time of deployment, Spider I was announced as the fastest and largest known Lustre parallel file system in the world. We use Spider I as a case study for our work as its field failure data is publicly available [33]. Spider I was built using 48 SSUs, each one consisted of a DDN S2A9900 controller couplet [4], with 280 1 TB SATA disks configured in 5 disk enclosures. Each couplet was connected to 4 file system servers. Spider I offered an aggregate system performance of 240 GB/s, and provided over 10 PB of RAID 6 formatted capacity, using 13,440 SATA disks and 192 file system servers. It served more than 26,000 file system clients from several clusters and the Jaguar supercomputer. Each Spider I DDN couplet was composed of two singlets. Host-side interfaces in each singlet was populated with two dual-port 4x DDR IB HCAs. The back-end disks were connected via ten SAS links on each singlet. For a SATA based system, these SAS links connected to expander modules within each disk shelf. The expanders then connected to SAS-to-SATA adapters on each drive. All components had redundant paths. Each singlet and disk tray had dual power-supplies where one power supply was powered by the house power and the other by the UPS. Figure 1 illustrates the internal architecture of a Spider I DDN S2A9900 couplet.

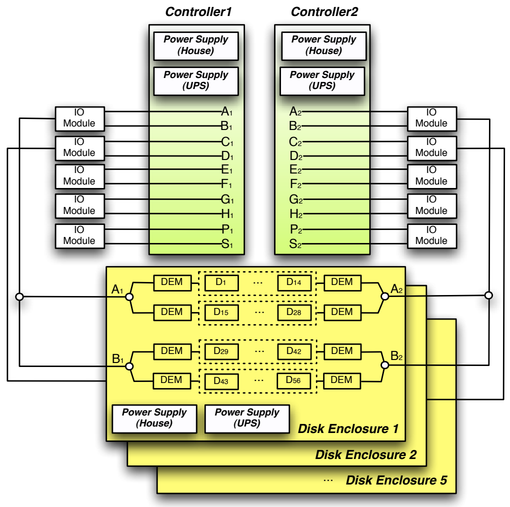


Figure 1: Spider I S2A 9900 architecture

3.2 Field-gathered Failure Data Analysis

In this section, we analyze the field gathered failure data of different hardware components used in the Spider I file system at OLCF, and provide an account of the key findings learned through the process.

3.2.1 *Vendor Provided Reliability Metrics*

System vendors often provide AFR (annual failure rate) or MTTF (mean time to failure) of each type of FRU (field replaceable unit). As stated earlier, Spider I consists of 48 SSUs, and the AFRs of the FRUs are listed in Table 2. Vendor provided reliability metrics can be used to derive a coarse-grained estimation of a storage subsystem's reliability. As an example, one model that has been widely used to estimate the data availability of disk redundancy groups is continuous Markov chain, which has an underlying assumption that the failure rates of disk drives are constant (time independent) [3, 10, 20, 27]. With such a model, the vendor-provided metrics, AFRs and MTTF, can be used to establish the failure model of each disk drive, which assumes that the time to failure of disk drives is an exponential distribution.

	Number	IDs	Unit Cost ($)	Vendor AFR	Actual AFR
Controller	2	15-16	10,000	4.64%	16.25%
House Power Supply (Controller)	2	1-2	2,000	0.83%	4.38%
Disk Enclosure	5	27-31	15,000	0.23%	1.17%
House Power Supply (Disk Enclosure)	5	3-7	2,000	0.08%	8.50%
UPS Power Supply	7	8-14	1,000	3.85%	NA
I/O Module	10	17-26	1,500	0.38%	0.92%
Disk Expansion Module (DEM)	40	32-71	500	0.23%	0.29%
Baseboard	20	72-91	800	0.23%	NA
Disk Drive	280	92-371	100	0.88%	0.39%

Table 2: FRUs in one scalable storage unit

3.2.2 *Field Failure Data*

Besides the vendor-provided metrics, system administrators typically maintain field-gathered failure and replacement data. Such information is much closer to the reality than vendor provided reliability metrics. In fact, by analyzing the field-gathered failure data of storage systems, several existing studies have shown that the failure rates of disk drives and other hardware components can vary over time [11, 22].

The failure and replacement data for Spider I was collected from all of the 48 SSUs during its 5-year operational period. The dataset contains timestamps when device replacement was needed. We first count the number of failures of each type of FRU during 5 years, and then calculate their actual AFRs. The results are summarized in Table 2. Below is a list of the key findings:

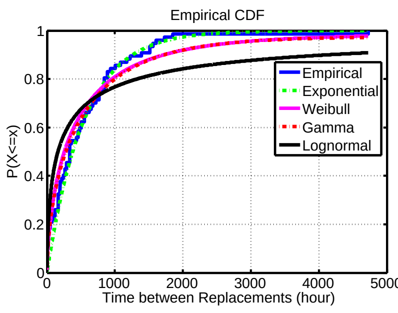
Finding 1. *The actual annual failure rate (AFR) of Spider I disks is only 0.39% – much smaller than what has been reported in previous studies [26]. It is hard to generalize this as the environment, testing conditions and vendors are quite different. Efficient facilities support, e.g., better power and cooling infrastructure, might be a factor here. However, it is not possible to quantitatively establish a causal relationship between operating conditions and disk drive failure rate.*

Finding 2. *Aggressive burn-out tests at the time of system deployment help eliminate potential problematic or slower disks early on, which improves the overall aggregate parallel performance. It also keeps the disk AFR low by removing potential problematic disks from the population.*

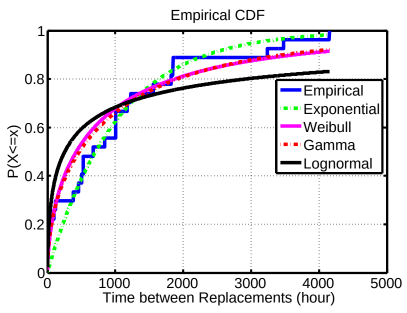
On the point of stress testing and slow disk identification, there are no community standards for this process. Our method involved individually stressing each SSU, and identifying the slowest disk RAID groups. Then, we exercised those groups separately, and collected latency statistics on the disks individually. This process should be performed during initial deployment, and repeated periodically to keep a healthy and uniformly performing disk population. Our records indicate that the AFR before the acceptance of the Spider I system was much higher (2.2%). Our early testing helped remove close to 200 slow or bad disks. This resulted in a much lower AFR during production (0.39%).

