UNIVERSITY OF CALIFORNIA

SANTA CRUZ

TRACE ANALYSIS OF LARGE-SCALE STORAGE SYSTEMS

A report submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

COMPUTER SCIENCE

by

Devashish Purandare

March 2019

The report of Devashish Purandare
is approved:

Professor Ethan Miller, Chair

Professor Peter Alvaro

Dean Alexander Wolf
Dean of the Baskin School of Engineering

Contents

List of Figures

List of Tables

Abstract

Trace Analysis of Large-Scale Storage Systems

by

Devashish Purandare

Storage systems for scientific and industrial workloads involve working with petabytes of data. These systems often have complex hierarchies of different types of storage media through which data movement takes place. It is important to understand the behavior of such a system, including migration, replication, per user read/write patterns, per task usage, as well as trends over longer periods of time, such as a month or a year. Such analysis will help us identify the system usage, reduce redundant reads and writes. Trace analysis also allows us to identify and differentiate between recurring tasks, related tasks, and schedule them with necessary priorities, to improve throughput and reduce latency.

We present our uniform trace analysis framework, which is designed to take in traces across multiple large scale systems, and compare the behavior of the archives over time. The system can take in data across multiple formats, and present a 1:1 comparison of attributes as well as usage across systems. We present an analysis of the CERN EOS filesystem traces, traces gathered from CERN's production system over a year. The analysis is across 2.49 billion unique events that happened on the EOS filesystem. We plan to integrate this trace analysis with traces from other scientific labs and archives, to compare and contrast behavior of large scale storage systems.

Acknowledgments

This project was made possible due to the generous support provided by the Center for Research in Storage Systems (CRSS). The European Organization for Nuclear Research (CERN) made shared their EOS file system traces, something that was vital for this project. I am very thankful for the help and advice I have received from Dr. Dirk Duellmann about the traces. Matt Bryson, another grad student with the SSRC is responsible for some of the early work which this project expands on.

This project expands on the work of Ian Adams, Brian Madden, Joel Frank and other past SSRC students.

I would also like to thank Dr. Peter Alvaro and Dr. Ethan Miller for their help and guidance during this project. Special thanks to the grad students in the CRSS research group and the University of California, Santa Cruz.

Chapter 1

Introduction

Modern scientific and archival storage systems capture, store, and process large amounts of data. They often have custom designs built to optimize for a specific characteristic of the data. These custom designs are influenced by the expected workload, making it challenging to make generalized observations on large scale systems. Capturing system traces which document the flow of the system in work from state to state helps us analyze how the system worked over the period of time. Traces record actions performed by a system and sometimes the state of the system after such an action. These kinds of system traces allow us to do a variety of tasks, such as analyzing how the system works, hunting down bugs and to come up with methods for optimizing the system and making it more efficient.

Scientific storage systems generally have multi-tiered storage hierarchy [15, 2, 9, 11].

Generally, the cold captured data is stored in a large capacity, long term storage archive, while the data being actively worked on is on a faster but lower capacity tier. This model carefully navigates the trade-off of storage space against access speed, however in doing so ends up in complex tiered architectures and significant data movement. Such systems can be hard to reason about and difficult to optimize without understanding how the system is executing the ideas it was built with. Tracing such a system offers us deep insight into the performance of the system, the type of workloads which can help us come up with optimizations. For example, knowing the probability of access of a certain object can help us decide if we want it to be cached on the faster storage medium.

Trace analysis has been an area of interest in supercomputing and systems research over the last few decades. As machine learning and deep learning frameworks expand into systems, system traces can be transformed into training data for deep learning frameworks to pick the placement of the data for fastest access [14].

This work also uses the NCAR data from Adams et al., in 2012 [2] to compare and contrast scientific archival systems, which has not been done in large systems before. We present analysis which is not influenced by design of a particular system, but looks at characteristics shared by many large-scale systems.

In this work:

- We present a comparative analysis of file system traces from the European Organization for Nuclear Research (CERN) and the National Center for Atmospheric Research (NCAR).

- Our analysis compares and contrasts the :
  1. Nature of the workload
  2. The trace capturing mechanism
- We looked at the current practices from the supercomputing community and whether they hold up for these systems.
- We present best practices for capturing traces based on our experience analyzing various trace formats.
- We motivate a framework which will be able to do automated analysis on a wide variety of traces.

The layout of this report is as follows: In the next chapter we will talk about the various trace analysis research done in the past, and what kind of insights it can give us. We then talk about the structure and layout of the CERN and NCAR system, the system architecture and the trace formats. We briefly describe our processing system and present our observations from the trace data. We compare our observations to the conventional wisdom in supercomputing and analyze the core causes. This is followed by meta analysis on trace capture formats as well as a need for a uniform framework.

Chapter 2

Related Work

Trace analysis has been an area of interest in systems research for decades. Jensen and Reed [12] looked at file archive activity at the National Center of Supercomputing Applications in 1993. In the same year Miller and Katz [15] performed an analysis on file migration in the NCAR environment. The analysis from these publications provides a great insight into large scale archives in the 1990s. While the scale of files, accesses and file sizes has shifted, many of the observations about large systems from these works are still observed today.

There have been several studies on analyzing storage changes over long periods of time. Agrawal et al. [5] looked at changing file system metadata. Breslau et al. [6] looked at web caching logs and presented a perspective on how Zipf's law applies to storage. We observe Zipfian distributions in our results as well. Leung et al. [13]

looked at a large scale industrial network based file system.

We discuss the paradigm of "Write Once, Read Maybe" in Chapter 6 there have been multiple designs which address power or access efficiency optimizations for such a model. Grawinkel et al. [10] describe Lonestar system that does aggressive power optimizations assuming such a pattern of reads and writes. Colarelli et al. [7] demonstrated the Massive Array of Idle Disks (MAID) framework to reduce power consumption in a write once, read maybe model. Pergamum [17] was an optimization over MAID suggested by Storer et al. which added NVRAM at each of the idle disks to provide high performance. These inspired the design of Internet Archive [11] by Jaffe et al.

Adams et al. [2, 3, 4] presented an analysis on NCAR MSS traces from 2008–2010. We utilize the same traces in our analysis. A lot of this work is based on that original analysis, we compare CERN traces that we have to that workload. Grawinkel et al. [9] presented an analysis on the ECMWF storage system. This analysis is comparable in size and scale. It comes from a different domain, the weather forecasting data from ECMWF, as opposed to the particle capture data from CERN.

