Analysis of the ECMWF Storage Landscape

Matthias Grawinkel, Lars Nagel, Markus Mäsker, Federico Padua, and André Brinkmann, ***Johannes-Gutenberg University Mainz;*** **Lennart Sorth,** ***European Centre for Medium-Range Weather Forecasts***

https://www.usenix.org/conference/fast15/technical-sessions/presentation/grawinkel

This paper is included in the Proceedings of the 13th USENIX Conference on File and Storage Technologies (FAST '15).

February 16–19, 2015 • Santa Clara, CA, USA

ISBN 978-1-931971-201

Open access to the Proceedings of the 13th USENIX Conference on File and Storage Technologies is sponsored by USENIX

Analysis of the ECMWF Storage Landscape

Matthias Grawinkel, Lars Nagel, Markus Mäsker, Federico Padua, André Brinkmann
Johannes Gutenberg University Mainz

Lennart Sorth
ECMWF

Abstract

Despite domain-specific digital archives are growing in number and size, there is a lack of studies describing their architectures and runtime characteristics. This paper investigates the storage landscape of the European Centre for Medium-Range Weather Forecasts (ECMWF) whose storage capacity has reached 100 PB and experiences an annual growth rate of about 45%. Out of this storage, we examine a 14.8 PB user archive and a 37.9 PB object database for metereological data over a period of 29 and 50 months, respectively.

We analzye the system's log files to characterize traffic and user behavior, metadata snapshots to identify the current content of the storage systems, and logs of tape libraries to investigate cartridge movements. We have built a caching simulator to examine the efficiency of disk caches for various cache sizes and algorithms, and we investigate the potential of tape prefetching strategies. While the findings of the user archive resemble previous studies on digital archives, our study of the object database is the first one in the field of large-scale active archives.

1 Introduction

The number and size of computing sites with domain-specific archives has reached new heights and is still increasing. Next to the ever faster compute systems, storage systems are also growing in multiple dimensions like available capacity, access frequency, and required throughput. With the increased computing power and new algorithms from the big data area, computations tend to use and create more data. Here, many archives become active in the sense that every stored datum may be read at any point in time.

For building storage systems that meet the demands of active archives, it is necessary to understand how today's systems evolved, how they work and in which direction the development is heading. Unfortunately, only a small number of publicly available studies exist that analyze the storage infrastructure and characterize the stored data as well as storage access patterns and growth patterns. The result is a lack of representability of previous studies, as comparable studies are missing. Most of today's multi-petabyte storage systems follow a tape backend + disk caching approach. While disks offer the better performance and more flexibility in their access characteristics, tape is still cheaper in terms of capacity. The disk-to-tape ratio is therefore a tradeoff between price, performance, and capacity.

Previous studies investigated traces of desktop or network file systems [19, 28], internet accessible content delivery networks [15, 17], in-memory caches [3], or digital archives and content repositories [21, 1, 14]. The presented study is the first analysis of an active archive – a large-scale content repository where all data is subject to be accessed at any time.

The contributions of this paper are an in-depth system analysis of two archival systems from the previously uncharted weather forecasting domain, a simulator-driven evaluation of workload trace files to improve the disk cache efficiency, and a feasibility evaluation of tape prefetching strategies. The subject of our study is the storage environment of the *European Centre for Medium-Range Weather Forecasts* (ECMWF)[1], which provides medium-range global weather forecasts for up to 15 days and seasonal forecasts for up to 12 months. To achieve this, they utilize supercomputers[2] and, as of September 2014, storage with a capacity of 100 PB. Next to fast HPC storage, they run two in-house developed archival systems: a general-purpose user-accessible archive (ECFS) for file storage hosting 14.8 PB of data and a large object database for meteorological data (MARS) that hosts 37.9 PB of primary data consisting of 170 billion fields. It is regarded as the world's largest

[1] http://www.ecmwf.int/

[2] http://www.top500.org/site/47752

archive of numerical weather prediction data. Both systems consist of multiple tape libraries with disk-based caches in front of them. We have developed a trace-based storage simulator for the ECFS traces to determine the efficiency of various cache strategies and to optimize the hit-ratio of the disk caches. Additionally, we look into the logs of the tape libraries and the backend HPSS system [12]. Examining logs with more than 9.5 million tape load operations in 2012-2013, we investigate the feasibility of tape prefetching strategies.