Finding 3. *Non-disk components of Spider I have higher AFRs than vendor provided metrics.*

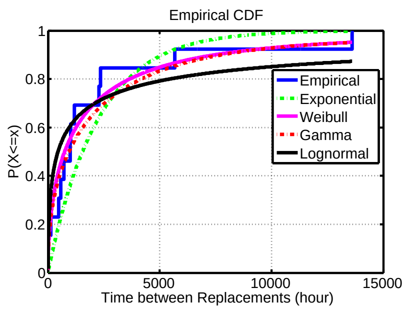
While this comes as a surprise, it also suggests that future studies should carefully model and account for the reliability of non-disk components as they contribute heavily towards the overall reliability of the system.



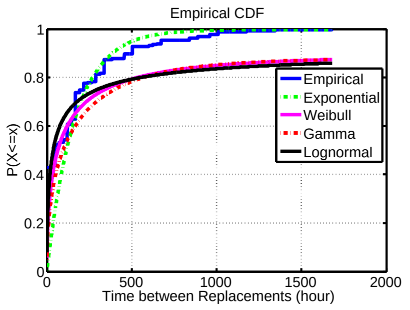
(a) Controllers



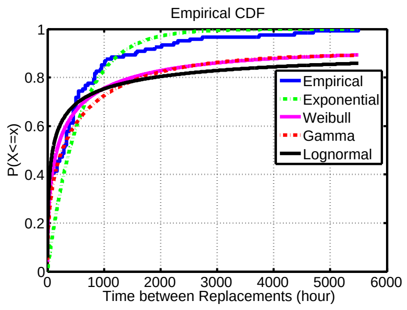
(b) Dems (Disk Expansion Modules)



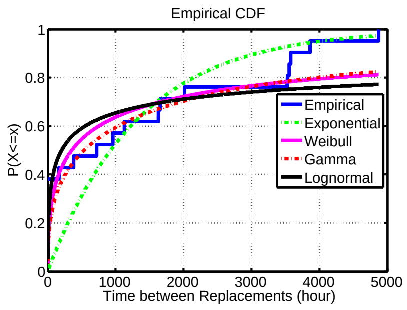
(c) Disk Enclosures



(d) Disk Drives



(e) House Power Supply



(f) I/O Modules

Figure 2: Distribution of time between device replacements for different types of FRUs in Spider I

Next, we derive the empirical, cumulative distribution function (CDF) of the time between device replacements for different types of FRUs (Figure 2). We use four distributions to fit the CDF (Figure 2).

For example, as shown in Figure 2(d), when the time between disk replacements is relatively small, a Weibull distribution with decreasing failure rate is a better fit; with increasing time between disk replacements, the failure rate is stable, and an exponential distribution is a better fit. This observation indicates that in reality the failure rate of disk drives could be neither constant nor monotonically increasing or decreasing, which differs from what is usually assumed by many existing studies [7, 8, 11, 28].

Finding 4. *Disk drive failures can be more accurately modeled by joining two different distributions.*

FRU replacement times for Spider I have not been recorded or shared with the public. However, it was stated that most of these replacements were completed within 24 hours, if spare parts were available on-site. If there were no spare parts on-site, a replacement was awaited, and usually took at least 7 days [21].

3.3 Storage System Provisioning Tool

Based on our understanding of the architecture of extreme-scale storage systems and failure data obtained from the field and vendors for Spider I, we have built a generic provisioning tool to study how different provisioning policies impact data availability.

3.3.1 Design Considerations

A large-scale storage system is often composed of thousands of FRUs, and the failure dependencies between them are complex. Specifically, one FRU's failure might have a

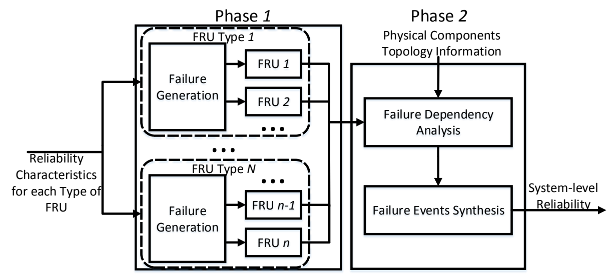


Figure 3: Framework of the provisioning tool

cascading effect on other FRUs, as there is often a correlation between the failures of closely-coupled hardware components in a storage system. For example, a disk enclosure's failure might lead to the unavailability of hundreds of disk drives. Inspired by a conventional diagrammatic method for modeling the reliability of complex systems called the reliability block diagram (RBD) [24], we build a provisioning tool, which can estimate not only the number of failures during a certain period of operation for each type of FRU, but also the system-level reliability by analyzing the failure propagation.

As shown in Figure 3, in phase 1, the failure events of each type of FRU are randomly generated based on the reliability characteristics, which are determined by vendor-provided metrics, historical failure data and the provisioning policies used. Thereafter, the failure events are randomly allocated to FRUs that belong to the same type, and logged throughout each FRU's life cycle. In phase 2, the framework extracts the failure dependencies from all FRUs, and builds an RBD based on the topology of the storage system. For example, the RBD of the SSU in Table 2 is shown in Figure 4, where each block is assigned a unique ID to represent

FRU's Type	Time between Failure		Time to Repair		Time to Repair (without spare part)	
	Distribution	Parameters	Distribution	Parameters	Distribution	Parameters
Controller	Exponential	rate = 0.0018289	Exponential	rate = 0.04167	Shifted exponential	rate = 0.04167, offset = 168
House Power Supply (Controller)	Weibull	shape = 0.2982, scale = 267.7910	Exponential	rate = 0.04167	Shifted exponential	rate = 0.04167, offset = 168
Disk Enclosure	Weibull	shape = 0.5328, scale = 1373.2	Exponential	rate = 0.04167	Shifted exponential	rate = 0.04167, offset = 168
House Power Supply (Disk Enclosure)	Exponential	rate = 0.0024351	Exponential	rate = 0.04167	Shifted exponential	rate = 0.04167, offset = 168
UPS Power Supply *	Exponential	rate = 0.001469	Exponential	rate = 0.04167	Shifted exponential	rate = 0.04167, offset = 168
I/O Module	Weibull	shape=0.3604, scale =523.8064	Exponential	rate = 0.04167	Shifted exponential	rate = 0.04167, offset = 168
Disk Expansion Module (DEM)	Exponential	rate = 0.000979	Exponential	rate = 0.04167	Shifted exponential	rate = 0.04167, offset = 168
Baseboard *	Exponential	rate = 0.000252	Exponential	rate = 0.04167	Shifted exponential	rate = 0.04167, offset = 168
Disk Drive	[0, 200], Weibull	shape = 0.4418, scale = 76.1288	Exponential	rate = 0.04167	Shifted exponential	rate = 0.04167, offset = 168
	[200, ∞], Exponential	rate = 0.006031				

*Field data missing, vendor-provided AFRs are used.