We expand on the original NCAR work and compare it with the CERN traces. With 2.49 billion such data points, these traces capture more data and this analysis is over a bigger dataset than what has been done before. We plan to do more 1-1 comparisons in the future with similar systems as it will help us extract the larger trends which are not influenced by individual designs of such systems. We also reflect on the differences between trace capture and the format of trace capture and present our observations about the system.

Chapter 3

Background

The European Organization for Nuclear Research (CERN) was established in 1954 as a joint research initiative between member states in the area of particle physics. Current membership of CERN is at 22 member states and two nominated members. CERN is one of the largest scientific labs in terms of international collaboration. CERN captures scientific data from a range of particle accelerators, most popular of which is the Large Hadron Collider. As of 2017, CERN had 230 petabytes of permanent storage on magnetic tape, 70 petabytes of which was captured in the same year [1]. To process and store this data, CERN uses a custom developed file system, known as the CERN EOS file system [16].

3.1 The CERN storage system

CERN uses a three tiered storage system,

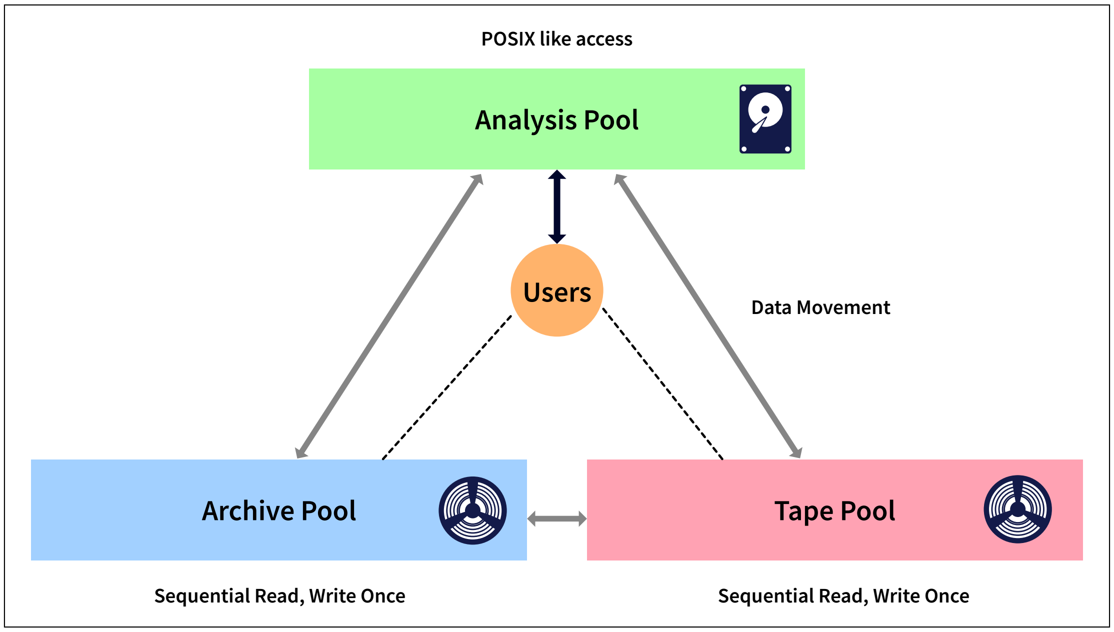

Figure 3.1: The multi-tiered CERN storage system

3.1.1 Analysis Pool

Analysis pool is the only storage users have direct access to, they have POSIX like access, and this is where all of the data analysis happens. Analysis pool can be up to a petabyte and is made from spinning hard disks. Analysis pool is where data is fetched to, from the archival and tape pools and processed. Updates to data are only possible in the analysis pool.

3.1.2 Archive Pool

Archive pool is magnetic tape based storage, which only allows sequential read and write once semantics. Archive pool isn't open to users for access, and has privileged access. Archive pool is medium latency, which can be a few milliseconds to a second. Archive pool stores data at file granularity and the two actions possible are to get the file and store the file. Updates are not supported.

3.1.3 Tape Pool

The tape pool is high density magnetic tape based storage that puts batches of files into containers and supports sequential read, write once semantics. Fetching data from tape pool has a latency of 10^1 - 10^3 seconds. Tape pool is mainly used to store cold data which is not in use. Users do not have direct access to the tape pool.

While the disk pool can go up to a few petabytes, the tape pools can go all the way up an exabyte. These could be scaled in case of increased requirement. CERN utilizes these three tiers of storage for different activities. Data actively being processed in pulled into the analysis pool and users have direct random read/write access to the data stored on this pool. Getting data in the analysis pool requires physically fetching the tape, reading it sequentially and copying data. This incurs a significant penalty in access time and significant movement of data.

3.2 CERN EOS Traces

The EOS file system is shared by 4 experiments at CERN, namely Atlas, Alice, CMS and LHCB. From our collaboration with CERN, we acquired traces from the Compact Muon Solenoid (CMS) experiment. The traces document all system activity over 326 days, ranging from 13th October 2016 – 3rd September 2017.

Some facts about the system :

Range	326 days
Records	2.49 billion
Files	188 million
Users	1977
Data Moved	95 petabytes

Table 3.1: Overview of CERN trace data

The traces are available in the Apache Parquet format, compressed using gzip compression, spread across individual files separated by days. The total size of traces is 133 gigabytes compressed. These traces, unlike the NCAR traces, capture the state of the file after an operation occurs on it. While we can infer the operation based on some of this information, it is not explicitly recorded. Actions other than simple read and write, can present difficulties. File creation is hard to detect, and since we have a snapshot over a period of time, it can be hard to distinguish files that existed before the snapshot over the ones that were created.

3.3 The NCAR storage system

The US National Center for Atmospheric Research (NCAR) is an atmospheric science research institution managed by University Corporation for Atmospheric Research (UCAR). NCAR has a custom designed storage system known as the NCAR Mass Storage System (MSS). We look at how the MSS operated between 2008 and 2010 and analyze the traces. At the end of the trace capture, NCAR had 11.7 petabytes of stored data [2].