Our study shows that the two storage systems are used in different ways. While the ECFS is an archive with mostly write accesses and only a small set of actively used data, the MARS system is read-dominant, and all its data are subject to be read. Both systems face an exponential data increase with a compound annual growth rate (CAGR) of about 45% over the last years and about 50% today. In total, the ECFS logs cover 29 months of 2012-2014 and the MARS logs 50 months of 2010-2014. These logs cover the integration of new applications, models and hardware. Especially, the additional throughput and capacity required by a newly commissioned supercomputer becomes visible at several points and is one of the reasons for the exponential growth in storage capacity.

2 Related Work

Large-scale storage and archival systems have been investigated for many years. Baker et al. and Rosenthal et al., for example, discuss the technical and non-technical challenges for building long-term digital repositories [4, 5, 24]. The technical problems include large-scale disasters, component and media faults, and the obsolescence of hardware, software and formats. Furthermore, human errors, loss of data context, or misplanning need to be considered. Rosenthal especially emphasizes the economical aspects of long-term archives. Economic faults, erroneous capacity planning, or the wrong use of storage technologies can be a threat for long-term data availability. Many previous studies help to encounter these threats and to build reliable and successful digital archives.

Most studies that analyze the contents or behavior of file systems deal with workstations, general-purpose network file systems, or HPC storage systems [13, 2, 19, 11, 8, 28]. They examine static file system snapshots, request traces and operating system logs to investigate multiple dimensions of storage systems. Meister et al. investigated the possible impact of applied deduplication on HPC storage [23].

Another investigated area are large-scale publicly accessible systems, their usage patterns, and the efficiency of caching [7, 25, 15, 3, 17]. It is especially important to understand and characterize traffic and user behavior to build and improve caching infrastructures.

There exist only a few publicly available archival traces [20] and analyses of recent archives. Madden et al. wrote a technical report on the user behavior in the NCAR archival system over a three year period from 2008 to 2010 [21]. They also started an investigation of namespace locality of user sessions. Frank et al. compare the logs of an NCAR system to a previous study of the system from 1992 [14]. In the interval, the read-to-write ratio on the system changed from 2:1 to 1:2 which indicates that archives are becoming increasingly write-only. From the traces it was also derived that 30% of the requests have a *latency to first byte* of more than three minutes. In order to improve the latency, the authors suggest large disk caches that permanently hold the small files. The most comprehensive study was conducted by Adams et al. [1] who examined multiple public and scientific long-term data repositories for their content and workload behavior. Especially for the scientific LANL and NCAR repositories, disks play an increasingly important role, which becomes visible by comparing the 1:262 disk-to-tape ratio at NCAR in 1993 with the 1:3.3 ratio at LANL in 2010.

Previous studies document the rise of disk drives, used either as a complement to or as a replacement for tape in large-scale archival scenarios. Colarelli and Grunwald, for example, argue for the replacement of tape archives by large disk arrays that are switched off when not in use [10]. This idea was refined in the Pergamum system by Storer et al., a distributed system of powered-down disks for archival workloads [26]. Grawinkel et al. proposed a high-density MAID system optimized for "write once, read sometimes" workloads [16]. Today, large-scale archival systems are in production that primarily build on disk technology, like the Internet Archive [18].

3 Background

The European Centre for Medium-Range Weather Forecasts (ECMWF) is an independent intergovernmental organization supported by 20 European member states and 14 co-operating states. The center was established in 1975 and hosts one of the largest supercomputer complexes in Europe. Storage for the computation environments is driven by HPC storage systems and two large archival systems developed in-house, namely ECFS and MARS, that will be investigated in this paper. Today the center hosts a combined storage capacity of about 100 PB. This includes the HPC storage and backups. All files of the archival system are stored on tape, and important files are stored to a second tape copy.

ECFS is used as a general-purpose archival system and accessible for users of the ECMWF compute envi-

ronment. In September 2014 it stored 137 million files with a total size of 14.8 PB. The system provides 0.34 PB of disk caches so that the disk-to-tape ratio is 1:43. All data is written to a disk cache first before it is migrated to tape.