Table 3: Parameter settings of the provisioning tool

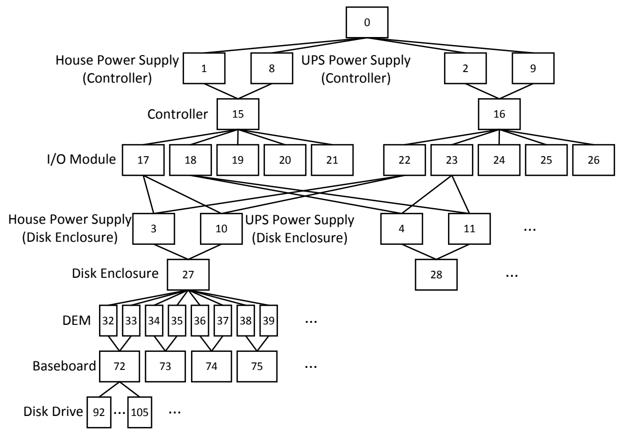


Figure 4: RBD of a scalable storage unit

an FRU (for the convenience of using graph algorithms, we create a dummy block that does not represent any real FRU as the root (block 0) of all blocks in the RBD). In the RBD, the reliability of each block depends on its parents, while determining that of its children. Given all such extracted failure dependencies, our tool synthesizes the results across all components, and provides detailed information on the estimates of the various metrics of interest, e.g., the average number of failed FRUs, events leading to data unavailability or data loss and for how long.

3.3.2 Implementation and Validation

As stated in Section 3.2, for each type of FRU, we fit the empirical data of the time between device replacements in Spider I to four different distributions. In order to choose the best parameter settings for the provisioning tool, we apply the Chi-squared test [12] to determine the probability distribution and corresponding parameters that are more realistic to generate the failure events. To generate the repair time, we use the exponential distribution with two different mean values, 24 hours for FRUs with a spare part and 168 hours (7 days) for those that do not. In certain cases, when the repair time is much longer, the provisioning tool will be even more critical to improve the overall performance, as an unoptimized provisioning strategy could lead to a much longer window of vulnerability and higher probability of data unavailability. The chosen distributions and parameters are listed in Table 3.

An interesting fact worth noting in Table 3 is the distribution parameter settings for generating disk drive failure events. As our analysis of disk failure data reveals (see Figure 2(d)), when the time between disk replacements is relatively small (less than 200 hours), a Weibull distribution with decreasing failure rate is a better fit; with the time between disk replacements increasing, the failure rate becomes stable, and an exponential distribution fits the empirical data better. Therefore, we use a method called inverse transform sampling [6] to generate disk failure events so that the time between failures fits a crafted distribution, which is actually a join of a Weibull distribution with decreasing failure rate and an exponential distribution with constant failure rate.

After a failure event of a specific FRU type is generated, it will be randomly allocated to an attribute device belonging to that FRU type in the system, and logged as a failure of that device. A random repair time will then be generated and logged for that device so that we can derive all its failure time intervals for the operational period. Once the failure logs of all devices for the operational period are obtained, the tool will synthesize them based on the RBD to derive the duration of temporary data unavailability and permanent data loss. For example, in the RBD shown in Figure 4, if the execution results indicate all parents of an FRU are down during a time period, the FRU is tagged as unavailable no matter what its own results are during the same time interval.

During the 5-year operation of Spider I, we only observed two data unavailability events. The lack of empirical data makes it difficult to validate our tool on system-level data availability. However, we can validate the results of each type of FRU using the field-gathered failure data. As listed in Table 4, we compare the number of failures of each type of FRU observed in the empirical data against the results from the provisioning tool during a 5-year period (we run the tool 10,000 times, and calculated the average number of failures). We can see that the results of our tool approximates to the empirical data, which demonstrates its accuracy.

Component Type	# of Total Units	Empirical # of Failures	Estimated # of Failures	Estimation Error
Controller	96	78	79	1.04%
House Power Supply (Controller)	96	21	27	6.25%
Disk Enclosure	240	14	20	2.5%
House Power Supply (Disk Enclosure)	240	102	105	1.25%
I/O Module	480	22	24	0.42%
Disk Expansion Module (DEM)	1920	28	42	0.73%
Disk Drive	13440	264	338	0.55%

Table 4: Validation on FRU failures estimation

4. INITIAL PROVISIONING

Provisioning an HPC storage system for initial deployment involves understanding the tradeoffs between performance, cost, capacity and reliability. Often times, system architects are provided with a fixed budget for an initial acquisition and deployment, with an emphasis on optimizing for performance and capacity. Reliability characteristics at the SSU-level or at the system-level are also factored in during this phase, with vendor support and spare part pools as the primary vehicles for maintaining system reliability. In this section, we attempt to reconcile these factors for an initial deployment, and study their interplay.

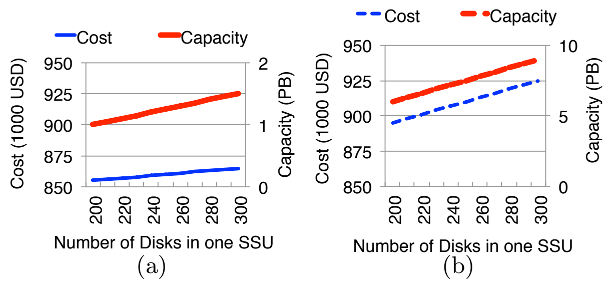


Figure 5: The cost and capacity trade-offs for 200 GB/s system-wide I/O bandwidth performance target. 1 TB drive (a) vs. 6 TB drive (b)

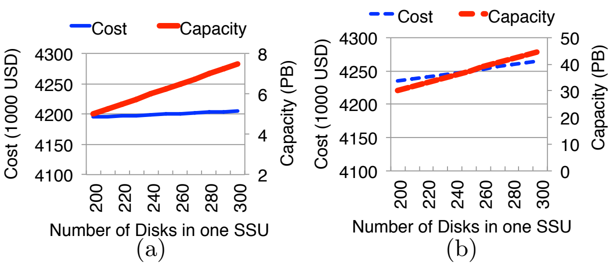


Figure 6: The cost and capacity trade-offs for 1 TB/s system-wide I/O bandwidth performance target. 1 TB drive (a) vs. 6 TB drive (b)

Optimizing for performance: Each SSU can achieve a theoretical peak performance that is primarily determined by the type of the I/O controller and the number of disks in each SSU. An SSU does not necessarily have to be 100% populated (in terms of the number of disks it can accommodate) in order to achieve its peak I/O performance. Therefore, the overall performance of a storage system, consisting of multiple SSUs, can be expressed as:

$$\text{Performance} = N_{SSU} * max(SSU_{Perf}, D_{SSU} * BW_{disk}), \quad (1)$$

where N_{SSU} is the number of SSUs in the system, SSU_{Perf} is the peak performance of one SSU, D_{SSU} is the number of disks in one SSU and BW_{disk} is the bandwidth achievable from one disk. Equation 1 can be optimized independently for sequential or random I/O workloads. However, the selected workload should reflect the design parameters of the storage system and represent the expected production environment.