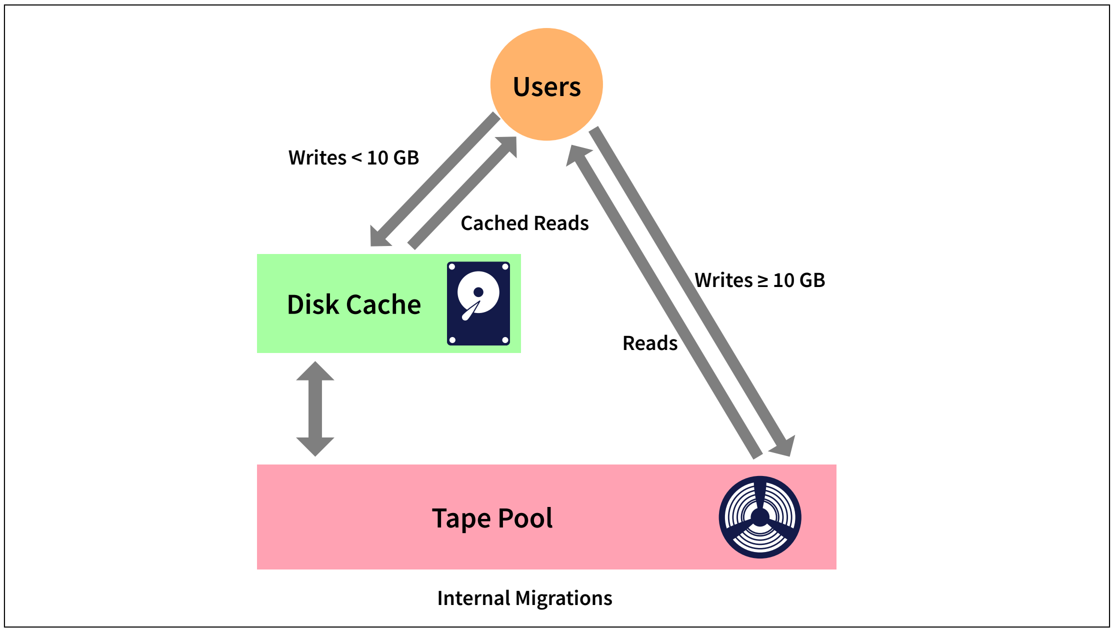

Figure 3.2: The multi-tiered NCAR storage system

In the NCAR MSS, files above 10 gigabytes are written directly to tape, while smaller files are written to an intermediate disk cache which batches updates to the tape. If a file is read more than once in 24 hours, it is cached in the disk based cache. The system checks the cache and gets the file from the cache if it exists, else reads are made directly from the tape. Internal load balancing and migration also occurs in the tape archive.

Some facts about the system :

Range	2008 – 2010
Records	188 million
Files	69 million
Data Moved	11.7 petabytes

Table 3.2: Overview of NCAR trace data

The NCAR traces span a 3 year period from 2008 – 2010.

3.4 The NCAR MSS Traces

The traces span 3 years of work at NCAR. Unlike CERN traces, they capture specific actions such as MIGRATE or CREATE rather than the state of files. As seen in table 3.2, they capture 188 million unique events dealing with 69 million unique files. The traces are stored in Tab Separated Value (TSV) files in multiple formats. They are stored compressed across 1097 individual files taking up 31 gigabytes of storage.

In the next chapter, we will go over the processing methodology and setup for these traces.

Chapter 4

System Setup for Analysis

Since the CERN and NCAR traces are compressed text files, a variety of tools were used to aid processing of this data. The tools are across various categories including preparing the data, processing, analysis and finally plotting graphs.

4.1 Preprocessing and Preparing the data

System traces from different systems often use different formats, different file layouts, and different compression schemes. This makes it hard to process them without having the exact tools that they were intended to be processed with. For instance, CERN traces are stored as Apache Parquet [19] files, and were intended to be processed with Apache Spark. At their core however, traces are just compressed text files, and a variety of unix utilities provide an easy way to transform them.

GNU `parallel` [18] is a unix tool which can perform operations on batches of files in parallel. It gives your script or program an individual thread for each file it needs to work on, and speeds up processing on a multi-core system. This is particularly helpful for trace data which is split across hundreds of files. `grep`, `sed` and `awk` were used at various points to do format alterations and get a uniform format.

4.2 Analysis Tools

After extracting a consistent format from the traces, it is processed either using a custom thread-pool golang framework or Apache Spark.

4.2.1 Thread Pool Framework

As seen in figure 4.1, the query and analysis system implemented in golang assigns each trace file to a thread in the pool, and performs the specified λ operation on it. Golang natively allows to pass functions as arguments and gives a great lightweight threading environment with its "goroutines" which makes this easy to implement.

4.2.2 Apache Spark

Apache Spark [20], based on the Map-Reduce [8] framework, is designed for large-scale data processing. It distributes the files on its cluster with mapper nodes that

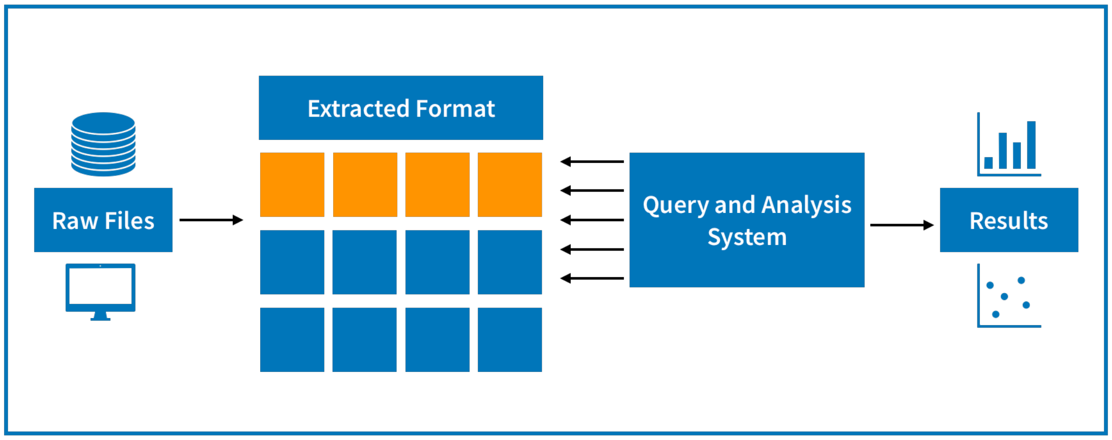

Figure 4.1: Thread pool implemented in the Go programming language

transform data and reducer nodes that collect and aggregate data. Spark supports a subset of SQL, which can run SQL queries directly across trace files, which this analysis made extensive use of. The GROUPBY feature of SQL in particular is indispensable for any analysis which needs several aggregations based on different fields.

