Getting Back Up: Understanding How Enterprise Data Backups Fail

George Amvrosiadis, ***University of Toronto;*** **Medha Bhadkamkar,** ***Veritas Labs***

https://www.usenix.org/conference/atc16/technical-sessions/presentation/amvrosiadis

This paper is included in the Proceedings of the 2016 USENIX Annual Technical Conference (USENIX ATC '16).

June 22–24, 2016 • Denver, CO, USA

978-1-931971-30-0

Open access to the Proceedings of the 2016 USENIX Annual Technical Conference (USENIX ATC '16) is sponsored by USENIX.

Getting Back Up: Understanding How Enterprise Data Backups Fail

George Amvrosiadis
Dept. of Computer Science, University of Toronto
gamvrosi@cs.toronto.edu

Medha Bhadkamkar
Veritas Labs
medha.bhadkamkar@veritas.com

Abstract

In the enterprise world, retaining data backups is the de-facto solution against data loss in the event of catastrophic failures. As backup software evolves to achieve faster backup and recovery times, however, backup systems deploying it become increasingly complex to administer. This complexity stems from optimizations targeted to specific applications, which increase the number of configuration parameters for the system. Still, there is no work in the literature that attempts to study the error characteristics of enterprise backup systems, despite our reliance on the guarantees they provide.

With this study we aim to help researchers and practitioners understand how backup system jobs fail, and identify factors that can be used to predict these failures. Our results are derived from an analysis of data on 775 million jobs, collected from more than 20,000 backup software installations over a span of 3 years. We confirm that trends reported in the software reliability literature also hold for backup systems, such as that the majority of job errors are due to misconfigurations. For the systems in our dataset, we find that error rates remain stable across software versions and over time. To better understand these errors, we investigate the effect of several factors on the system's error rate, such as job sizes and policy complexity, and demonstrate their predictive power for future errors.

1 Introduction

From enterprise organizations to home users, data backups are still the preferred mechanism for the prevention of data loss in the event of catastrophic failures. The guarantees provided by backup software, however, rely on the assumption that periodic backup jobs will complete successfully. Unfortunately, backup software is becoming increasingly complex to configure and manage, as more tuning parameters are introduced to handle diverse application requirements. Recent surveys of CIOs and IT professionals show that 27% of businesses have trouble recovering from backups due to backup jobs not completing successfully [18], and 80% experience challenges managing backup data and configuring the backup software [35]. At the same time, data generation rates increase. In a recent study, we found that full backups are performed as often as every 1-4 days to accommodate data churn [3], leaving less time to repeat failed jobs.

The goal of this study is to help researchers and practitioners understand how backup system jobs fail, and identify the factors that can be used to predict these failures. Our results are based on the analysis of periodic reports collected from customer installations of Veritas NetBackup [37], a commercial backup software product, over the span of 3 years. In total, we studied 775 million jobs from more than 20,000 *backup systems*, i.e. customer installations. Depending on their type, these jobs perform specific operations, such as data backup, recovery, and backup data management (e.g. replication). Jobs are scheduled according to *backup policies*, which are sets of configurable parameters dictating how individual applications should be backed up. For example, VMware policies expose parameters specific to virtual machines, while Oracle policies expose parameters relevant to database instances.

First, we investigate the prevalence of errors in backup systems and their causes. We find that backup system jobs fail frequently, and the majority of errors are attributed to misconfigurations, which confirms a recurrent trend in the literature for other system types [11, 24, 25, 34, 44]. On the bright side, we find that the errors themselves are not diverse, and the 10 most frequent error codes returned by jobs account for more than 78% of job errors. We explain these errors in detail, providing guidelines that designers can leverage to improve the robustness of future backup software.

Next, to understand why errors occur, we study the context in which they appear. Specifically, we study characteristics of the backup system and individual jobs, and the degree to which they affect the frequency and diversity of the occurring errors. For example, we find that factors such as the size of the job, and the configuration complexity of a policy, are likely linked to job failures. On the other hand, factors such as the frequency of configuration updates and software versions, are likely not tied to failures. By quantifying the relationships between

Characteristic	Observation	Section	Previous work
Prevalence	In the average backup system, 15.2% of jobs terminate with an error. The overall job error rate across all systems in our dataset remains stable over time.	4.1	None
Diversity	The 10 most frequent error codes correspond to more than 78% of job errors.	4.2	None
Causes	More than 75% of job errors are due to misconfigurations.	4.3	Similar trends for different applications [11, 24, 25, 27].
Dependence on job properties	Jobs managing (e.g. replicating) existing backup images exhibit the highest error rates.	5.1	None
	Systems with larger jobs tend to experience higher job error rates.	5.2	None
	Policies with more configurable parameters exhibit higher job error diversity.	5.3	None
Dependence on system properties	Backup software and operating system versions have no effect on job error rate.	6.1	None
	The size and load of a backup system is indicative of its job error diversity.	6.2	None
	The rate of configuration changes has no effect on the system's error rate or diversity.	6.3	None
Predictability	The number of daily occurrences of a given error code follows predictable trends, but error inter-arrival times do not.	7.1	None
	Job errors can be fairly accurately predicted based on the occurrence of other error codes. The factors in previous observations, however, make superior predictors.	7.2	None

Table 1: A summary of the most important observations of our study regarding backup system job errors.

each factor and the system's resulting error rate, we aim to provide administrators with rules of thumb that can be consulted during system configuration.

For monitored systems, we receive weekly reports that allow us to observe how these systems evolve over time. We show that the number of daily errors in a backup system follows predictable trends, while error inter-arrival times do not. Finally, we demonstrate that while different error types are correlated, the factors we identify in the observations of our study are better predictors of future error occurrences. We find these results encouraging, suggesting that future backup systems could rely on prediction models to automatically detect and self-heal from errors, without affecting their reliability guarantees.

Table 1 summarizes the most important observations of our study. The remainder of the paper is organized as follows. In Section 2 we present an overview of prior work on backup systems and related work in the software reliability literature. The dataset we use in this study is introduced in Section 3. First, we analyze this data to understand the prevalence and causes of job errors in backup systems in Section 4. Then, we identify job characteristics that can be correlated to higher error rates in Section 5. In Section 6 we examine the relationship between properties of the backup system, and its reported error rate. In Section 7 we evaluate the predictability of errors, and the predictive power of our observations from previous sections for future errors. Finally, we outline the implications of our results in the design of future backup systems in Section 8, and conclude in Section 9.

2 Related work

Little work exists to characterize the operation of backup systems. Park and Lilja [26] used traces of weekly full backup operations from 6 systems to characterize their deduplication ratios. Wallace et al. [39] study the contents and workload of file systems that store data produced by the jobs of backup systems such as NetBackup. Our earlier work [3] identifies and characterizes trends in the configuration and job characteristics of backup systems. Other work [10, 36] has focused on modeling the growth rate of storage capacity in backup systems, but it does not examine the prevalence of storage capacity errors. Overall, none of the aforementioned work looks into the characteristics of errors in the operation of backup systems.

Despite the lack of prior work on backup systems, several studies have examined the characteristics and prevalence of software and administration errors in other system types. In his seminal paper on system faults, Gray [11] reports that 25% of faults in high-end mainframes are due to software errors, while 42% are attributed to administrator errors. Similarly, Patterson et al. [27] observed that 8-34% of failures in telephone networks and Internet systems were due to software, while 51-59% were due to administration errors. Oppenheimer et al. [25] studied the failures of Internet Services and report that failures are due to software errors 33% of the time, while 57% are administration mistakes, and Nagaraja et al. [24] confirm these findings in a user study. Tang et al. [34] report that 16% of high-impact incidents over a three-month period at Facebook were due to misconfigurations, but do not provide a more detailed breakdown. Our study is the first to consider backup systems, but it is worth noting that our breakdown of backup system errors by cause (Figure 5) matches closely the results reported in the literature for other system types.