The cost of the storage system is the sum of the cost of all components (as listed in Table 2, with their respective price points per unit). The capacity of the whole system can be expressed as:

$$\text{Capacity} = D_{SSU} * N_{SSU} \quad (2)$$

Impact of number of disks and disk capacity on the overall cost and performance: While investigating how these factor into building an HPC storage system, we concluded that disk prices do not have the first order impact on provisioning a cost-effective or high-performance storage system. This is mainly because disks constitute only 15-20% of the cost of one SSU. Therefore, when designing a storage system with performance as the primary objective, it is optimal to buy as many SSUs as possible before optimizing or negotiating for disk price or capacity. Once the number of SSUs are fixed (i.e., the peak achievable performance point is fixed), it remains unclear how the number of disks and the storage capacity per disk affect the cost and capacity of the overall system. To study that, next we present a case study where we set performance goals as 200 GB/s and 1 TB/s, and build a storage system with the SSU as characterized by Table 2 and Figure 1. Note that, our results assume specific parameters for disks and other components of the SSUs, but the same study can be carried out for other chosen parameters.

We assumed that each disk can provide 200 MB/s of bandwidth, therefore 200 such disks are enough to saturate one SSU (assuming a 40 GB/s peak I/O bandwidth per controller pair). Each SSU in our case accommodates up to 300 disks, therefore buying any disks beyond 200 is equivalent to buying more capacity. Also, filling an SSU with less than 200 disks (the number of disks that saturate our SSU) always resulted in lower performance per unit price. The underlying reason is that other components of an SSU significantly dominate the cost of the whole system compared to the disks. Therefore, we focus on how filling an SSU with 200 to 300 disks changes the cost and capacity of the system (Figure 5 and 6). We consider two types of disks (1TB and 6TB, with same I/O performance bandwidth but different costs: 100 and 300 USD, respectively). As expected, the relationship is linear in terms of performance and capacity. It is worth noting that the relative increase in the cost of the system is very modest when going from 200 to 300 disks. However, the difference between the choice of 1 TB disks vs 6 TB disks may itself bring cost differences of over $50K (Figure 6).

Finding 5. *The I/O controller and disk enclosures play a more dominant role than disk drives in building a high-performance, cost-effective storage system. It is more cost-efficient to saturate the I/O controllers of one SSU before scaling out if performance is the highest design priority.*

One may also note that there are availability and reliability issues involved in increasing the number of disks. However, we want to point out that the MTBF of an SSU is not affected significantly by increasing the number of disks from 200 to 300. Vendor-provided statistics listed in Table 2 as well as the field-collected failure data indicate that the peripheral components have lower MTBF than disks. Also, redundancy schemes (e.g. RAID 6) significantly decrease the probability of an interruption. However, there may be cases of multiple concurrent disk failures in the same RAID group, which may necessitate a rebuild process. In such a scenario, 1 TB disks are better than 6 TB disks as rebuilding is faster for the same amount of disk space that needs to be reconstructed. This is because the bandwidth does not change significantly across these disk types for a given family of disks. We believe this assumption will be valid for the near future. Of course, there are technologies that can improve the dynamics of disk redundancy or rebuild process. However, such new technologies are slow to penetrate the storage market. Parity declustering, as an example, substantially reduces the rebuild window by distributing data and redundancy stripes over a number of disks [13]. It was first proposed more than two decades ago, and today there are only two products in the HPC storage market that support the parity declustering feature. Based on our experience, we are not predicting a disruptive change to this dynamic.

Effect of increasing disks/SSU on availability and the provisioning budget: Next, we study how the increase in cost due to extra capacity affects the reliability of the system. If the increase in the number of disks also significantly decreases the spare provisioning budget, then perhaps even this moderate increase in cost is not acceptable. Using the provisioning tool (Section 3.3), we calculate the number of events when data becomes unavailable in a 1 TB/s system (25 SSUs) for a period of 5 years if no provisioning policy is applied. Based on the disk failure rate calculated from the failure data, we also estimate the potential cost of disk replacement for a 1 TB/s system during a 5-year period. As can be seen in Figure 7, the number of data unavailability events and disk replacement cost increase with the number of disks per SSU.

Finding 6. *Fixed initial provisioning can be optimal from cost efficiency and capacity perspectives. However, it alone is not sufficient for improving the reliability dynamics. A well-designed, continuous provisioning policy is needed to maintain the system's data availability requirement under a fixed provisioning budget. This is also true if we plan to increase the disks/SSU for extra capacity.*

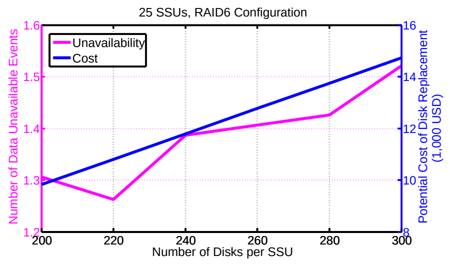


Figure 7: Number of data unavailable events and potential disk replacement cost for 1 TB/s systems (25 SSUs).

5. CONTINUOUS PROVISIONING

Ideally, if we have an unlimited budget for spare provisioning, we can provide unlimited spares for each component in the system. However, in reality the budget is always limited, and we can only provision a limited number of spares. Therefore, the goal of continuous provisioning policy is to explore such dynamics under constraints.

5.1 Ad Hoc Provisioning

To the best of our knowledge, most of the provisioning policies used in large-scale HPC storage systems are ad hoc, and are based on system administrators' intuition and experiences. We use Spider I to illustrate the ad hoc policies. As listed in Table 2, vendor-provided statistics for Spider I indicate that controllers have the highest failure rate among all FRUs, which can also be verified by the actual device replacement data. Thus, the first intuitive provisioning policy would be to provision as many controller spares as possible for a given provisioning budget. However, the controller-first provisioning policy does not improve the system data availability significantly when compared against not provisioning any budget for spares at all, as the two controllers in the same SSU are configured as a fail-over pair in the Spider I architecture. Only when both of them are down simultaneously does it lead to data unavailability, which is a rare event in practice.

The deficiency of the controller-first provisioning policy suggests that the built-in hardware redundancy might have more impact on data availability compared to the component failure rates. In other words, if the hardware redundancies are not well-designed, it could make the system more vulnerable to failures of some devices (also observed in Spider I). As shown in figures 1 and 4, the failure of a disk enclosure causes two disks in the same RAID group to become unavailable simultaneously. On the other hand, all the other FRUs combined will lead to at most one disk unavailability in each RAID group. This means that the storage system is more vulnerable to disk enclosure failures. Therefore, a more effective ad hoc provisioning policy for Spider I is to provide spares for disk enclosures first.