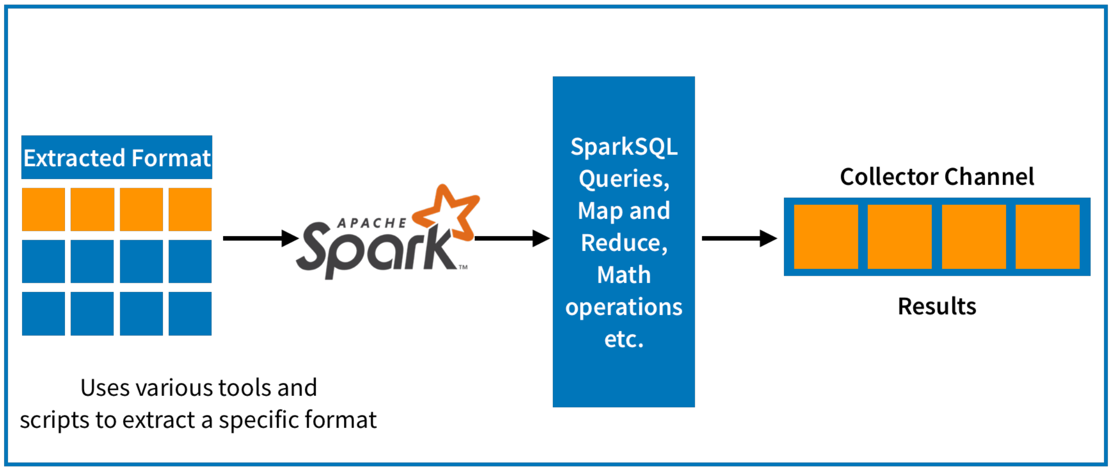
Extracted Format
APACHE Spark
SparkSQL Queries, Map and Reduce, Math operations etc.
Collector Channel
Results
Uses various tools and scripts to extract a specific format

Figure 4.2: Processing with Apache Spark

Chapter 5

Observations from the data

The CERN trace data is extensive, capturing 2.49 billion unique actions and spanning over 11 months. We performed different kinds on analysis on it, and have a wide variety of observations. We have broken down the results into four main categories :

5.1 General Statistics

While the CERN system traces span a third of the timespan of NCAR traces, they capture a much larger system. CERN traces were captured in 2016, as compared to 2008 in case of NCAR traces, this helps us observe trends in supercomputing while normalizing the differences in systems 8 years apart.

	CERN	NCAR
Events	2.49 B	188 M
Unique Files	188 M	69 M
Data Stored	95 PB	11.7 PB
Timespan	326 days	3 years

Table 5.1: CERN and NCAR trace statistics

5.2 Files

We use the unique `fid` identifier to identify unique files. Due to the nature of the traces, we consider the file size to be the last recorded size in the traces, which in most cases is also the maximum recorded size for that file.

5.2.1 File Size Distribution

As seen in fig 5.1, 98% of all files stored are below 4 gigabytes, but make up only about 40% of the storage space that is utilized. The CDF curve gets steeper when we consider the volume, files between 1 gigabyte and 32 gigabytes make up about 78% of the total volume, with twelve 256 terabyte files making up the rest. These 12 files make up about 22% of the storage volume. These files are snapshots of virtual machines, and would greatly benefit from having a dedicated caching strategy.

Compared to similar analysis on NCAR [2] we observe that though the overall trend

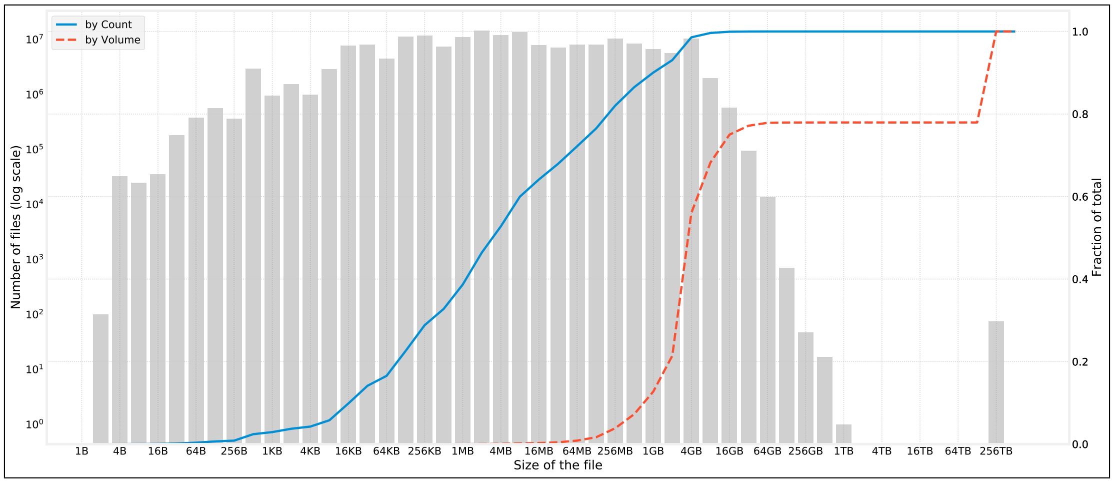

Figure 5.1: The Cumulative Distribution Function (CDF) and histogram for file size distribution in CERN EOS system

is similar, larger files occupy more volume in CERN as opposed to NCAR. Some of this can be attributed to the fact that these sets are years apart, and CERN files about collision data tend to be a couple of orders of magnitude larger than atmospheric data captured by NCAR.

5.2.2 File and Directory Depths

The file depth histogram for CERN, again, differs significantly from what we observed in NCAR. There are a large number of temporary files which are created at root during the data movement, and overall, depths 9 and 13 have the most files. These depths represent the ftp mount and user directories respectively. However in order to see how data layout corresponds to depth at which it resides, we plot the CDF of files at a particular depth vs volume of data.

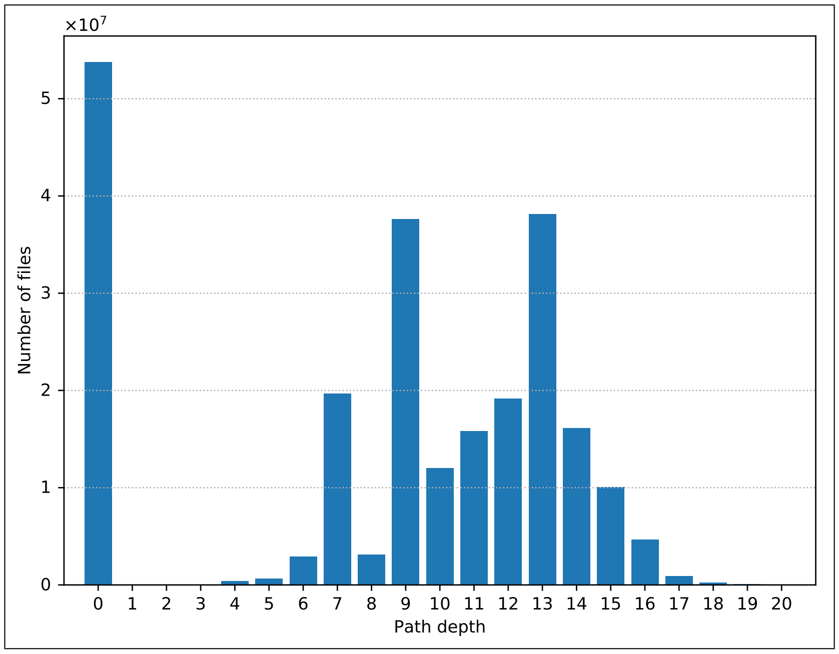

Figure 5.2: Histogram of file depth in CERN EOS system

The access for most scientific data follows the path `/eos/cms/store/group/<group name>/<experiment name>` followed by a custom directory structure depending on the experiment.