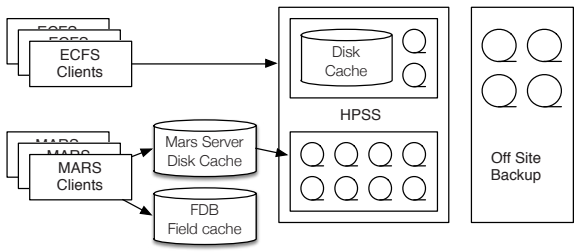

Figure 1: Abstract overview of storage environment.

MARS is an object store for meteorological data with a database-like API. A custom query language is used to specify a list of relevant fields. The system assembles these fields into a package and stores it on a target storage system so that it can be accessed from the HPC systems. To reduce the metadata overhead and keep the file backend manageable on tape, fields are stored in appendable files. A dedicated database allocates and maps fields to offset-length pairs on files. New, recently, and often used fields are staged and cached in a dedicated field database (FDB), which is the primary target for queries. If the FDB does not contain a field, the MARS servers are queried. MARS manages its own disk cache. In case of a cache miss, the required files are loaded from tape. All MARS servers for all projects provide a total of 1 PB of disk cache which results in a 1:38 disk-to-tape ratio. The FDBs are stored on the HPC storage systems connected to the supercomputers and can grow to multiple PB. In September 2014 MARS stored 54 PB of primary data consisting of 170 billion fields in 11 million files. Additionally, the system uses 800 GB of metadata. Each day, 200 million new fields are added.

Figure 1 provides a high level overview of the storage environment, which is implemented around the High Performance Storage System[3] (HPSS) that provides the disk caches for ECFS and manages the tape resources for both ECFS and MARS. Tape is considered to be the final destination for data. Every cached file has a copy on tape. In contrast to ECFS, MARS manages its own disk caches outside of HPSS. The ECMWF runs multiple project-specific MARS databases that are mapped to individual storage pools in the HPSS system. In the following, we will treat ECFS and MARS as two different storage systems.

[3]http://www.hpss-collaboration.org/

3.1 Available Log Files

This work analyzes log files and database snapshots provided by ECMWF. All log files were obfuscated by replacing user information and each part of a file's path by a hash sum, except file extensions. No user information can be revealed, but access patterns and file localities are preserved. We developed a set of python scripts to obfuscate, sanitize, and pack the raw source data. The following analysis is based on compressed log files from multiple ECFS, MARS, and HPSS servers. The gathered and investigated files are:

ECFS access trace: Timestamps, user id, path, size of GET, PUT, DELETE, RENAME requests. 2012/01/02 - 2014/05/21.

ECFS / HPSS database snapshot: Metadata snapshot of ECFS on tape. Owner, size, creation/read/modification date, paths of files. Snapshot of 2014/09/05.

MARS feedback logs: MARS client requests (ARCHIVE, RETRIEVE, DELETE). Timestamps, user, query parameters, execution time, archived or retrieved bytes and fields. 2010/01/01-2014/02/27.

MARS / HPSS database snapshot: Metadata snapshot of MARS files on tape. Owner, size, creation/read/modification date, paths of files. Snapshot of 2014/09/06.

HPSS WHPSS logs / robot mount logs: Timestamps, tape ids, information on full usage lifecycle from access request till cartridges are put back to the library. 2012/01/01 - 2013/12/31.

The traces and tools used are publicly available as outlined in Section 9.

4 ECFS User Archive

Users of the ECMWF compute environment use ECFS as an intermediate and long-term storage for general purpose data. New and recently retrieved files are stored in disk pools and are migrated to the tape storage by HPSS. Files are categorized by their size and are spread to six pools with different capacities and properties. The ranges as well as the number and size of stored files are listed in Table 1. Though tape is considered as the primary storage, files of the *Tiny* class are primarily stored on mirrored disks and only backed up to tape. Therefore, to read tiny files, tape is never used. In ECFS, no files are updated in place, but a file may be overwritten.

4.1 Metadata Snapshot Analysis

In contrast to the trace files that only yield data being accessed within the investigated time frame, the HPSS

Group	From	To (incl.)	Count	Used Capacity
Tiny	0	512 KB	36.0 mil.	4.4 TB
Small	512 KB	1 MB	9.1 mil.	6.3 TB
Medium	1 MB	8 MB	29.5 mil.	101 TB
Large	8 MB	48 MB	30.0 mil.	585 TB
Huge	48 MB	1 GB	29.7 mil.	6.2 PB
Enormous	1 GB	∞	3.1 mil.	8 PB

Table 1: File size categorization. Count and capacity refer to Sept. 2014.

database snapshot gives a full view on all stored files on 2014/09/05.