Finding 7. *The 5-disk enclosure architecture of Spider I was selected for minimizing the cost. However, this selection resulted in lower data availability. This was a lesson learned from the Spider I experience, and rectified in Spider II by switching to a 10-disk enclosure configuration.*

Based on the analysis of the system architecture and the redundancy characteristics of Spider I, we realized that most potential data unavailability scenarios could be caused by simultaneous failures of different types of FRUs (e.g., a disk enclosure failure coupled with a double power supply failure on another enclosure). If the budget is allocated for provisioning a specific type of FRU first, there might not be enough left to maintain spares for other types of FRU, which could negatively impact the data availability. Thus, neither controller-first nor enclosure-first provisioning policy is optimal. We will introduce our optimized dynamic spare provisioning model in the next section, and compare our model with the ad hoc provisioning policies in Section 5.3.

5.2 Dynamic Spare Provisioning Model

Our proposed model aims to optimize the spare provisioning policy for large-scale storage systems in order to achieve high data availability, given a limited provisioning budget.

N	Number of types of FRU in system
FRU_i	i-th type of FRU
$f_i(x)$	PDF of time between failures of FRU_i
$F_i(x)$	CDF of time between failures of FRU_i
$h_i(x)$	Hazard rate of FRU_i
$MTBF_i$	Mean time between failures of FRU_i
$MTTR_i$	Mean time to repair of FRU_i
τ_i	Delay caused by waiting for a new FRU_i to be delivered
t_i^{fail}	Time point when last failure of FRU_i occurred
t^{cur}	Current time when we need to update the spare pool
t^{next}	Next time when we need to update the spare pool
m_i	Impact FRU_i has on data unavailability
b_i	Unit price of FRU_i
B	Annual budget for spare provisioning

Table 5: Notations of Symbols

The notations of all the symbols used in this section are listed in Table 5.

5.2.1 Intuition and Assumption

The impact each FRU has on system availability is usually determined by two factors: its own reliability (e.g., some FRUs fail less often than others, or can be repaired more quickly) and the system architecture (e.g., in Spider I, as the lack of hardware redundancy makes the system more vulnerable to disk enclosure failures, disk enclosures have more impact on data availability). However, few existing ad hoc provisioning policies focus on these two factors simultaneously. Therefore, the basic idea behind our spare provisioning optimization model is to quantify both of these factors, and allocate more budget towards provisioning spare parts for FRUs that have more impact on the data availability of the storage system.

The assumption we made about the provisioning budget is simple but realistic. Specifically, at the beginning of each year, system administrators get a fixed budget that we call the annual budget, and use it to prepare spares for different FRUs according to a specific provisioning policy.

5.2.2 Reliability Characteristics of Different FRUs

Since we already have the field-gathered failure data and vendor-provided AFR, quantifying the reliability of each type of FRU is easy. We obtain the probability density function (PDF) of the time between the failures of a type of FRU by fitting the failure data. Thereafter, we can estimate the number of failures of such a type of FRU that will occur during a future period. For example, we use $f_i(x)$ to denote the PDF of the time between failures of FRU_i, then the CDF can be calculated as $F_i(x) = \int_0^x f_i(t)dt$. Based on this definition, the hazard rate of FRU_i is given as:

$$h_i(x) = \frac{f_i(x)}{1 - F_i(x)} \tag{3}$$

Let us assume the last failure of FRU_i occurred at time t_i^{fail}. We need to estimate the number of failures, y_i, of FRU_i between the current time, t^{cur}, when the spare pool is being updated and the next time, t^{next}, the spare pool will need an update. This is given as follows:

$$y_i = \int_{t^{cur}-t_i^{fail}}^{t^{next}-t_i^{fail}} h_i(x)dx \tag{4}$$

The above formula can estimate the expected number of failures between t^{cur} and t^{next} accurately if the time between the failures fits an exponential distribution, which has a time-independent hazard rate. However, for a Weibull distribution, if the time between updating the spare pool is relatively longer compared to the mean time between failures, this formula cannot give an accurate estimation. This is because, once a failure occurs between t^{cur} and t^{next}, which is very possible because of the short mean time between failures, the hazard rate should increase. Therefore, for a Weibull distribution, if

$$\frac{t^{next} - t^{cur}}{MTBF_i} > \int_{t^{cur}-t_i^{fail}}^{t^{next}-t_i^{fail}} h_i(x)dx, \tag{5}$$

we can use

$$y_i = \frac{t^{next} - t^{cur}}{MTBF_i}, \tag{6}$$

instead of (4), where $MTBF_i$ is the mean time between failure of FRU_i. Given the PDF of the time between failures of FRU_i, $f_i(x)$, $MTBF_i$ can be calculated by solving $\int_0^\infty x f_i(x)dx$.

In Spider I, if no spare part was available on-site, a device replacement will be delayed by at least 7 days. If we use $MTTR_i$ to denote the mean time to repair of FRU_i when spare parts are available on-site, τ_i to denote the delay caused by waiting for a new FRU_i to be delivered, then the total time spent on replacing FRU_i is $MTTR_i + \tau_i$, if there is no spare on-site when the replacement is required.

5.2.3 Impact of System Architecture on Availability

To quantify the impact of the system architecture on data availability, we need to analyze the physical structure of the system, and derive failure dependencies between the different FRUs. Here we consider one SSU of Spider I as an example. All FRUs of this SSU are listed in Table 2.

The RBD illustrates the failure dependencies between the different FRUs. By analyzing the structure of the RBD, we can derive the impact each FRU has on the data unavailability of the storage system. For instance, each RAID group in the SSU in Figure 4 contains 10 disk drives, which are organized as RAID level 6, and can tolerate 2 disk failures. Each leaf block represents a disk drive, and there are 16 different paths from one leaf block to the root. On each of these 16 paths, if one FRU fails, that path will be unavailable. If and only if all of these 16 paths are unavailable, the associated disk drive will become unavailable. If more than 2 disk drives in one RAID group are unavailable, a data unavailability occurs. In fact, the more available paths each RAID group has, the more reliable each RAID group is. Therefore, we can quantify the impact of each FRU on data unavailability by counting the number of paths that will become unavailable in one RAID group, if such an FRU has been removed from the RBD.

Specifically, since triple-disk unavailability in one RAID 6 group leads to data unavailability, we only count unavailable paths of each triple-disk combination in one RAID group. For example, failure of one controller makes every disk in one RAID group lose 8 paths, while the failure of one disk enclosure only makes two disks in one RAID group totally unavailable (each loses 16 paths). Therefore, we use $8 \times 3 = 24$ as the impact of a controller, while $16 \times 2 = 32$ as that of a disk enclosure. Table 6 shows the impact of each FRU quantified in this way.