In fig 5.3 we look at the fraction of files at a particular depth compared to the volume they occupy at that depth. In the CERN EOS system, files stored up to depth 9, make up about 30% of the files and 30% of the data. Files with depth greater than 9 tend to be larger, and their share of volume is higher than their share of the count. Files between depths 0–8 make up a greater percentage of count rather than volume, which means there are smaller but more numerous files at these depths.

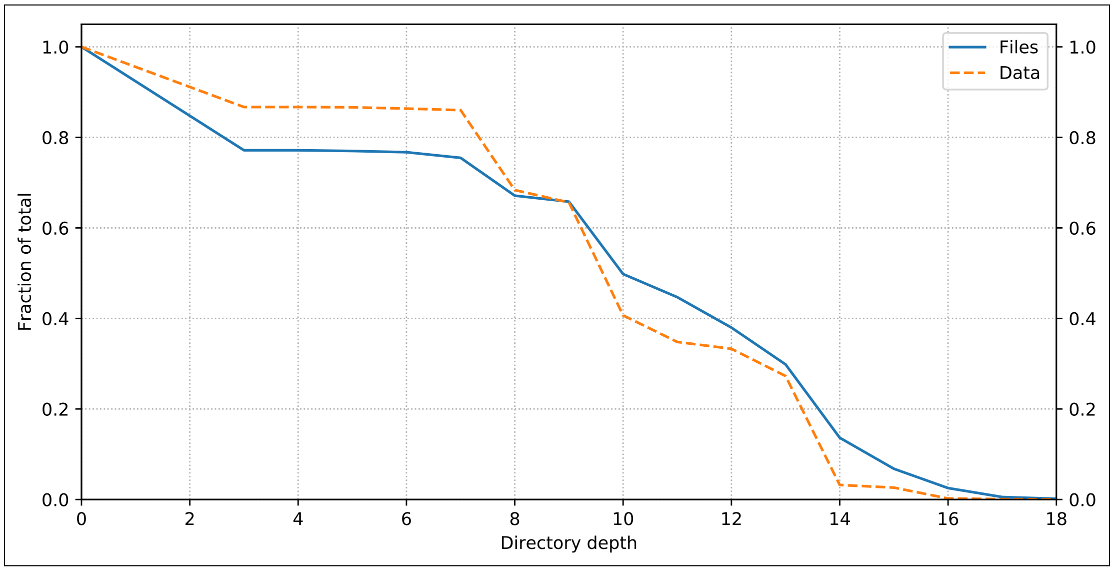

Figure 5.3: The Cumulative Distribution Function (CDF) for files at a particular depth in the filesystem hierarchy, along with the volume at the depth (CERN).

Connecting this to the depths we know about, we can see a steep curve at depth 13, where the user directories are, and they make up about 25% of the volume. Similarly the crossover point into large files is at depth 9, which hosts the FTP mount.

Overall this is different from what was observed in the NCAR system, where volume always leads the share with respect to count. In NCAR, most of the data was concentrated at a single depths, unlike the multiple depths we see in the CERN dataset. This stems from the fact that the CERN data store is shared by a variety of experiments with custom directory conventions, while NCAR uses a more uniform approach. This tells us at which depths files are more likely to be stored and accessed, and the path depth can help us classify what kind of file it is.

Due to the nature of trace files, it is difficult to accurately infer what is the state of directory depths at different levels.

5.3 Read/Write Activity

In our traces, we observe the existence of 188 million files, of which we observe creation of 139 million files. While CERN does not specify the type of action, we specify creations as a `fid` starting at size (`osize`) 0 and the operation doing a write (written bytes `wb`) that is positive.

- We observe the creation (first write) of 139 million files, out of which 0.22% or 314,000 are written-to a second time.
- Less than 0.09% files are written to ever again.
- The most actively written file has 81,000 writes to it.
- For reads, the pattern is significantly different, we see 85 million reads, of which 42% are read at least 2 times.
- 12.5% of those are written to three times or more

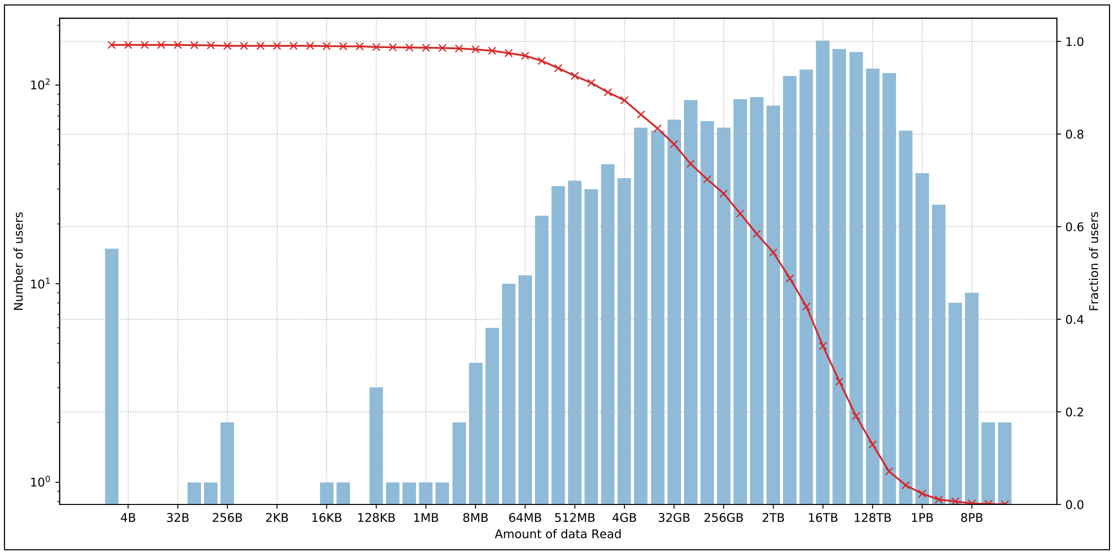

Figure 5.4: The Cumulative Distribution Function (CDF) and histogram for the amount of data read by users (CERN).