File system stats	
Total #files	137.5 mil.
Total used capacity	14.8 PB
Largest file size	32 GB
#Directories	5.5 mil.
Max files per directory	0.43 mil.
#Files never read from tape	101.3 mil. (11.3 PB)

Most common file types	
by file count	**by used capacity**
unknown (27.8232%)	unknown (39.3306%)
.gz (20.4319%)	.tar (21.2699%)
.tar (7.8015%)	.gz (12.4954%)
.nc (7.6312%)	.nc (7.8819%)
.grib2 (1.9438%)	.lfi (2.2399%)
.raw (1.7284%)	.pp (1.0087%)
.txt (1.5095%)	.sfx (0.9327%)
.Z (1.4862%)	.grb (0.8471%)
.bufr (1.4451%)	.grib (0.3977%)
.grb (1.4402%)	.bz2 (0.3083%)

Table 2: Statistics on ECFS tape storage.

The summary in Table 2 presents a total of 137.5 million files that are stored in 5.5 million directories and use 14.8 PB of capacity. The table also presents the most common file types by count and occupied capacity. Next to the unknown files that do not yield an extension on their file names, packed, compressed, and weather domain specific files are highly represented. Figure 2 shows a histogram of the most common file sizes. The system stores a large amount of files between 0 bytes and 1 KB, but else visually follows a Gaussian distribution that peaks at 8-16 MB.

The database excerpt also contains the *creation*, *modification*, and *read* timestamps of files. These timestamps mark the access times of a file on the tape drives and do therefore not reflect the access times of cached files. Files of the *Tiny* group (see Table 1), for example, are fully cached on disk and never retrieved from tape. The file system statistics of Table 2 show 101.3 million files that were never read from tape. These files were only uploaded or modified. If they were accessed, they were read from the HPSS' disk cache.

The upper graph of Figure 3 visualizes the absolute number of existing files at a particular point in time and the number of files that were *unread* or *unmodified* since

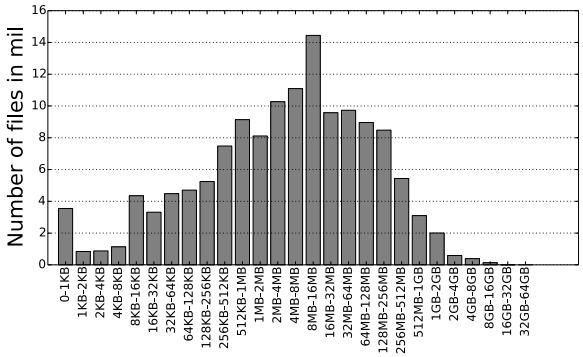

Figure 2: Histogram of stored ECFS files sizes.

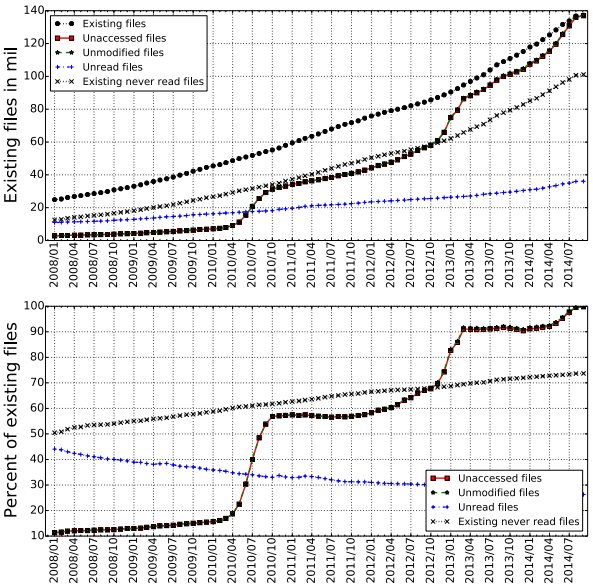

Figure 3: Top: Total number of existing files with amounts of unmodified and unread files since the point in time. Bottom: Fraction of unread or unmodified files relative to existing files.