5.2.4 Optimization Model and Dynamic Provisioning Algorithm

In order to maximize data availability, our optimization

	Quantified Impact
Controller	24
House Power Supply (Controller)	12
UPS Power Supply (Controller)	12
Disk Enclosure	32
House Power Supply (Disk Enclosure)	16
UPS Power Supply (Disk Enclosure)	16
I/O Module	16
Disk Expansion Module (DEM)	8
Baseboard	16
Disk Drive	16

Table 6: Quantified impact of each type of FRU

model tries to minimize the total unavailable time of the end-to-end paths that belong to each triple-disk combination of a RAID group in the RBD. For example, as we mentioned above, one disk enclosure failure makes a triple-disk combination in one RAID group lose 32 end-to-end paths. If the disk enclosure has no spare part on-site and cannot be replaced quickly, those 32 end-to-end paths will be unavailable for a longer duration, which increases the probability that all end-to-end paths of the triple-disk combination become unavailable within the same time interval.

We define a variable x_i to denote how many spare parts are provided for FRU_i. Then, the total unavailable time of the end-to-end paths, caused by failures of FRU_i can be calculated as

$$\Delta t_i^{down} = m_i x_i MTTR_i + m_i(y_i - x_i)(MTTR_i + \tau_i), \quad (7)$$

where m_i is the number of unavailable end-to-end paths caused by failures of FRU_i (see Table 6) and y_i is the estimated number of FRU_i failures that would occur before the next spare pool update.

Let us assume the unit price of FRU_i is b_i, the annual budget for spare provisioning is B. Then, we can establish the following linear programming optimization model to find out how many spares should be prepared for each type of FRU in the coming year.

$$\arg\min_{x_i} \sum_{i=1}^{N} m_i y_i(MTTR_i + \tau_i) - m_i x_i \tau_i; \quad (8)$$

$$\text{s.t.} \sum_{i=1}^{N} x_i b_i \le B; \quad (9)$$

$$x_i \le y_i, \forall i \in \{1 \ldots N\} \quad (10)$$

In this linear programming model, the objective function is to minimize the total unavailable time of the end-to-end paths that belong to each triple-disk combination of a RAID group in the RBD. The two constraints are that the total provisioning cost cannot exceed the annual budget and for each type of FRU, the number of provisioned spares should not exceed the expected number of failures.

The pseudo-code of the spare provisioning algorithm is shown in Algorithm 1. At the beginning of each of year, system administrators can first check the spare pool, and find out which FRU has no spare part. Then they can calculate all required parameters for these unprovisioned FRUs, and resolve the optimization model to find out those that need a spare part. Finally, based on the optimization results, they add the needed spare parts into the spare pool.

5.3 Continuous Provisioning Evaluation

Recall that we introduced two different ad hoc provisioning policies in Section 5.1 (controller-first and enclosure-first). We compare the performance of our continuous provisioning model with the two ad hoc provisioning policies

Algorithm 1 Spare Provisioning Algorithm

```
Input: Current spare pool $SP$, replacement log and unit price of each
  type of FRU, annual budget for spare provisioning $B$.
Output: Spare provisioning results $\mathbf{X} = [x_1, x_2, \ldots, x_N]$.
  Obtain number of spares in $SP$, $\mathbf{n} = [n_1, n_2, \ldots, n_N]$;
  Calculate $[m_1, m_2, \ldots, m_N]$;
  Calculate $[MTTR_1, MTTR_2, \ldots, MTTR_N]$
  for $i = [1, 2, \ldots, N]$ do
    Calculate $y_i$, the expected number of failures of $FRU_i$;
    Add $y_i$ into $\mathbf{Y}$, and $MTTR_i$ into $\mathbf{MTTR}$;
    Add $m_i$ into $\mathbf{m}$, and $b_i$ into $\mathbf{b}$;
  end for
  $\mathbf{X} = \text{ResolveOptimizationModel}(\mathbf{Y}, \mathbf{MTTR}, \mathbf{m}, \mathbf{b}, B)$
  for $i = [1, 2, \ldots, N]$ do
    if $n_i < x_i$ then
      Add $(x_i - n_i)$ spares $FRU_i$ in to $SP$;
    end if
  end for
```

in terms of data availability and provisioning cost, by using the tool introduced in Section 3.3. We again use the Spider I file system for the evaluation.

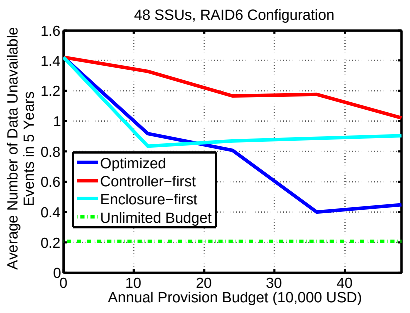
5.3.1 *Evaluation of Data Availability*

In this section, we present the evaluation results of different provisioning policies to demonstrate the effectiveness of our optimized provisioning policy in reducing data unavailability. Note that besides the two ad hoc provisioning policies we mentioned before, we also include the evaluation results of the scenario when unlimited provisioning budget is provided, which gives the lower bound for the data unavailability. Here, unlimited provisioning budget means every individual component in the system can have a spare part on-site. For example, in Spider I there are 96 controllers, thus we can maintain 96 spare controllers in the spare pool if unlimited budget were provided.

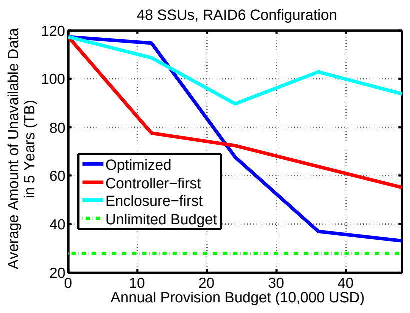
First, we present the results of the average number of data unavailability events during the 5-year operation of 48 SSUs, using different provisioning policies in Figure 8(a). The results illustrate that at least one data unavailability event will occur during 5 years if no provisioning policy is used (no provisioning budget is provided). Further, the optimized provisioning policy can reduce the data unavailability more significantly with increasing provisioning budget when compared to the ad hoc policies.

Since one data unavailability event might cause multiple RAID groups to become unavailable simultaneously, the volume of data that can become unavailable due to even a single unavailability event can range in the tens of terabytes. For the Spider I file system, each RAID level 6 group is composed of 10 1TB disks. Figure 8(b) shows the average amount of data that might become unavailable under different provisioning policies during 5 years of operations. We calculate this with the knowledge of how many RAID groups are affected by each data unavailability event. Similar to the results shown in Figure 8(a), the optimized provisioning policy can also reduce the amount of unavailable data significantly. For example, with an annual spare provisioning budget of just \$480K, the optimized provisioning policy can protect as much as 90TB from becoming unavailable during the 5-year operation of the storage system.