Due to the prevalence of misconfigurations, several research efforts have put forth ideas to detect, diagnose, and automatically fix these errors. PeerPressure [40] uses a statistical approach and a repository of configurations

Job type	Entries	Function
Backup	604.9 M	Create backup images
Management	105.8 M	Manage (e.g. delete, replicate, migrate) backup images
Snapshot	58.2 M	Create Copy-on-Write data snapshots
Recovery	6.3 M	Recover data from backup images
Total	**775.2 M**	

Table 2: Breakdown of job types in our dataset.

to identify parameter errors in a given configuration. In a similar manner, Yuan et al. [45] identify error root causes using support vector machines to detect anomalous patterns in system call sequences. CODE [46] extends this idea by identifying invariants for configuration access rules. EnCore [47] adopts a learning approach guided by sample configurations, augmented with information on the execution context of the configurations. Chronus [42] retains disk state through periodical checkpoints, and traverses them to detect the configuration changes responsible for the misconfiguration. ConfAid [6] instruments application binaries to trace the configuration entry responsible for the misconfiguration. AutoBash [5, 33] leverages carefully crafted bug-tracking predicates to detect deviations from healthy machines, and uses a speculative OS kernel to find a fix through trial and error and a solution database. Such approaches can benefit greatly from studies outlining the contributing factors to errors. Observations derived from such studies can be used as heuristics for learning approaches, and to prune the state space for data-flow analysis techniques. Furthermore, while these are successful reactive approaches to error occurrence, we also examine the feasibility of proactive approaches, such as error prediction.

3 Dataset description

Our analysis and models are based on data from *telemetry reports* collected from enterprise customer installations of a commercial backup product, Veritas NetBackup [37]. Reports are only collected from customers who opt to participate in the telemetry program, and contain no personal identifiable information. This section outlines the characteristics of our telemetry dataset.

Reported information. Telemetry reports received from customer backup systems contain runtime and configuration information about these systems. This runtime information describes the jobs that run in each system, and whether they completed successfully. The different job types recorded in monitored backup systems are described in Table 2. Jobs run in backup windows, the time and duration of which are configured by the system administrator. When a job fails to complete successfully, it can be retried in the current or next backup window, subject to time availability and priority compared to other jobs. By default, backup jobs are retried once, while other job types are never retried. The job scheduler will also cancel retries of jobs that are scheduled to recur in the current backup window. Our telemetry reports record the error codes returned by failed jobs, but do not allow us to distinguish job retries. In our analysis we use job error codes in conjuction with other system or job characteristics, to investigate why jobs fail.

Dataset size and breadth. Our dataset consists of 1 million telemetry reports, collected over the span of three years, between September 2012 and August 2015. These reports represent 22 minor versions of the backup software, all under the same major version: NetBackup 7.6. More than 20,000 customer installations are represented, across most modern operating systems. Note that this dataset is different from the one used in our earlier work [3]: we have excluded systems that deploy versions of NetBackup earlier than 7.6, since their telemetry reports do not include error information, and extended the dataset by collecting reports for another year.

Architecture. Modern backup systems typically consist of three tiers of operation: a master server, one or more storage servers, and several clients [3]. The *master server* is responsible for scheduling and monitoring backup jobs, while *storage servers* manage the archival of backup images. In smaller systems, the master server can be consolidated with a storage server. In our dataset, 27.9% of backup systems use dedicated storage servers, while the rest consist of one master/storage server.

Monitoring duration. The backup systems described in our study were each monitored for 5.7 months on average, and for a maximum of 34 months. Note that the monitoring time is not always equivalent to the total lifetime of the backup system, as most of these systems were still online at the time of this writing. For some systems, our dataset contains only a single weekly telemetry report. We are unable to distinguish whether these reports correspond to systems with a lifespan of one week or less, refer to failed installations, or are due to reporting errors. Our dataset also contains reports from systems used internally in Veritas for development and quality assurance. As part of pre-processing, we have chosen to exclude almost 5,000 systems from our analysis, which belong to either one of these categories.

4 Error characteristics

In this section, we use our dataset to characterize the frequency of job errors in backup systems. We further investigate the diversity in the error codes returned by in-

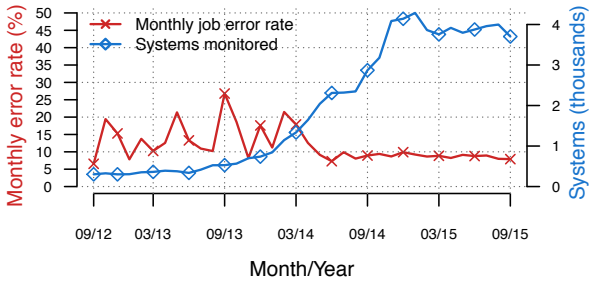

Figure 1: Monthly job error rates in a 3-year time period.

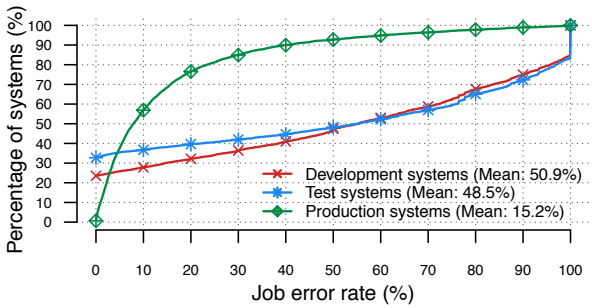

Figure 2: CDFs for the job error rate of individual backup system types.

Category	Version	Jobs	Lifespan	Systems
Production	Stable	176,173	76 days	4,220
Development	Alpha, Beta	5,965	18 days	5,217
Test	Stable	112	10 days	6,066

Table 3: Characteristics of the different types of backup systems in our dataset. The number of jobs and lifespan refer to averages across individual systems.

dividual jobs, and analyze them to find the most popular causes for job errors.

4.1 Prevalence of job errors

After pre-processing, our dataset consists of 775.2 million jobs across 15,503 systems. Of these jobs, 69.4 million (or 8.7%) terminate with an error code indicating a partial, or complete failure. In Figure 1, we show the monthly job error rate across all recorded jobs. Note that as more systems join the telemetry program (right y-axis), the error rate becomes more stable (left y-axis). Since error rates are not improving over time, a better understanding of backup system errors seems imperative.

It is common for customers to test development or stable versions of the backup software. The installations used in this process have radically different error and workload profiles, as will be demonstrated later in the paper. Thus, we have grouped these systems and report results on each system type separately. The different system categories are listed in Table 3. *Production systems* are deployed in the field. They have long lifespans, most still being online at the time of this writing, and are by far the busiest systems. *Development systems* are run by partner organizations that are given early access to new software features through alpha and beta versions. These systems are short-lived, but run workloads with job frequencies comparable to production systems. Finally, *test systems* are customer systems that run stable versions of the software, for the purpose of testing its operation before deploying it in production. These systems are also short-lived, but run a small number of jobs, presumably to test different aspects of the system's operation.

Only 3,243, or 20.9%, of the systems in our dataset exhibit no errors in the time they are monitored. Surprisingly, only 13 of these systems are used in production, while the rest are used for test and development purposes. The three system types also differ with regards to the job error rates they exhibit. Figure 2 shows the empirical cumulative distribution functions (CDFs) for the error rates of systems of each type. The error rate for jobs in production systems is 15.2% on average. Through Kolmogorov-Smirnov tests [23], we confirmed that the job error rates for production systems follow a Weibull distribution, with a shape parameter of $(76.1 \pm 0.9) \times 10^{-2}$ and a scale parameter of 13.0 ± 0.3. Note that the Weibull distribution is typical in systems reliability research, especially for modeling failure rates [30, chapter 2.12]. On the other hand, job error rates for development and test systems approximate the uniform distribution. On average, jobs in development and test systems fail 48.5% and 50.9% of the time, respectively. This is not surprising, as one would expect these systems to be used for testing a variety of scenarios in order to iron out problems before deployment in production. Note that these numbers differ from those in Figures 1 and 9, which aggregate jobs across all backup systems.