that date. An *unaccessed file* was neither read nor modified and an *existing never read file* has no read time stamp and was therefore created and possibly updated only. The lower graph presents the fractions relative to the existing number of files to delineate the behavior over time. Since 2012/01 60 million new files were created. With the beginning of 2013, the new supercomputer becomes visible as more files are created and the graph steepens. In general, the number of existing files follows an exponential growth. The number of unmodified files closely follows the number of unaccessed files. This means that most of the last actions on files were modifications, which also includes the initial creation, and not reads. In the last year (2013/07 - 2014/07), the amount of never read files slightly grew from 68% to 73%, while the number of unread files shrank from 27% to 25%. In 2013/07 about 90% of the data stored on tape were neither read nor

modified. This value changes by looking back an additional year till 2012/07 which covers the introduction of the new supercomputer. About 64% of the files existing at that point in time were not accessed until 2014/07.

4.2 Workload Characterization

For the time period from 2012/01/01 to 2014/05/20 a full trace of all ECFS operations has been investigated. Table 3 summarizes the key metrics.

Total GET requests	38.5 mil.
Total GET bytes	7.24 PB
Total PUT requests	78.3 mil.
Total PUT bytes	11.83 PB
Total DEL requests	4.2 mil.
Total RENAME requests	6.4 mil.
Total different files	73.4 mil.
Total different dirs	6.2 mil.
#Files with PUT	66.2 mil.
#Files with GET	12.2 mil.
Cache hit ratio by requests	86.7%
Cache hit ratio by bytes	45.9%

Table 3: Characterization of ECFS workload 2012/01/02 - 2014/05/21.

During the observed timespan, a total of 38.5 million GET requests were executed on 12.3 million unique files, a total of 78.3 million PUT requests on 66.2 million unique files were counted, and 4.2 million files were deleted from ECFS. As there are more unique written files than PUT requests, some files were updated or overwritten. In comparison to the NCAR analysis [14] where 30% of the stored files were read during the 29 month observation timespan, ECFS saw reads on 12.3 million distinct files which is 9% of the total corpus.

We analyzed the number of PUT and GET requests and their respective traffic based on the ECFS file size categories of Table 1. Figure 4 breaks down these metrics on a monthly basis. Until 2013/03, the system has a balanced throughput of 200-300 TB PUT and GET traffic per month with slightly prevailing write traffic. With the introduction of the new computer in the first quarter of 2013, the amount of written data doubles both, traffic- and request-wise, while the retrieval rates and volume merely rise.

Figure 5 visualizes the hotness of files during the observed timespan. As the analysis in Section 4.1 shows, the system contains a lot of data that have not been accessed within this time frame or were just uploaded without being retrieved again.

The main argument that is underlined by the plot is that in total 50% of all GET requests hit less than 5% of the stored files. This argument is supported by the overall 86.7% disk cache hit ratio with a disk-to-tape ratio of 1:43. The cached requests are responsible for

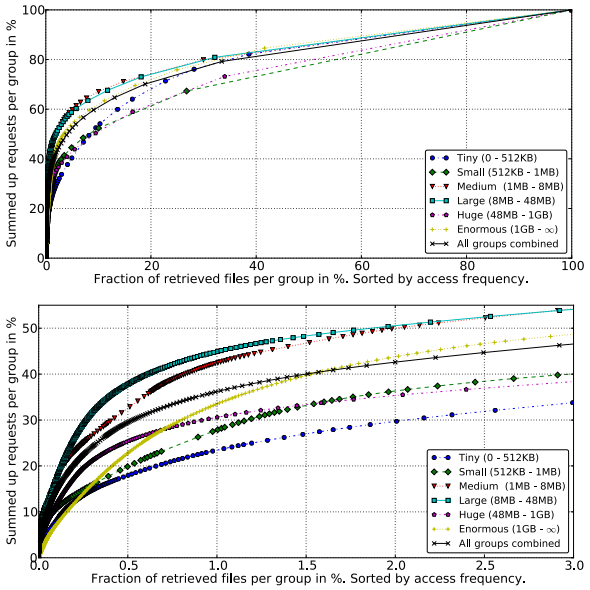

Figure 5: Top: CDF over file GET requests per file per size group.
Bottom: Zoomed to most frequently retrieved 3 %.