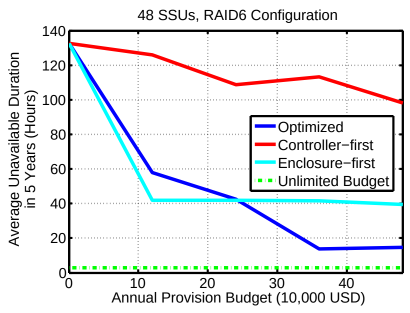
Moreover, the optimized provisioning policy also decreases the duration of data unavailability as shown in Figure 8(c). For the same \$480K annual provisioning budget, the optimized provisioning policy reduces the duration of data unavailability for the 48 SSUs in aggregate by as much as 52%



(a) Number of Data Unavailability



(b) Amount of Unavailable Data



(c) Average Unavailable Duration

Figure 8: Performance comparison between different provisioning policies

(more than 20 hours) and 81% (more than 80 hours) compared to the enclosure-first and controller-first provisioning policies.

Finding 8. *Optimal provisioning increases data availability significantly. In fact, as the provisioning budget increases, it continues to perform even better, getting closer to the unlimited budget policy.*

5.3.2 *Evaluation of Provisioning Cost*

This section presents an evaluation of the cost of different provisioning policies. First, we illustrate the total provisioning cost during 5 years using different provisioning policies, given different annual budgets, in Figure 9. Different from the two ad hoc policies, which try to squeeze every penny of the budget, the cost of our optimized provisioning policy does not increase with the budget linearly. The reason behind this is that the optimized provisioning policy allocates budget based on an accurate failure estimation and failure dependency analysis, which are more economical and efficient compared to the ad hoc policies.

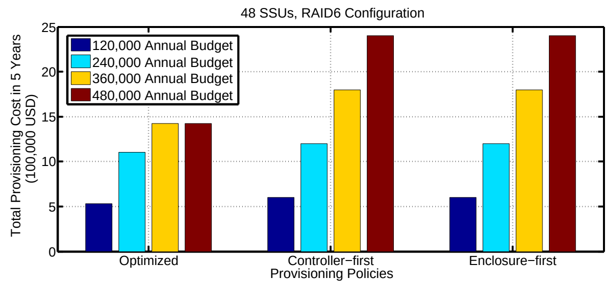


Figure 9: Total provisioning cost in 5 years using different provisioning policies.

Finally, we illustrate the cost for spare provisioning the 48 SSUs in each year using the optimized provisioning policy, given different annual budget limits (Figure 10). Two interesting observations can be made from Figure 10. First, the annual provisioning cost decreases year after year. This is because many FRUs in Spider I have decreasing failure rates (see Figure 2). Second, increasing the annual provisioning budget does not necessarily increase the annual provisioning cost. For example, when the annual budget is increased to $480K, the provisioning cost is almost the same as when the annual budget is $360K. This is because the optimized provisioning policy attempts not to over-provision the spare parts, i.e., no more spare parts will be added if what is available is equal to what is expected to fail next year, resulting in cost savings.

Finding 9. *Optimal provisioning results in significant cost-savings. The resulting savings can be more than 10% of the total storage system cost over the operational life of a large-scale storage system.*

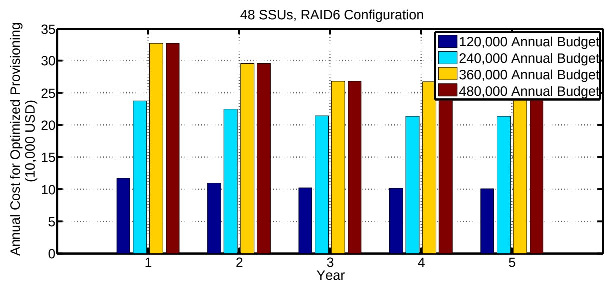


Figure 10: Annual cost for optimized provisioning policy.

6. RELATED WORK

Storage system reliability and data availability have been studied on several fronts. Analytical modeling, coupled with field data analysis and fitting are among the most common approaches. A large body of existing work focusses on building probability models for failures of disk drives and data loss in RAID groups [3, 10, 20, 23, 27, 34]. A few existing studies have also tried to estimate the reliability of a storage system through simulation [7, 8, 11]. In particular, Elerath and Pecht [7] have implemented a Monte Carlo simulation for RAID 4 groups to evaluate how time dependent failure and repair rates impact the average number of data loss events that could occur during a given mission time. Greenan developed a high-fidelity reliability simulator for erasure-coded storage systems [11]. All of the simulation-based approaches focus at the component-level, i.e., disk or RAID group failures. Our work is novel in the sense that we take an end-to-end approach to study reliability and its impact on the provisioning of a large-scale storage system. As aforementioned, field data suggests that failures of other hardware components contribute to considerable percentages of storage system failures. Therefore, an end-to-end and system-level approach is better suited to capture such failures, and bring it bear on both initial system provisioning as well as spare provisioning that is needed for continuous reliable operations after deployment.

Spare provisioning optimization has been extensively s-tudied in the industrial engineering area. Different optimization models have been proposed by a number of researcher-s. For instance, in order to guarantee a specific availability metric for the system, queuing theory based approaches have been frequently used to determine the number of spare part-s that should be prepared [1, 15, 16, 17]. Besides queuing theory, some optimization-based models [9, 32] were also proposed in the operations research (OR) area. However, due to the complexity of the extreme-scale distributed storage system, existing OR related spare provisioning models cannot be directly applied or non-trivially extended. Our optimized provisioning model relies on realistic system-level assumptions, and accounts for respective RBD diagrams, redundancy factors for different components, different repair times and the impact of errors To the best our knowledge, the insights drawn from our study have not been presented before, and are potentially useful to the community at large.

7. CONCLUSIONS

Designing an extreme-scale storage system is a balancing act to reconcile multiple decision factors. We have proposed a two-phased design approach, namely initial and continuous provisioning that can help alleviate this situation. Initial provisioning addresses the early stage of the procurement, and explores the tradeoffs between cost, performance and availability. Continuous provisioning provides a spare part provisioning model to ensure a highly available system operational experience. Both approaches leverage the insights gained from a detailed analysis of field operational data and a system-agnostic provisioning tool.

Our results provide several valuable insights. We find that the actual failure rate of disk drives is on par with vendor-provided statistics, but the time between disk failures fits better with a join of a Weibull and an exponential distribution. We also observe that scalable building blocks play a more dominant role in a performance-oriented HPC storage system, than the number of disks or their capacity per u-nit. Ad hoc provisioning policies that have been exploited by most supercomputer sites might not be optimal, and our dynamic provisioning policy is able to achieve higher data availability.

While our analysis was carried out based on *a* particular extreme-scale storage system, the approach, the provisioning tool and proposed policies are generally applicable to different storage architectures and configurations.