Observation 1: *15.2% of jobs terminate with an error in production backup systems. Testing and development systems exhibit up to 3.3 times more errors on average.*

4.2 Error diversity

In NetBackup, there are 1,194 distinct error codes that can be returned when a job fails partially, or completely [38]. Of these error codes, only 333 (or 27.9%) are reported in our dataset.

Backup systems of different types experience different sets of error codes, as jobs fail for different reasons. Specifically, 21 error codes occur only in development systems. These errors are caused by commands failing due to permission or communication issues, invalid in-

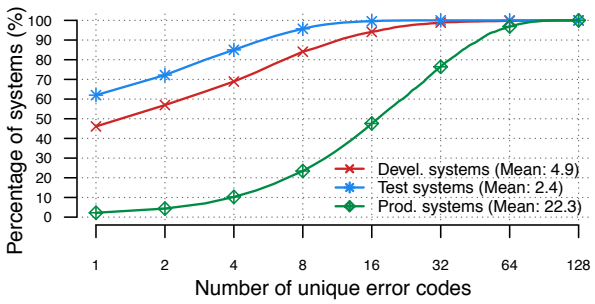

Figure 3: CDFs for the number of unique job error codes for each type of backup system.

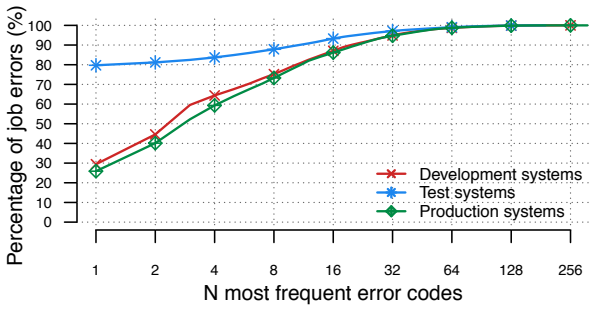

Figure 4: Percentage of job errors due to the N most frequent error codes for different backup system types.

puts, and unsupported or unavailable software features. Similarly, 59 error codes are specific to production systems. These errors occur as a result of communication errors due to offline system components, product licensing issues, misconfigurations, or the inability to fit all scheduled jobs in the specified backup window. Interestingly, the set of error codes that occur in test systems is a strict subset of the errors occurring in both development and production systems. In other words, error codes that occur during testing are likely those that survived development, but they make up only a strict subset of the errors that occur in production.

The set of error codes experienced by individual backup systems can also differ. Figure 3 shows the empirical CDFs for the number of unique error codes exhibited in individual backup systems throughout their lifespan. While average test and development systems experience 2.4 and 4.9 error codes respectively, production systems can exhibit 22.3 error codes on average, almost an order of magnitude difference.

Observation 2: *Production systems experience an order of magnitude more error codes, for three orders of magnitude more jobs run compared to test systems.*

The number of error codes exhibited by a system is not an indicator of the system's job error rate. This is a result of a few error codes being responsible for the vast majority of job errors. In Figure 4, we show the percentage of job errors that correspond to the N most frequent error codes. Jobs are partitioned by system type. Overall, we find that the 10 most frequently occurring error codes correspond to 78.1%, 79.0%, and 89.5% of job errors in production, development, and test systems respectively. Among these errors, half are unique for each system type, while the remaining 5 error codes are common across all systems. Of those, the most common error code denotes a partially successful backup job, which failed to back up some of the requested data because the backup process was unable to access it. Other common error codes occur due to insufficient backup storage capacity, as a result of the backup window being too short to fit all scheduled jobs, or because the storage unit was unreachable.

Observation 3: *The 10 most frequent error codes correspond to 78.1-89.5% of job errors, depending on the system type.*

4.3 Error causes

A smaller number of errors in a backup system implies fewer failure scenarios that need to be accounted for. It is not indicative, however, of the effort necessary to resolve each failure. Since many error codes in NetBackup are overloaded, we are unable to classify them based on their severity. Instead, we categorize error codes based on their cause, and analyze the distribution of job error causes across backup system types.

We manually categorized the error codes that appear in our dataset into three categories, using the troubleshooting guide provided to administrators of the backup software [38]. We classify an error as a *misconfiguration* if it can be attributed to configuration parameters that have been assigned incorrect values, or system components configured in a way that obstructs the correct operation of the backup system. Examples of such errors include: inability to backup data due to incorrect file permissions or locked files, jobs that were aborted because the backup window was configured too short, authentication errors, and jobs that failed due to insufficient backup storage capacity. An error is classified as a *system error* when it describes events that are not directly in the system administrator's control, in both the software and hardware layers. Examples of such errors include: unavailability of system components due to issues at the storage layer, errors at the operating system level, and internal errors of the backup software or applications it interfaces with. Finally, we classify as *informational messages* all error codes that describe non-fatal but unusual scenarios, such as jobs terminated by the administrator, warning messages, and product licensing issues.

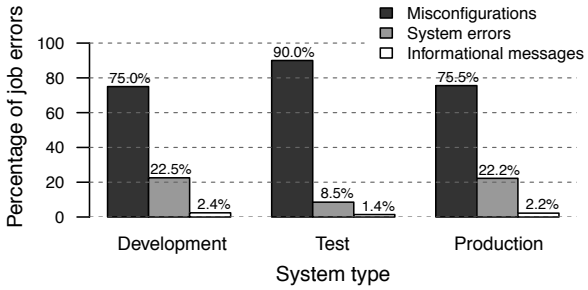

Figure 5: Breakdown of job errors based on their cause.

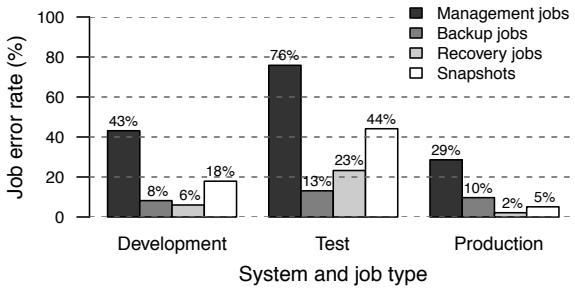

Figure 6: Error rates for different job types, across backup system types.

In Figure 5, we show a breakdown of job errors based on their cause. A separate breakdown is provided for each system type. As is evident, the majority of job errors are due to misconfigurations: 90% in test systems, and over 75% for development and production systems. The remainder of the errors are mostly due to system errors, while 1-2% are informational messages. This result is encouraging because misconfiguration errors usually identify which parameters were incorrectly set, or the constraints that did not apply, causing the error. As a result, they are significantly easier to address compared to system errors, which could start at the hardware level, or any of the storage and operating system layers above. Therefore, we expect that misconfigurations in NetBackup will be a good fit for self-healing approaches. Motivated by this result, we next look into factors that could be used to predict misconfigurations.

Observation 4: *Depending on system type, 75-90% of job errors are attributed to misconfigurations.*

5 Dependence on job properties

To help us prevent job errors, we attempt to identify heuristics that would help us predict them. To derive these heuristics, this section studies the effect of job properties on the manifestation of errors. Specifically, we analyze correlations between job errors, and the characteristics of the affected jobs, such as their type, size, and backup policy type they are part of.