5.3.1 Reads

We observe that the total amount of data read over the 326 day period is 300 petabytes. Almost all users perform reads. At least half the users read more than 4 terabytes of data and as seen in fig 5.4 more than 80% of the users read at least 16 gigabytes of data.

Type	**Top reader/writer**	**Top 5**	**Top 5%**
Read	42%	62%	89%
Written	27%	95%	99%

Table 5.2: Amount of data Read/Written by Users

As seen in table 5.2, the top reader read 43 petabytes, performing 42% of all reads. The top 5% users who make the most reads make up about 89% of all data that was read in the system. This however is much sharper in case of writes.

5.3.2 Writes

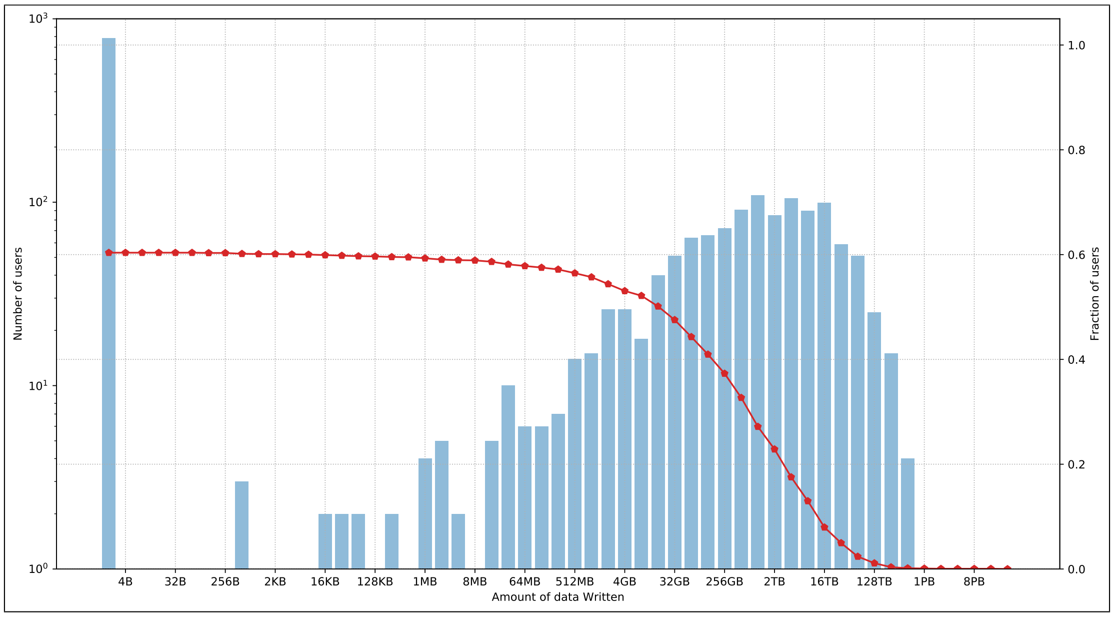

Figure 5.5: The Cumulative Distribution Function (CDF) and histogram for the amount of data written by users (CERN).

Right off the bat, we can see significant differences between fig 5.4 and fig 5.5. As many as 782 users, which make up almost 40% of all users, perform no writes to the system. Of those who do write data to the system, at least half of them write more than 16 gigabytes to the system. The writes associated per user are much smaller than the reads. Which is not surprising for a scientific data archive, where a lot of the captured data is refined and aggregated to produce results.

Coincidentally, the process mirrors the one used to write this document, as hundreds of gigabytes of trace data is processed into observations and plots that fit into megabytes worth of storage.

Writes are performed by a small subset of users, which is very clear from table 5.2. The top writer wrote 131 petabytes of data, and was responsible for 42% of all data written. The top 5 writers wrote as much as 95% of all the data by volume, and almost all writes were made by the top 5% of writers. Most of the users that write large amounts of data are system users that handle data movement and copying. This can be inferred from the user IDs. UID 0 is the root users, system users have UIDs below 100. It is important to therefore break this usage down into a user and system-user comparison.

5.4 Users

	Actions	**Written**	**Read**	**Affected Files**
System	204,756,128	26.03 PB	26.05 PB	38,609,850
User	2,214,873,512	131.4 PB	285.2 PB	119,920,321

Table 5.3: Breakdown of reads and writes between user actions and system usage

When we break down the tasks by the user accounts (`uid`) performing the task, we see that system accounts perform about 9% of all actions. They read about 26 petabytes, and write about the same. Since system tasks are mostly about load balancing, these numbers are very similar. We can further break down system tasks as seen in table 5.4.

5.4.1 System

Application	Actions	Written	Read	Affected Files
archive	4	0 B	70.56 GB	2
converter	500	0 B	0 B	241
eos/balancing	154,387,709	20.24 PB	20.24 PB	38,609,850
eos/draining	50,238,334	5.74 PB	5.74 PB	22,725,214
eos/replication	129,581	60.87 TB	72.83 TB	27,132

Table 5.4: System tasks

System user accounts perform maintenance tasks such as balancing and replication. While they perform only 9% of all actions, they write about 16% of all data written. System writes are on an average significantly larger than user writes. While system tasks cause a lot of data movement, the majority of reads, writes are performed by individual user accounts.

Application	Actions	Written	Read	Affected Files
eos/gridftp	92,130,333	33.5 PB	45.75 PB	46,280,644
eoscp	12,618,810	376.38 TB	270.3 TB	3,825,686
filecopy	174	0 B	0 B	89
fuse	200,572,627	0.49 PB	2.88 PB	19,796,087
other	1,894,603,867	85.24 PB	229 PB	119,920,321
point5	5,432,691	7.4 PB	0 B	2,787,447
restore	13,122	15.65 GB	0	6,304
tpc	9,501,888	4.32 PB	7.24 PB	6,682,565

Table 5.5: User applications

5.4.2 User Applications

Looking at the data usage by application across multiple users, we see a lot of applications associated with moving data. `eoscp`, `filecopy` are used to move data within the file system, `eos/gridftp` is the file transfer protocol mount used for moving data, and performs a significant amount of writes (more than all system tasks) and reads. `fuse` is a popular userspace file system application. Here, `point5` refers to all the tasks related to the Point5 endpoint of the Large Hadron Collider. `tpc` is the third party copy tool, used by researchers outside of CERN to access CERN data.