45.9% of the retrieved bytes, which leads to the assumption that most small reads can be satisfied by the cache and mostly larger files are retrieved from tape. The later cache analysis in Figure 15 (see Section 7) visualizes the cache hit ratios observed from the ECMWF traces for the different file size categories. While *Tiny* files achieve 100%, only 60% of the *Huge* and 50% of the *Enormous* file retrieval requests are served from disk.

4.3 User Session Analysis

For every request in the trace, the user id and the host that executed the command are known. A user id can be used by all sorts of processes running on multiple systems at the same time. The trace reveals 1,190 unique user ids and 2,647 unique hostnames. As presented in Table 3, a total of 11.8 PB of data have been written and 7.2 PB have been retrieved from the archive. Figure 6 visualizes how the bytes and requests are distributed to the identified users. The plot shows that only 850 users wrote data while 1,075 users retrieved data. Furthermore, only a small fraction of less than 100 users make up more than 90% of the traffic.

We gathered all commands issued by a user id from the same hostname and create clusters of executed requests that occurred within close succession. If the time between two requests is longer than the window, they are clustered into different groups. We call all requests within such a group a user session. Figure 7 presents the number of actions per identified user session for a growing time window based on the methodology used

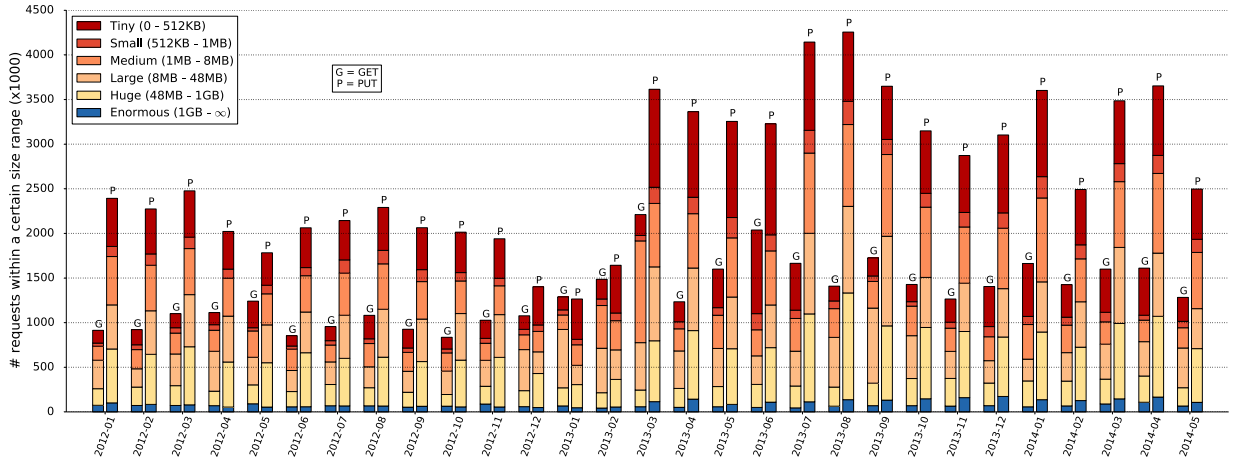

Figure 4: Total requests per month

Key	P05	P50	mean (+-95%)	P95	P99	Count
#Sessions per user_id	1	35	2,276.76 (± 1,006.85)	7,315	28,861	1,190
#Sessions per user_id@host	1	4	92.70 (± 7.50)	352	1,512	29,227
Total #Actions	2	7	47.04 (± 0.69)	126	579	2,709,343
GET Requests per session	1	4	35.55 (± 0.78)	108	571	1,083,067
ReGET requests per session	1	2	31.98 (± 3.66)	99	442	132,515
PUT Requests per session	1	5	34.43 (± 0.44)	97	373	2,274,645
Dirs with GETs	1	2	8.15 (± 0.16)	21	96	1,083,067
Dirs with PUTs	1	2	6.06 (± 0.07)	14	71	2,274,645
Retrieved files per directory	1	1.78	10.05 (± 0.23)	30	149.75	1,083,067
Archived files per directory	1	2	7.44 (± 0.12)	27	74	2,274,645
Retrieved MBytes	0.56	192.13	7,172.91 (± 221.74)	17,150.69	86,444.15	1,083,067
Archived MBytes	0.02	206.04	5,591.52 (± 94.84)	19,588.74	64,995.07	2,272,399
Session lifetime in s	0	154	2,601.60 (± 21.13)	9,295	38,456	2,709,343
Gap between sessions	120	250	896.26 (± 11.70)	3,070	3,500	29,227

Table 4: ECFS user session analysis. A total of 2,709,343 sessions were identified.