5.1 Job type

We categorize jobs based on the types described in Table 2. Of all the error codes that appear in our dataset, we find that 45.6% are exclusive to a specific job type, while 13.3% are common across all job types. Error codes that are exclusive to specific job types are mostly due to misconfigurations, while error codes that are universal across job types mostly refer to component unavailability, or generic system errors. The remaining error codes are shared between all job types except recovery jobs. Recovery jobs are initiated and controlled directly by users, and only a single error code is logged on error. We do not have sufficient information to correlate data loss due to recovery errors with other job errors.

Recall that in our dataset, the majority of jobs are backups (Table 2). These jobs, however, exhibit low error rates across all system types. In Figure 6, we show the error rates for each job type, across all system types. As expected, production systems have significantly lower job error rates compared to other systems. Overall, management jobs exhibit by far the highest error rates, as high as 29% even in production systems. We find that across system types, management job errors are attributed to operations that attempt to replicate backup images either within the same backup system, or by migrating them to remote backup systems that are administered independently. Most operations that replicate backup images within the same backup system fail due to insufficient storage space, while some fail due to storage device unavailability. On the other hand, most inter-system replication operations fail due to configuration incompatibilities between the systems.

Observation 5: *Management jobs, especially jobs replicating backup images, exhibit the highest error rates.*

5.2 Job size

A limitation of our dataset is that job sizes are not collected individually for each job. Instead, telemetry reports contain the total number of bytes transferred in a given backup system on a given day. We use this information in conjunction with the number of jobs that were executed on the system, to derive the average job size and error rate over the lifespan of the system. Using this data, we study correlations between a system's average job size and its job error rate.

Test and development systems run few jobs that are mostly small in size, and thus there are no significant trends to report. On the other hand, we find that production systems exhibit significant positive correlations be-

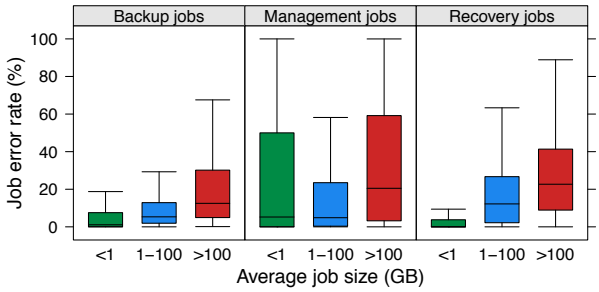

Figure 7: Box plots describing the relationship between the average backup system job size, type and error rate.

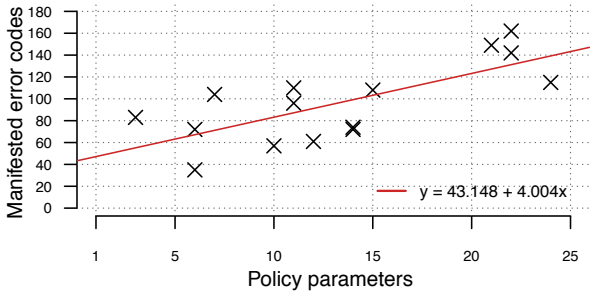

Figure 8: The number of unique error codes due to jobs of a policy type, as a function of the number of policy parameters. Each data point represents a policy type.

tween the average job size in the system, and the system's average job error rate. This holds in different degrees for backup, management, and recovery jobs. In Figure 7, we show box plots describing this relationship for each job type. The lower and upper edges of the boxes represent the 25^{th} and 75^{th} percentiles of job sizes across systems, respectively. The middle bar corresponds to the median, and the whiskers mark the largest and smallest data points in the distribution. Snapshot operations are excluded, as they incur no data transfer. Note that backup and recovery job error rates tend to increase significantly for larger job sizes. Smaller management jobs, however, are characterized by a wider spread. Management jobs of such small sizes are mostly operations used to make changes to the backup system that do not incur data transfers, such as deleting backup images, or examining virtual machine configurations prior to backup. Overall, we find that the most common errors in systems with larger jobs are due to insufficient available storage. In systems with smaller jobs, errors relevant to component unavailability are more common.

Observation 6: *Production systems with larger job sizes tend to experience higher job error rates.*

5.3 Policy type

To achieve consistent backups, applications may require a specific sequence of operations to take place. Thus, backup software offers predefined *policies* tailored to individual applications. For example, a Microsoft Exchange Server policy also backs up the server's transaction log, capturing updates since the backup started. Users can configure policies to specify the characteristics of the backup process, such as the frequency of backup jobs and the retention rate for backed up data. The majority of parameters, however, are policy-specific, e.g. whether to skip unallocated blocks of virtual disks, or back up database redo logs.

We find that the number of unique error codes returned by jobs running under a given policy, is strongly correlated (Pearson's $r_p = 0.73$) with the number of policy parameters exposed to users. In Figure 8 we show the number of unique error codes due to jobs running under a given policy type, as a function of the number of configurable parameters for the policy type. Our results focus only on production systems, which account for 95.9% of jobs. To fit a line to the data we used R's `lm` package [28]. The residual standard deviation (RSD) for the fit is 25.7 error codes, and the line's slope and intercept were chosen with p-values $< 10^{-2}$. The fit indicates that backup system policies are responsible for an additional 4 error codes per configurable parameter, in addition to 43 error codes that are common across policies. Note, however, that we find no correlation between a policy's parameters and its error rate. More complex policies increase the system's error diversity, but not the error count.

Figure 8 reports the number of unique error codes across all systems. Looking at individual backup systems, we also find a strong correlation ($r_p = 0.73$) between the number of policy types deployed, and the number of error codes in a given system. This is important, because we find that individual systems in our dataset deploy 10.7 policy types on average. The best least-squares fit for this relationship is $e = 1.1p + 0.07p^2$, where e is the number of error codes in the system, and p is the number of policy types. The RSD of the fit is 11.7 errors, with coefficient p-values $< 10^{-11}$. While our findings suggest that many error codes do not manifest in a given system, they highlight the relationship between policy configuration and error diversity.

Observation 7: *The number of configurable parameters in a backup policy is positively correlated with the diversity of errors returned from its jobs.*

6 Effect of system setup

We have shown that diversity within the backup system, at the level of policy and job characteristics, can be linked to the number and diversity of job errors. Backup

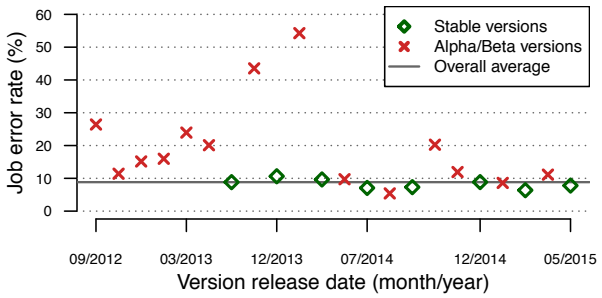

Figure 9: Job error rates across different versions of the backup software.

systems, however, can also differ with regards to the characteristics of the environment in which they are deployed. This section looks into the effect of such factors on job errors: the backup software and operating system versions, the backup system's size and load, and the rate of configuration changes in the system.

6.1 Software components

Overall, we examined backup systems running 22 versions of the backup software. We grouped jobs based on the software version under which they ran, and Figure 9 shows the job error rate for the different versions. We find that the job error rate remains stable across most software versions, approximating the average error rate across all jobs, which is 8.7%. As one would expect, earlier alpha and beta versions of the backup software, running on development systems, demonstrate higher error rates. These error rates are also accentuated by the short lifespan (see Table 3), and the purpose of these systems.

The systems we study are also diverse with regards to the operating environments of their clients, i.e. the machines contributing their data for backup. Backup system clients in our dataset represent several operating system families: Windows, Linux, OS X, AIX, and others. Moreover, they deploy 45 different versions of these operating systems, and many of them often appear in the same backup system. We find no evidence that this client-side heterogeneity can affect the job error rate in the system. This heterogeneity, however, does explain the large number of deployed backup policies (recall Section 5.3). Some policies are tailored to specific operating system families, and we find that backup systems are usually diverse enough to deploy many such policies concurrently, across different clients. Specifically, the majority of backup systems consist of clients deploying at least 6 operating system versions spanning 3 operating system families.