However, as we can see, a majority of user reads and writes are not tagged by a particular application type. Categorized under other, these are various tools outside of those discussed above to process and transfer data.

In the next chapter, we compare these observations with what we have seen across large system analysis literature, and compare if conventional wisdom holds up to current systems.

Chapter 6

Conventional Wisdom vs Modern Systems

As we saw in chapter 2, trace analysis is a common technique used in several studies of large-scale systems. These studies worked on formulating some workload observations which are observed in such system. In this chapter, we compare conventional wisdom, to observations from the CERN data. Most of these observations are quoted from the original work on the same NCAR set [2].

6.1 "Write Once, Read Maybe"

"Write Once, Read Maybe" is a common assumption that archival systems make. Most archival data is written once, rarely updated, and may not be read at all. This rule allows us to make many optimizations, such as not allowing updates to files. This can help us make files immutable, offering many benefits especially in indexing and compression, however this requires larger writes in case the file needs to be updated, since it is rewritten.

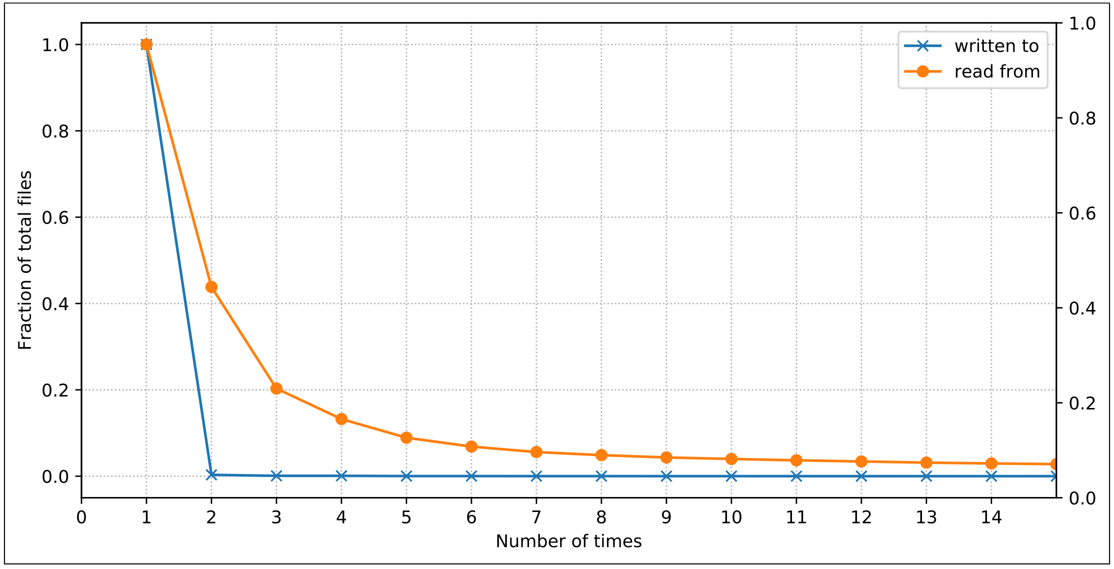

Figure 6.1: CERN : Likelihood of reads and writes to a file

As seen in fig 6.1 this assumption is valid and useful in the CERN file system traces. The chance that a file will be written to a second time is 0.22%, going down to 0.09% for a third write. The read graph on the other hand shows that while most files are read once, a significant number of files are read a second and a third time, and follow a heavy-tailed distribution.

6.2 "Automated migrations make up majority of the activity"

In NCAR MSS traces, it was observed that system actions made up 66% of all actions observed in the traces. Most of these were automated migrations. This pattern has been observed in several archival systems.

In CERN traces however, we did not observe this pattern. While majority of actions are related to migration of data, many of these actions are user initiated. CERN archive also differs from traditional archives, as it is an active processing system, and the traces we analyzed are from the disk cache where data is fetched and processed. As we saw in table 5.3 only 9% actions in the CERN system are automated, 91% are user driven.

6.3 "Most actions come from a small subset of users"

This is observed in CERN data as well, as observed in table 5.2 top 5% of readers/writes perform almost all of the writes and about 89% of all reads.

6.4 "Most files are at the same directory depth"

We did not find support for this observation in the CERN dataset. While most files are concentrated at depths 0, 9, and 13, files are distributed at a variety of depths as seen in fig 5.2.

Chapter 7

Future Work

We discussed our observations from the CERN and NCAR traces spanning hundreds of days. However, this analysis has focused on comparing the CERN system to other systems which were analyzed before. The CERN trace data is an extensive documentation of a living system over 11 months and there are additional details in the traces that are not discussed here.

7.1 Additional Analysis

Additional details in the traces include details about the physical devices on which data is stored, which opens us up to new metrics such as hardware failure rates and performance of disks over time. We have access to additional information such as read calls and write calls to the OS, along with information on forward seeks and backward seeks which can be helpful in tuning the system for performance and observing the access patterns.

The traces further tag access by security roles of projects, users, and organizations. This will be valuable information for training deep learning frameworks for optimizing placement and access of data. There are some experimental details available such as vector reads and writes which are in need of further exploration.

7.2 Uniform Framework for Trace Capture and Analysis

This project involved working with different trace formats, which captured system state differently. Yet, there is common information that can be inferred from these differing formats, which is useful for comparison and contrast. Going forward, we would like to formalize this project into a modular framework for trace analysis, where a variety of formats of traces can be plugged in and with a few custom transformation rules, this analysis can be automated. The tools that work on the processing part : both our golang framework and spark expect a uniform format over raw files. In the future we plan build tools which unify such a format.

Such a framework would have a common set of parameters, which are shared by most trace capture mechanisms, and will allow custom transformations to include their values. For example, if a trace captures read speed, and another captures amount of data read and the time takes, you could specify the read speed using a mathematical operation on these separate fields. A language like `tcl` which allows expressing operations as expressions would be well suited for this purpose.

7.3 Testing Cache and Placement Algorithms

As seen in the work done by Garwinkel et. al. [9], traces can be used as a test dataset for a variety of cache algorithms. With the emergence of deep learning techniques to decide file placement [14], we plan to use this data to test out automated optimization strategies for large scale system.

Chapter 8

Conclusions

This work analyzed trace data from the CERN EOS file system over a period of 326 days. We analyze system and user behavior over the period and compare it to past analysis. We have a set of observations from the trace analysis that we performed which we believe will be helpful to people performing similar analysis or capturing traces.

8.1 Trace Formats

The CERN and NCAR traces used very different formats to record the system state:

CERN traces recorded the state of each file that was modified, noting open time, close time, user, written bytes, read bytes and a variety of information about the file.