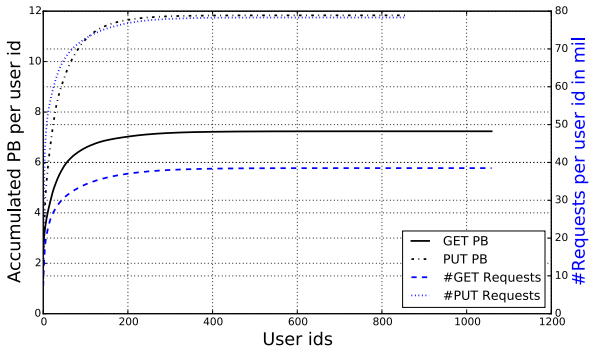

Figure 6: Summed up traffic for GET and PUT requests per unique user id

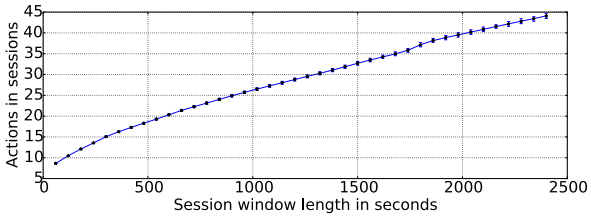

Figure 7: Mean actions per user sessions for growing window sizes.

by Madden et al. [21]. In contrast to Madden's approach, the graph does not show a plateau that would characterize a typical session. Therefore, we used a machine learning approach over all actions of each user to identify a window size that produces the most stable clusters for that user. For the 1,190 users, we identified a total of 2,709,343 sessions and Table 4 presents statistics over some key performance points. As for all following statistics, we present the 5, 50, 95 and 99 percentiles, a mean with a 95% confidence interval and the count of occurrences of the performance points. The statistics for a metric like *GET requests per session* is only counted if the session had at least one GET request. A write only session would not be counted here.

The results underline the trend shown in Figure 6. A small fraction of users is very active which becomes visible in the high differences of the P50 and P95 percentiles of multiple performance points. Also the volume in terms of requests and transferred bytes follows this trend, where a small fraction of sessions dominate

the workload. The session lifetime considers the time from the beginning of the first until the end of the last action within a user session. Many sessions consist of a single action that is executed within a second or less. On the other hand, we observe sessions longer than 10 hours for the P99 which are most probably regular operations like cron jobs for backups.

The trace file yields full obfuscated paths of accessed files. Therefore, we investigated the locality of accesses within a session. If a session retrieved files, on average 36 GET requests were issued which accessed 8 directories with about 10 files per directory. If a session also archived data, on average 34 files were uploaded to 6 different directories with about 7 files per directory. As observed in the study by Adams et al. [1], we also see user sessions that re-retrieve the same file within the lifetime of the session, which occurred in 132,515 of the 1,083,067 sessions with GET requests. Out of the total 38.5 million GET requests, 11% (4.2 million) were re-retrievals of files within a user session.

5 MARS Database

The Meteorological Archival and Retrieval System (MARS) is the main repository of meteorological data at ECMWF. It contains petabytes of operational and research data, as well as data from special projects. In contrast to the ECFS, where files are identified by a unique path, MARS hosts billions of metereological fields that cannot be directly addressed by a user, but are the result of a query. The available log files of the MARS system contain all parameters of the queries and the number and source of the returned fields, but do not allow to identify the exact keys of the accessed fields. Therefore, this analysis cannot investigate the hotness of fields or files, but can only characterize the observed traffic.

MARS is based on a 3-tier storage architecture with the FDB as the first, the MARS servers' disk caches as the second and the HPSS tapes as the third layer. All fields in MARS are eventually persisted to the files in the HPSS tape backend, but requests are primarily served by the FDB and the MARS servers. The system also applies domain specific knowledge to improve the cache hit rates. For example, if files or full tapes are identified as hot, they can be manually loaded and locked to the MARS servers' disk caches. Currently 250 TB are reserved for this manual cache optimization.