Observation 8: *A system's job error rate remains unaffected across backup software versions and heterogeneity in client operating systems.*

6.2 System size and load

The weekly telemetry reports we collect from customer systems also contain runtime information about the system, such as the number of clients that are active in the system on a given week. We use the number of clients as indication of the system's size, which we find to be non-linearly correlated (Spearman's $r_s = 0.67$) to the system's load. We define the load, or *weekly job rate*, of the system as the average number of jobs that run in the system in a given week. Due to the non-linear relationship between the two quantities, we apply a logarithmic transformation on both quantities before using linear regression to model their relationship. The resulting model is $c = 0.69\,j^{0.6} - 1$, where c is the number of clients, and j is the weekly job rate of the system. The model's RSD is 8.67 clients, with coefficient p-values $< 10^{-8}$. This relationship is likely attributed to clients that join backup policies once they are added to the system, although not all systems follow this tactic [3].

The weekly job rate of a backup system is also correlated ($r_s = 0.56$) to its client heterogeneity. As in Section 6.1, we define a system's client heterogeneity based on the number of different operating system versions represented across its client base. After applying a logarithmic transformation to the weekly job rates, we were able to fit a linear model to the data with RSD of 5.9 operating systems, and coefficient p-values $< 10^{-2}$. Note that while residual errors are too large for prediction purposes, due to high variability in the data, this model indicates the relationship between heterogeneity and system load. The number of client operating systems is expressed as $o = 0.77 \log j + 0.1(\log j)^2$.

Finally, we find that the weekly job rate is indicative ($r_s = 0.62$) of error diversity in the system. Recall that error diversity is defined as the number of unique error codes returned by jobs in a given system. We have fitted a linear model after applying a logarithmic transformation on weekly job rates. The model's RSD is 13.5 errors, with coefficient p-values $< 10^{-15}$. The number of unique error codes in the system is expressed as $e = 0.55(\log j)^2$.

Figure 10 shows the shapes of the curves describing the relationships between a given backup system's weekly job rate, and the three quantities: the number of clients, the number of client operating systems, and the number of unique error codes. We find no evidence, however, that any one of these quantities is indicative of the system's job error rate. In other words, while larger systems should be expected to be busier, more heterogeneous, and exhibit higher error diversity, they shouldn't be expected to have jobs fail at higher rates.

Observation 9: *A backup system's size tends to be indicative of: its load, its clients' heterogeneity, and its error diversity.*

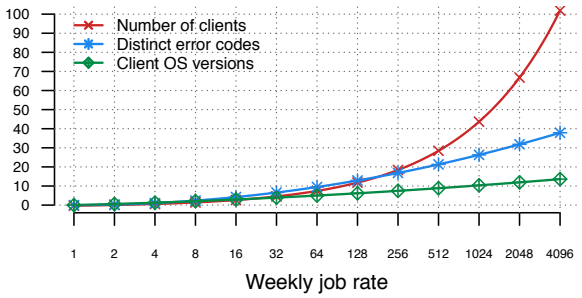

Figure 10: Fits describing the relationships between the average weekly rate of jobs in a backup system and its size, error diversity, and client heterogeneity.

6.3 Configuration changes

For each monitored backup system, we collect weekly statistics on the number of clients, policies, and storage devices configured with the system. To describe the variability for these quantities over time, we estimate their *coefficient of variation*. This metric is defined as the ratio of the standard deviation across our weekly measurements, compared to the overall mean value for the system. As a result, the coefficient corresponds to the fraction of the mean by which a given measurement is expected to deviate, i.e. larger values imply wider spread.

We find that variability in the number of clients of a backup system is strongly positively correlated to the number of backup policies in the system. This implies that the addition and removal of policies in a system coincides with variability in the number of clients. This is likely due to policies that define clients that follow them upon their addition to the system, and due to deleted policies rendering some clients inactive upon their removal.

Lastly, we find no correlation between the variability of a backup system's configuration and its job error rate, or its error diversity. This is attributed to the fact that across all systems, these quantities are characterized by low variability throughout the system's lifespan. Variability in the number of storage devices in the system is also not correlated to any of the previous metrics, for the same reason.

Observation 10: *The rate of configuration changes in a backup system has no effect on its job error rate, or its error diversity.*

7 Error predictability

The majority of errors occurring in a backup system require administrator intervention in order to be resolved (Section 4.3). As a result, we expect these errors to recur over time. In Section 7.1 we examine whether this property describes the errors in our dataset, and whether it can help us predict future errors. Furthermore, jobs in a backup policy execute in a fixed sequence, and thus we expect their success to be dependent on the success of prior jobs. To test this assumption, we examine the existence of dependencies between different error codes in Section 7.2. We further assess the predictive power of factors described in previous sections when forecasting future errors.

7.1 Auto-predictability of error codes

Using the periodic reports we receive for each backup system in our dataset, we can construct time series that reveal information about the timing of error occurrences. Specifically, we have information on the number of individual job error codes that occur in a given backup system, on a specific day. We use this data to construct two types of time series: a series of *error counts*, i.e. the number of times that a given error code occurs daily, and a series of *inter-arrival times*, i.e. the number of days between occurrences of the same error code. Across all systems, we analyze 55,862 error count time series, and 54,461 inter-arrival time series in total [1].

Most of the errors we study are misconfigurations that require administrator intervention in order to be resolved. We expect these errors to be recurrent, and therefore predictable. To test the predictability of our time series, we use the Hurst exponent. This metric is widely used as a measure of the long-term memory of time series [21, 32]. The values of the Hurst exponent, H, range between 0 and 1: $H = 0.5$ indicates a completely uncorrelated series, and $0 < H < 0.5$ corresponds to mean-reverting series, i.e. observations switching between low and high values, gravitating towards the mean. Finally, $0.5 < H < 1$ indicates trend-reinforcing series where future values are likely to be similar to past values. Due to this dependence, the latter category of time series are easier to predict.

Figure 11 shows the CDFs of Hurst values for both types of time series. The values have been estimated using the traditional R/S analysis approach [41]. We find that 90.1% of daily error count series exhibit strong trends ($H > 0.5$). This implies that the occurrences of a given error code tend to follow a trend, instead of varying wildly. The remaining time series score lower because they belong to hosts that have been monitored for shorter periods, and error counts exhibit low variation, if any. Specifically, the majority of series with Hurst exponents less than 0.5 show 1.9 times less variance, and consist of 6.3 times fewer observations. On the other hand, as many as 46.1% of error inter-arrival time series show low Hurst values ($H \leq 0.5$). The reason for this is that these time

[1] The smaller number of inter-arrival series is due to 1,401 error time series that span only two weeks. In those cases, inter-arrival series cannot be defined.

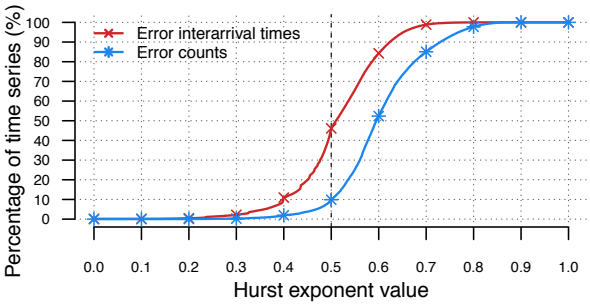

Figure 11: CDFs of Hurst exponent values for time series of error counts and inter-arrival times. Hurst values higher than 0.5 indicate persistent, predictable behavior.