NCAR traces on the other hand recorded actions. So the trace records CREATE, or MIGRATE, and all the information about the action, including the user initiating it and effects of the action were recorded.

Both formats involve trade-off in what is recorded. The CERN format is useful for observing how files change over time, but loses out information on what kind of action happened on the file. Some information can be inferred, such as an unobserved file going from size 0 to a positive size can be classified as creation. However, it is hard to distinguish other actions in such a case, actions such as MIGRATE vs DELETE.

NCAR format on the other hand is completely non-uniform. As each action has a varying set of recorded fields, analysis and storing such a format is significantly harder. While each row gives us more information about actions, doing such an analysis on large scale limits how this data could be organized in a structured way.

Ultimately, the trace capturing framework must use a format which is better suited for the kind of analysis they expect to be performed on the recorded data.

8.2 File formats

Since most of this analysis was run directly on files, there are a few optimizations we found useful for such a task. Row based vs Columnar file formats :

Depending on the kind of analysis you want to perform, columnar formats such as

Parquet offer significant benefits over row based formats. Aggregations and sums over a range can be sped up if a particular column is stored in sequential locations in the file. For operations that were dependent on nearby timestamps or ranged across fields, row based formats were much faster for the same operation. We ended up using both types of file formats over the duration of this project to speed up certain analysis tasks.

8.3 Reflections on Trace Analysis

The CERN trace data is a rich trove of system state information we have analyzed it extensively over the past several months. Yet the data has many more aspects that we have not yet looked at. Performing analysis on such a large set of data is difficult. We ran into significant issues with memory usage, crashes due to too many open files, crashes due to type overflows among other things. We have specific feedback on trace capture methodology:

- Capturing the action type is more useful that just the state, as using the state to infer the action can be error prone.
- Traces can grow very quickly, especially in case of large scale systems, it is important to use file formats which can allows quick filtering and projection.

  For example, using Parquet over TSV allows quick reduction of fields into a smaller set, as each analysis action is going to be over a small set of data. The `path` field takes up most of the storage, but is not required for more operations. It can be ignored using a single `filter()` in Parquet, but not so much in row based formats such as TSV.

- It would be really helpful to get a complete snapshot of the system just before and just after the trace capture duration, as it would give us a lot more information to the creation, deletion, and changes to the system and the files over the period.

- Trace analysis requires a lot of knowledge of the system under consideration and frequent communication with the people who use it. More published literature from scientific and industrial sources about their storage systems would be really helpful in this regard.

The CERN traces capture a large amount of data. One of the goals of this project was to simplify access to this data, and we have scripts and techniques available, open to CRSS members to quickly slice and dice through this data. The data can be used in a variety of systems for testing, and the framework we built makes it easy to do so.

Bibliography

[1] CERN Annual report 2017. Tech. rep., CERN, Geneva, 2018.

[2] Adams, I., Madden, B., Frank, J., Storer, M. W., and Miller, E. L. Usage behavior of a large-scale scientific archive. In *Proceedings of the 2012 International Conference for High Performance Computing, Networking, Storage and Analysis (SC12)* (Nov. 2012).

[3] Adams, I. F. *Understanding Long-Term Storage Access Patterns*. PhD thesis, University of California, Santa Cruz, 2013.

[4] Adams, I. F., Storer, M. W., and Miller, E. L. Analysis of workload behavior in scientific and historical long-term data repositories. *ACM Transactions on Storage 8*, 2 (2012).

[5] Agrawal, N., Bolosky, W. J., Douceur, J. R., and Lorch, J. R. A five-year study of file-system metadata. In *Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST '07)* (Feb. 2007), pp. 31–45.

[6] Breslau, L., Cao, P., Fan, L., Phillips, G., and Shenker, S. On the Implications of Zipf's Law for Web Caching. In *3rd International WWW Caching Workshop* (June 1998).

[7] Colarelli, D., and Grunwald, D. Massive arrays of idle disks for storage archives. In *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing (SC '02)* (Nov. 2002).

[8] Dean, J., and Ghemawat, S. MapReduce: Simplified data processing on large clusters. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI '04)* (San Francisco, CA, Dec. 2004).

[9] Grawinkel, M., Nagel, L., Mäsker, M., Padua, F., Brinkmann, A., and Sorth, L. Analysis of the ECMWF storage landscape. In *Proceedings of the 13th USENIX Conference on File and Storage Technologies (FAST '15)* (Feb. 2015), pp. 15–26.

[10] Grawinkel, M., Pargmann, M., Dömer, H., and Brinkmann, A. Lonestar: an energy-aware disk based long-term archival storage system. In *Proceedings of the 17th International Conference on Parallel and Distributed Systems (ICPADS '11)* (2011), pp. 380–387.

[11] Jaffe, E., and Kirkpatrick, S. Architecture of the Internet Archive. In *Proceedings of The Israeli Experimental Systems Conference (SYSTOR '09)* (May 2009).

[12] Jensen, D. W., and Reed, D. A. File archive activity in a supercomputer en-

vironment. Tech. Rep. UIUCDCS-R-91-1672, University of Illinois at Urbana-Champaign, Apr. 1991.

[13] Leung, A. W., Pasupathy, S., Goodson, G., and Miller, E. L. Measurement and analysis of large-scale network file system workloads. In *Proceedings of the 2008 USENIX Annual Technical Conference* (June 2008).

[14] Li, Y., Bel, O., Chang, K., Miller, E. L., and Long, D. D. E. CAPES: Unsupervised storage performance tuning using neural network-based deep reinforcement learning. In *Proceedings of the 2015 International Conference for High Performance Computing, Networking, Storage and Analysis (SC17)* (Nov. 2017).

[15] Miller, E., and Katz, R. An analysis of file migration in a Unix supercomputing environment. In *Proceedings of the Winter 1993 USENIX Technical Conference* (Jan. 1993), pp. 421–433.

[16] Peters, A. J., and Janyst, L. Exabyte scale storage at CERN. *Journal of Physics: Conference Series 331*, 5 (dec 2011), 052015.

[17] Storer, M. W., Greenan, K. M., Miller, E. L., and Voruganti, K. Pergamum: Replacing tape with energy efficient, reliable, disk-based archival storage. In *Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST '08)* (Feb. 2008).

[18] Tange, O. *GNU Parallel 2018*. Ole Tange, Apr. 2018.

[19] Vohra, D. Apache parquet. In *Practical Hadoop Ecosystem*. Springer, 2016, pp. 325–335.

[20] Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., and Stoica, I. Apache Spark: A unified engine for big data processing. *Communications of the ACM 59*, 11 (Nov. 2016), 56–65.