8. ACKNOWLEDGEMENT

We would like to thank the reviewers for their comments. This work was supported in part by the Oak Ridge Leadership Computing Facility, located in the National Center for Computational Sciences at ORNL, which is managed by UT Battelle, LLC for the U.S. DOE (under the contract No. DE-AC05-00OR22725). The work was also supported by a JDRD grant by the Science Alliance of the University of Tennessee and the National Science Foundation grant 0953238.

References

[1] M. Alam and V. Mani. Queueing Model of a Bi-level Markov Service-system and Its Solution using Recursion. *Trans. Reliability*, 37:427–433, Oct. 1988.

[2] B. Behlendorf. Sequoia's 55PB Lustre+ZFS Filesystem. In *Lustre User Group (LUG) Meeting*. OpenSFS, 2012.

[3] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. RAID: High-performance, Reliable Secondary Storage. *ACM Computing Surveys*, 26(2):145–185, June 1994.

[4] DataDirect Networks, Inc. S2A9900 Datasheet, http://www.ddn.com/support/downloads-documentation/, 2011.

[5] DataDirect Networks, Inc. DDN SFA12K Family, 2014.

[6] L. Devroye. Sample-based Non-uniform Random Variate Generation. In *Proceedings of the 18th Conference on Winter Simulation*, WSC '86, pages 260–265, New York, NY, USA, 1986. ACM.

[7] J. G. Elerath and M. Pecht. Enhanced reliability modeling of raid storage systems. In *In Proceedings of the International Conference on Dependable Systems and Networks (DSN*, pages 175–184, 2007.

[8] J. G. Elerath and J. Schindler. Beyond MTTDL: A Closed-Form RAID 6 Reliability Equation. *Trans. S-torage*, 10(2):7:1–7:21, Mar. 2014.

[9] B. Ghodrati, D. Benjevic, and A. Jardine. Product support improvement by considering system operating environment: A case study on spare parts procurement. *International Journal of Quality and Reliability Management*, 29(4):436–450, 2012.

[10] G. A. Gibson and D. A. Patterson. Designing Disk Arrays for High Data Reliability. *Journal of Parallel and Distributed Computing*, 17(1-2):4–27, Jan. 1993.

[11] K. Greenan. Reliability and Power-Efficiency in Erasure-Coded Storage Systems. Technical Report UCSC-SSRC-09-08, University of California, Santa Cruz, Dec. 2009.

[12] P. E. Greenwood and M. S. Nikulin. *A Guide to Chi-Squared Testing*. Wiley, New York, 1996.

[13] M. Holland and G. A. Gibson. *Parity declustering for continuous operation in redundant disk arrays*, volume 27. ACM, 1992.

[14] IBM DS8000 Series. http://www-03.ibm.com/systems/storage/disk/ds8000/overview.html, 2014.

[15] A. Jardine and A. Tsang. *Maintenance, Replacement, and Reliability: Theory and Applications*. Dekker Mechanical Engineering. Taylor & Francis, 2005.

[16] T. P. Lewis and J. K. Cochran. Applying Queueing Theory to Improve the Modeling of Spares Provisioning of Small Combat Aircraft Units. In *Proceedings of the 17th International Conference on Computers and Industrial Engineering*, ICC&IE '94, pages 297–301, Tarrytown, NY, USA, 1995. Pergamon Press, Inc.

[17] V. Mani and V. Sarma. Queuing Network Models for Aircraft Availability and Spares Management. *Trans. Reliability*, R-33(3):257–262, Aug. 1984.

[18] NetApp, Inc. FAS8080 EX, http://www.netapp.com/us/products/storage-systems/fas8000/, 2014.

[19] Panasas, Inc. ActiveStor 16,http://www.panasas.com/products/activestor, 2014.

[20] D. A. Patterson, G. Gibson, and R. H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data*, SIGMOD '88, pages 109–116, New York, NY, USA, 1988. ACM.

[21] Personal Communications. Spider I system administrators on component replacement time, June 13, 2014.

[22] E. Pinheiro, W.-D. Weber, and L. A. Barroso. Failure Trends in a Large Disk Drive Population. In *Proceedings of the 5th USENIX Conference on File and Storage Technologies*, FAST '07, pages 2–2, Berkeley, CA, USA, 2007. USENIX Association.

[23] K. K. Rao, J. L. Hafner, and R. A. Golding. Reliability for Networked Storage Nodes. In *International Conference on Dependable Systems and Networks (DSN)*, pages 237–248. IEEE Computer Society, 2006.

[24] M. Rausand and A. Hoyland. *System Reliability Theory: Models, Statistical Methods and Applications*. Wiley-IEEE, 3 edition, Nov. 2003.

[25] K. Sakai, S. Sumimoto, and M. Kurokawa. High-performance and highly reliable file system for the k computer. *FUJITSU Science Technology*, 48(3):302–209, 2012.

[26] B. Schroeder and G. A. Gibson. Disk Failures in the Real World: What Does an MTTF of 1,000,000 Hours Mean to You? In *Proceedings of the 5th USENIX Conference on File and Storage Technologies*, FAST '07, Berkeley, CA, USA, 2007. USENIX Association.

[27] M. Schulze, G. Gibson, R. Katz, and D. Patterson. How Reliable Is A RAID. In *COMPCON Spring ÂŠ89. Thirty-Fourth IEEE Computer Society International Conference: Intellectual Leverage, Digest of Papers*, pages 118–123. IEEE, 1989.

[28] T. J. E. Schwarz, Q. Xin, E. L. Miller, D. D. E. Long, A. Hospodor, and S. W. Ng. Disk Scrubbing in Large Archival Storage Systems. In *12th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2004), 4-8 October 2004, Vollendam, The Netherlands*, pages 409–418, 2004.

[29] Seagate Technology. ClusterStor 9000, 2014.

[30] G. Shipman, D. Dillow, S. Oral, and F. Wang. The Spider Center Wide File System: From Concept to Reality. In *Cray User Group (CUG) Conference*, Atlanta, May 2009.

[31] Top500 Site:. http://top500.org/lists/2010/06/, June, 2010.

[32] T. S. Vaughan. Failure Replacement and Preventive Maintenance Spare Parts Ordering Policy. *European Journal of Operational Research*, 161(1):183–190, 2005.

[33] L. Wan, F. Wang, S. Oral, S. S. Vazhkudai, and Q. Cao. A report on simulation-driven reliability and failure analysis of large-scale storage systems. Technical Report ORNL/TM-2014/421, Oak Ridge National Laboratory, December 2014.

[34] Q. Xin, E. L. Miller, T. J. E. Schwarz, D. D. E. Long, S. A. Brandt, and W. Litwin. Reliability Mechanisms for Very Large Storage Systems. In *IEEE Symposium on Mass Storage Systems*, pages 146–156, 2003.