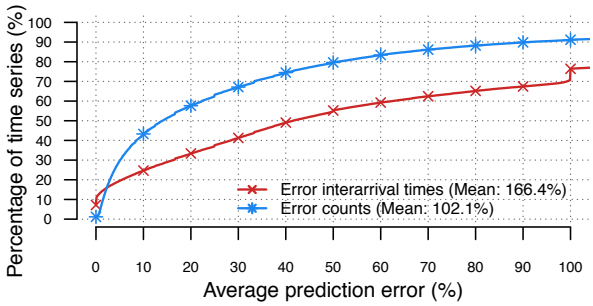

Figure 12: CDFs of the average prediction error of the best auto-regressive and exponential smoothing model fits for each time series.

series are 6.5 times shorter on average, as many errors occur in bursts. As a result, the number of inter-arrival times is smaller than the number of days we have data for, and the resulting trend consists of haphazard spikes.

Observation 11: *The number of daily occurrences of a given error code follows predictable trends, but error inter-arrival times do not.*

While Hurst exponent values reveal trends in time series, they cannot be used to suggest the right method to predict future occurrences. The most widely used forecasting methods for univariate time series are exponential smoothing, and auto-regression. *Exponential smoothing* models make predictions using weighted averages of past observations, with the weights decaying exponentially for older observations [9]. While these models try to capture the trend and seasonality in the data, *auto-regressive* models such as ARIMA [17, chapter 8.9] attempt to explicitly capture correlations with past observations.

For each time series, we estimated the best exponential smoothing and ARIMA model using Akaike's Information Criterion [2], a metric that rewards smaller prediction errors while penalizing models with more parameters. We consider a range of models, such as additive, multiplicative, and damped exponential smoothing. We trained each model using 75% of the available observations, and picked the model that scored the smallest average prediction error on the remaining test data.

In order to compare the prediction accuracy of models fitted to different datasets, we use the mean of the absolute prediction error percentage, a scale-independent error metric [17, chapter 2.5]. Since this metric divides the prediction error for a given observation by its value, it is sensitive to values approaching zero, and undefined for zero values. Therefore, we scaled all time series by adding the same constant prior to model fitting. In Figure 12, we show the CDFs of the average prediction error for each time series of error inter-arrival times, and error counts. A point (x,y) in the graph indicates that y% of time series predict future values with at most x% error.

We find that for the majority of time series, error inter-arrivals cannot be predicted with an error smaller than 42% on average. Error counts are more predictable, as their Hurst exponent values imply, with the majority of time series exhibiting less than 14% error in predictions. Both distributions are long-tailed, however, and predictions can exhibit errors as high as 2400% (or 166% on average) for inter-arrival series, and 886% (or 102% on average) for error counts. Since errors are expressed relative to true values, large errors are expected for short time series with high variability. In any case, these results indicate that job errors cannot be reliably predicted across many systems using only historical information on error occurrence. In the following subsection, we investigate the predictive power of other factors that affect the job error rate, as a means of increasing the prediction accuracy of our models.

Observation 12: *Forecasting methods that take into account only historical information of error occurrences fail to reliably predict future occurrences.*

7.2 Inter-error dependencies

Backup system policies define a fixed sequence of jobs that need to be scheduled before the assigned data has been backed up in a consistent manner. As a result, we expect the success of backup system jobs to be tied to the success of earlier jobs. To investigate whether dependencies exist between individual error codes in such a manner, we need to measure correlations between the time series of every possible error code pair in a system. Unfortunately, doing so manually is infeasible, because metrics that summarize cross-correlation as a number make assumptions about the data that must be verified through rigorous inspection of the time series involved. On the other hand, machine learning models can be used to discover patterns without being as restrictive.

We choose to model our data using random forests of decision trees for individual error codes, where the output for each forest determines whether an error code will occur on a given day, given inputs on the occurrence of other error codes on the same day, and in the last week. Random forests are groups of decision trees, each of which is built using a random subset of the input features and the training data. The output is decided by taking a simple majority vote over all the trees. Due to their construction, random forests are more robust against bias and over-fitting, compared to individual decision trees.

We trained individual random forests for every error code using daily data across all customer production systems, in order to detect trends that are independent of the administrative decisions made in individual systems. We use R's `randomForest` package [7], with 500 trees per forest, and we split our data so that 25% of the observations are reserved for testing the generated model. We measure model accuracy by computing the model's ROC curve [31], which is a function of the model's classification performance given its sensitivity, and then calculating the area under the curve (AUC) [12]. This number represents the model's ability to distinguish true positives without incurring additional false positives, as the sensitivity of the classification is varied. An accuracy of 50% represents the random classifier, while 100% indicates the perfect model that avoids false classifications entirely.

Overall, we find that random forest models trained using historical error data perform fairly well, achieving AUC values in the range of 70-83%, with a median of 76%. To find the features that contribute the most to the model's accuracy, the values of each feature are permuted, i.e. fuzzed, and the overall drop in the model's accuracy is measured. This approach helps us identify relationships between error codes. As a case in point, errors indicating that backups were rejected because the scheduling window closed are usually preceded by informational messages indicating that the administrator has dequeued or killed active jobs. Another example is that of errors indicating failures due to insufficient storage space, which are usually preceded by errors indicating that some, or no files have been backed up successfully.

While our results indicate clear dependencies, it is hard to imagine that the accuracy of the presented models will be sufficient to make predictions in production systems. To further test the validity of our observations, we extended our training data to contain features inspired from the observations made throughout this paper; the full list of features is shown in Table 4. Retraining our random forests with these extra features leads to consistently increased accuracy across all models. Specifically, the AUC values of these models lie in the range of 77-90%, with a median of 83%. We find that the most important feature in 44% of the retrained models is one of the features introduced in our earlier observations. In addition to that, 98% of the models contain at least two such features in their five most important features. It is worth noting that the features of highest significance across all models are the number of jobs, and the average number of parameters (i.e. complexity) of active policies.

Feature Description	Type
Occurrence of error code *i* today	Boolean
Past week occurrences of error code *i*	Numeric
Average number of jobs daily	Numeric
Occurrence of backup jobs today	Boolean
Occurrence of management jobs today	Boolean
Occurrence of recovery jobs today	Boolean
Average number of active policy parameters	Numeric
Average system job size	Numeric

Table 4: Features used in the training of the random forests. The first two feature types refer to all error codes except the one classified. The other features are specific to the system's operation and configuration.

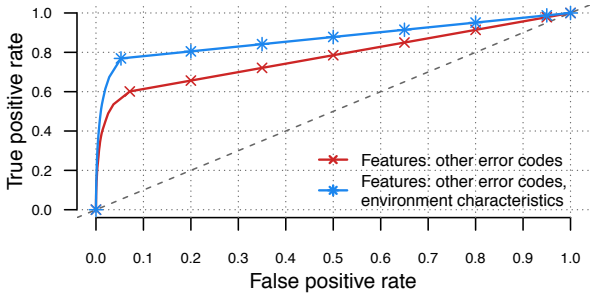

Figure 13: ROC curves for random forests predicting the occurrence of the error code indicating that a storage server is unreachable.

Figure 13 shows the ROC curves of two random forests built to predict the error code indicating that a storage server is unreachable to the backup system. The AUC for the model trained only using historical error data is 77.8%, and Figure 13 shows that it incurs a high rate of false positives once the true positive rate is adjusted above 60%. On the other hand, for the model taking into account factors from our study's observations, the AUC increases to 85.1% and a true positive rate of 80% is possible while retaining a low false positive rate.

Observation 13: *Most errors can be fairly accurately predicted based on the occurrence of other error codes. The factors in previous observations shown in Table 4, however, make superior predictors.*

8 Toward more resilient backup systems

This section outlines the implications of our results on the design of future backup systems. We identify four major topics with potential to increase the resilience of backup systems to job failures.

Error prediction. Very little work exists in the literature on the application of predictive models on backup systems. Chamness [10] and Vaughn et al. [36] use regression to forecast shortages in storage capacity. Ma et al. [22] use Naive Bayes to estimate the failure probability of hard disks, and RAID arrays. We have applied predictive models on job errors, and our results suggest that their occurrences follow predictable patterns. Our models, however, are constructed by simply applying well-known learning methods. We believe that more sophisticated approaches have the potential to yield higher levels of accuracy that would allow administrators to be proactive about errors. While we have trained our models to detect patterns that are universal across all systems in our dataset, prediction models used in individual systems could benefit from biases toward system characteristics.

Configuration automation. While self-healing systems automatically discover and correct faults, they do not improve the odds that they will not occur again in the future. A small body of prior work attempts to remedy this through system autoconfiguration. AutoBash [33] leverages a speculative OS kernel to fix misconfigurations. Chronus [42] makes use of checkpointing and rollback to detect the last working configuration. KarDo [20] takes a machine learning approach to learn the steps required to resolve the issue from a crowd-sourced collection of solutions, supporting heterogeneous environments as well. NetPrints [1] collects examples of good and bad network configurations, and generates a decision tree to determine the set of configuration changes required to transition the system to a good state. One of the challenges that these techniques face, is their timeliness in providing solutions. In some contexts, such as that of backup systems, allowing misconfigurations to remain latent can adversely affect the system's availability. Using the results from studies such as this one, these tools could improve their efficiency by narrowing down the list of potential errors. This filtering could be achieved based on the probability of occurrence for individual error codes, tracking the root cause faster.

Configuration validation. Research on misconfiguration errors mainly focuses on detecting, diagnosing, and troubleshooting these issues. While this provides a remedy after the fact, it does not spare users from the frustration of dealing with these errors in the first place. To address this issue, a line of research focuses on improving the design of configuration interfaces, and making systems more resilient in the face of misconfigurations. Some of the existing work includes the extraction of configuration parameter types [29], statically analyzing code to infer configuration variable constraints [43], permuting valid configuration settings [19], testing operator actions in sandboxes [24], storing application state to seamlessly undo and replay events in the case of errors [8], and using a declarative language to express configuration specifications [15]. Our findings in this study pinpoint configuration variables typically linked to error frequency or diversity in backup systems, and identify other factors as having no effect. Furthermore, our previous work [3] suggests that configuration parameters of these systems are often set to default values. We expect that using these observations as heuristics will assist future work in improving the efficiency of configuration correctness proofs.

Work reduction. One of the most common errors across systems indicates jobs being aborted because the scheduled window for backups was too short. While this error could be resolved by increasing the length of the backup window, it may not be feasible to do so. Today, capacities of hard drives are increasing faster than their data transfer speeds, causing near-line hard drives to take 1.7 hours to access a single terabyte of data [13]. To avoid these delays, it is worth looking into approaches that would allow the amount of maintenance work to be reduced. This could be achieved via content-aware backups that exclude temporary or unimportant files [14, 16], or by piggybacking on other ongoing I/O [4].

9 Conclusion

We have analyzed an extensive dataset from customer backup systems, consisting of 775 million jobs. We find that job errors are prevalent in backup systems, mostly due to misconfigurations. Fortunately, the errors that occur most frequently are not diverse in nature, with 10 error codes accounting for over 78% of job errors.

We identified factors that are responsible for job failures in backup systems, such as the job's type, size, and the complexity of the backup policy that issued the job. We also highlight factors with little effect on job failures, such as software characteristics, system size and load, and configuration variability. We show that the most influential factors make superior predictors for future failures compared to historical data on job errors.

We hope that our observations can be used as guidelines in the design of more robust backup software. We identify four promising directions for future work: error prediction, configuration automation and validation, and work reduction. We believe these features will be necessary in the face of modern data growth rates, which increase the time required to finish backups, leaving less time to rerun failed jobs.

Acknowledgments

The study would not be possible without the telemetry data collected by Veritas' NetBackup team. We especially thank Liam McNerney for his tireless assistance in understanding and extending the telemetry collection infrastructure. We also thank the four anonymous reviewers and our (recurring) shepherd, Fred Douglis, for their invaluable help in improving our paper. Finally, we would like to thank Bruce Montague, CW Hobbs, Ashwin Kayyoor, Vish Janakiraman, Henry Aloysius, Steve Vranyes, and all other members of Veritas Labs for their feedback during earlier stages of our study.

References

[1] AGGARWAL, B., BHAGWAN, R., DAS, T., ESWARAN, S., PADMANABHAN, V. N., AND VOELKER, G. M. NetPrints: Diagnosing Home Network Misconfigurations Using Shared Knowledge. In *Proc. of the USENIX Symposium on Networked Systems Design and Implementation* (2009), NSDI'09, pp. 349–364.

[2] AKAIKE, H. Information theory and an extension of the maximum likelihood principle. In *Proc. of the International Symposium on Information Theory* (1973).

[3] AMVROSIADIS, G., AND BHADKAMKAR, M. Identifying Trends in Enterprise Data Protection Systems. In *Proc. of USENIX Annual Technical Conference* (2015), ATC'15, pp. 151–164.

[4] AMVROSIADIS, G., BROWN, A. D., AND GOEL, A. Opportunistic Storage Maintenance. In *Proc. of the 25th Symposium on Operating Systems Principles* (2015), SOSP'15, pp. 457–473.

[5] ATTARIYAN, M., AND FLINN, J. Using Causality to Diagnose Configuration Bugs. In *Proc. of the USENIX Annual Technical Conference* (2008), ATC'08, pp. 281–286.

[6] ATTARIYAN, M., AND FLINN, J. Automating Configuration Troubleshooting with Dynamic Information Flow Analysis. In *Proc. of the USENIX Conference on Operating Systems Design and Implementation* (2010).

[7] BREIMAN, L., CUTLER, A., LIAW, A., AND WIENER, M. `randomForest`: Breiman and Cutler Random Forests for Classification and Regression. https://cran.r-project.org/package=randomForest, Oct. 2015.

[8] BROWN, A. B., AND PATTERSON, D. A. Undo for Operators: Building an Undoable e-Mail Store. In *Proc. of the USENIX Annual Technical Conference* (2003), ATC'03.

[9] BROWN, R. G. *Exponential Smoothing for Predicting Demand*. Arthur D. Little Inc., 1956.

[10] CHAMNESS, M. Capacity Forecasting in a Backup Storage Environment. In *Proc. of the International Conference on Large Installation System Administration* (2011).

[11] GRAY, J. Why Do Computers Stop And What Can Be Done About It?, 1985.

[12] HANLEY, J. A., AND MCNEIL, B. J. The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve. *Radiology 143*, 1 (Apr. 1982), 29–36.

[13] HETZLER, S., AND COUGHLIN, T. Touch Rate: A metric for analyzing storage system performance, 2015.

[14] HILDRUM, K., DOUGLIS, F., WOLF, J. L., YU, P. S., FLEISCHER, L., AND KATTA, A. Storage Optimization for Large-scale Distributed Stream-processing Systems. *Trans. Storage 3*, 4 (Feb. 2008), 5:1–5:28.

[15] HUANG, P., BOLOSKY, W. J., SINGH, A., AND ZHOU, Y. ConfValley: A Systematic Configuration Validation Framework for Cloud Services. In *Proc. of the ACM SIGOPS/EuroSys European Conference on Computer Systems* (2015), EuroSys '15.

[16] HUGHES, D., AND FARROW, R. Backup Strategies for Molecular Dynamics: An Interview with Doug Hughes. *Proc. USENIX ;login: 36*, 2 (Apr. 2011), 25–28.

[17] HYNDMAN, R. J., AND ATHANASOPOULOS, G. *Forecasting: Principles and Practice*. oTexts, 2015.

[18] IRON MOUNTAIN. Data Backup and Recovery Benchmark Report. http://www.ironmountain.com/Knowledge-Center/Reference-Library/View-by-Document-Type/White-Papers-Briefs/I/Iron-Mountain-Data-Backup-and-Recovery-Benchmark-Report.aspx, 2013.

[19] KELLER, L., UPADHYAYA, P., AND CANDEA, G. ConfErr: A tool for assessing resilience to human configuration errors. In *Proc. of the IEEE International Conference on Dependable Systems and Networks* (2008).

[20] KUSHMAN, N., AND KATABI, D. Enabling Configuration-independent Automation by Non-expert Users. In *Proc. of the USENIX Conference on Operating Systems Design and Implementation* (2010), OSDI'10.

[21] LELAND, W. E., TAQQU, M. S., WILLINGER, W., AND WILSON, D. V. On the Self-similar Nature of Ethernet Traffic. *IEEE/ACM Trans. Netw. 2*, 1 (Feb. 1994), 1–15.

[22] MA, A., DOUGLIS, F., LU, G., SAWYER, D., CHANDRA, S., AND HSU, W. RAIDShield: Characterizing, Monitoring, and Proactively Protecting Against Disk Failures. In *Proc. of the USENIX Conference on File and Storage Technologies* (2015), FAST'15, pp. 241–256.

[23] MASSEY JR., F. J. The Kolmogorov-Smirnov Test for Goodness of Fit. *Journal of the American Statistical Association 46*, 253 (1951), 68–78.

[24] NAGARAJA, K., OLIVEIRA, F., BIANCHINI, R., MARTIN, R. P., AND NGUYEN, T. D. Understanding and Dealing with Operator Mistakes in Internet Services. In *Proc. of the USENIX Symposium on Operating Systems Design and Implementation* (2004), OSDI'04.

[25] OPPENHEIMER, D., GANAPATHI, A., AND PATTERSON, D. A. Why Do Internet Services Fail, and What Can Be Done About It? In *Proc. of the USENIX Symposium on Internet Technologies and Systems* (2003), USITS'03.

[26] PARK, N., AND LILJA, D. J. Characterizing Datasets for Data Deduplication in Backup Applications. In *Proc. of the IEEE International Symposium on Workload Characterization* (2010), IISWC'10.

[27] PATTERSON, D., BROWN, A., BROADWELL, P., CANDEA, G., CHEN, M., CUTLER, J., ENRIQUEZ, P., FOX, A., KICIMAN, E., MERZBACHER, M., OPPENHEIMER, D., SASTRY, N., TETZLAFF, W., TRAUPMAN, J., AND TREUHAFT, N. Recovery Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies. Tech. Rep. UCB/CSD-02-1175, EECS Department, University of California, Berkeley, Mar. 2002.

[28] R DOCUMENTATION. Fitting linear models. https://stat.ethz.ch/R-manual/R-devel/library/stats/html/lm.html.

[29] RABKIN, A., AND KATZ, R. Static Extraction of Program Configuration Options. In *Proc. of the International Conference on Software Engineering* (2011), ICSE'11.

[30] RAUSAND, M., AND HØYLAND, A. *System Reliability Theory: Models, Statistical Methods and Applications (Second Edition)*. Wiley-Interscience, 2003.

[31] Receiver operating characteristic. https://en.wikipedia.org/wiki/Receiver_operating_characteristic.

[32] RISKA, A., AND RIEDEL, E. Disk drive level workload characterization. In *Proc. of the USENIX Annual Technical Conference* (2006), ATC'06.

[33] SU, Y.-Y., ATTARIYAN, M., AND FLINN, J. AutoBash: Improving Configuration Management with Operating System Causality Analysis. In *Proc. of the 21st ACM Symposium on Operating Systems Principles* (2007), SOSP'07, pp. 237–250.

[34] TANG, C., KOOBURAT, T., VENKATACHALAM, P., CHANDER, A., WEN, Z., NARAYANAN, A., DOWELL, P., AND KARL, R. Holistic Configuration Management at Facebook. In *Proc. of the 25th Symposium on Operating Systems Principles* (2015), SOSP'15, pp. 328–343.

[35] VANSON BOURNE. Virtualization Data Protection Report 2013 – SMB edition. http://www.dabcc.com/documentlibrary/file/virtualization-data-protection-report-smb-2013.pdf, 2013.

[36] VAUGHN, C., MILLER, C., EKENTA, O., SUN, H., BHADKAMKAR, M., EFSTATHOPOULOS, P., AND KARDES, E. Soothsayer: Predicting Capacity Usage in Backup Storage Systems. In *Proc. of the IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems* (Oct. 2015), MASCOTS'15, pp. 208–217.

[37] VERITAS TECHNOLOGIES. Veritas NetBackup 7.7. https://www.veritas.com/product/backup-and-recovery/netbackup, Sept. 2015.

[38] VERITAS TECHNOLOGIES. Veritas NetBackup Status Codes Reference Guide (Release 7.6). https://www.veritas.com/support/en_US/article.DOC6471, Sept 2015.

[39] WALLACE, G., DOUGLIS, F., QIAN, H., SHILANE, P., SMALDONE, S., CHAMNESS, M., AND HSU, W. Characteristics of Backup Workloads in Production Systems. In *Proc. of the USENIX Conference on File and Storage Technologies* (2012), FAST'12.

[40] WANG, H. J., PLATT, J. C., CHEN, Y., ZHANG, R., AND WANG, Y.-M. Automatic Misconfiguration Troubleshooting with PeerPressure. In *Proc. of the USENIX Symposium on Operating Systems Design and Implementation* (2004), OSDI'04.

[41] WERON, R. Estimating long-range dependence: finite sample properties and confidence intervals. *Physica A Statistical Mechanics and its Applications 312* (Sept. 2002), 285–299.

[42] WHITAKER, A., COX, R. S., AND GRIBBLE, S. D. Configuration Debugging As Search: Finding the Needle in the Haystack. In *Proc. of the USENIX Symposium on Opearting Systems Design and Implementation* (2004), OSDI'04.

[43] XU, T., ZHANG, J., HUANG, P., ZHENG, J., SHENG, T., YUAN, D., ZHOU, Y., AND PASUPATHY, S. Do Not Blame Users for Misconfigurations. In *Proc. of the 24th ACM Symposium on Operating Systems Principles* (2013), SOSP'13, pp. 244–259.

[44] YIN, Z., MA, X., ZHENG, J., ZHOU, Y., BAIRAVASUNDARAM, L. N., AND PASUPATHY, S. An empirical study on configuration errors in commercial and open source systems. In *Proc. of the 23th Symposium on Operating Systems Principles* (2011), SOSP'11, pp. 159–172.

[45] YUAN, C., LAO, N., WEN, J.-R., LI, J., ZHANG, Z., WANG, Y.-M., AND MA, W.-Y. Automated Known Problem Diagnosis with Event Traces. In *Proc. of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems* (2006), EuroSys'06, pp. 375–388.

[46] YUAN, D., XIE, Y., PANIGRAHY, R., YANG, J., VERBOWSKI, C., AND KUMAR, A. Context-based Online Configuration-error Detection. In *Proc. of the USENIX Annual Technical Conference* (2011), ATC'11.

[47] ZHANG, J., RENGANARAYANA, L., ZHANG, X., GE, N., BALA, V., XU, T., AND ZHOU, Y. EnCore: Exploiting System Environment and Correlation Information for Misconfiguration Detection. In *Proc. of the International Conference on Architectural Support for Programming Languages and Operating Systems* (2014), ASPLOS'14.