5.1 Metadata Snapshot Analysis

The following analysis investigates an HPSS database snapshot of all MARS files on tape from the 2014/09/04. Table 5 presents a summary of the findings. Compared to ECFS, MARS stores a significantly smaller amount of files that use a larger total capacity of 37.9 PB. When the snapshot was taken, a total of 7.8 million of the 9.7 million stored files were never read from tape.

File system stats	
Total #files	9.7 mil.
Total used capacity	37.9 PB
Largest file size	1.34 TB
#Directories	555,799
Max files per directory	38,375
#Files never read from tape	7.9 mil. (24.9 PB)

Table 5: Statistics on MARS' tape storage.

Similar to ECFS, the histogram of the file sizes in Figure 8 resembles a Gaussian distribution, yet with a higher average file size and a higher maximum at 128-256 MB. The size of the largest file stored is 1.34 TB.

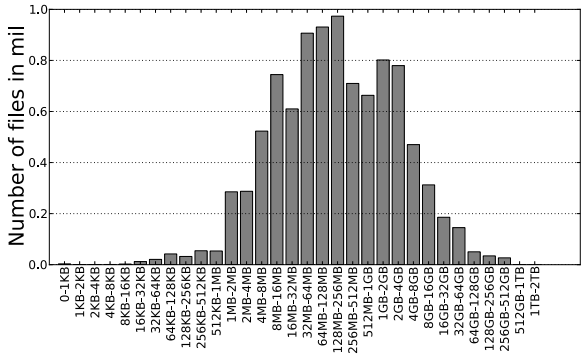

Figure 8: Histogram of files sizes

The visualization of *creation*, *modification*, and *read* times in Figure 9 follows the schematic described in Section 4.1. Again, the upper graph visualizes the absolute number of existing files at a particular point in time and the number files that have not been modified or read since that date. The lower graph presents the fractions relative to the existing number of files. The high rate of unaccessed files and a modification rate close to 100% shows that files are predominantly created, rarely updated and only read sometimes from the HPSS tape backend. The lower graph shows that up to 80% of the files on tape were written, but never read again. This behavior either indicates a cold storage or a strong caching infrastructure. In contrast to the ECFS analysis (see Figure 3), the introduction of the new supercomputer in Q1/2013 is not visible in Figure 9. Though Figure 10 shows a significant change of daily written fields and bytes, the file creation rate on the HPSS system does not change. This is because new fields are aggregated at the FDB and MARS server levels that are written as a file which then appears as a new file in HPSS. The assumption is that the average size of newly created files grows over time.

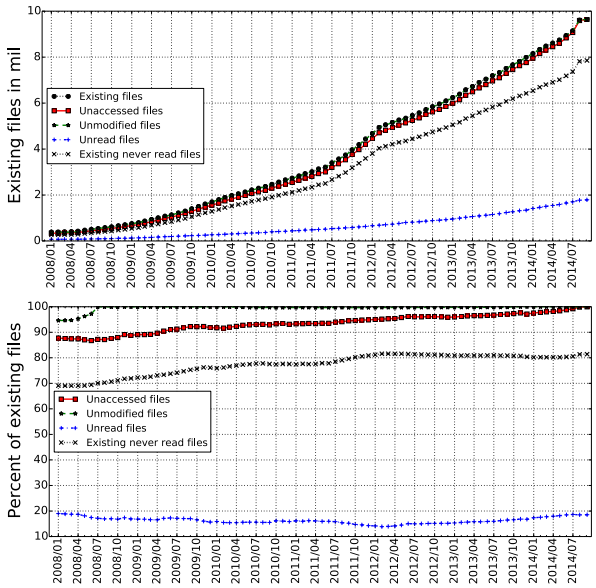

Figure 9: Top: Total number of existing files with amounts of unmodified and unread files since the point in time. Bottom: Fraction of unread or unmodified files relative to existing files.

5.2 Workload Characterization

We present the analysis of the MARS feedback logs over the timespan 2010/01/01-2014/02/27 that quantifies user requests and the resulting traffic. Figure 10 visualizes the daily throughput in bytes and requests. Again, the introduction of the new compute environment in Q1/2013 becomes visible in elevated throughput rates. The old environment can be characterized by a constant daily read rate of 40-50 TB (100-120 million fields) and 15-20 TB (about 50 million fields) of written data. With the new computer, the read rate doubles, but the write rate nearly increases threefoldly. During the peak throughput rate in 2013/01 both old and new supercomputers were running.

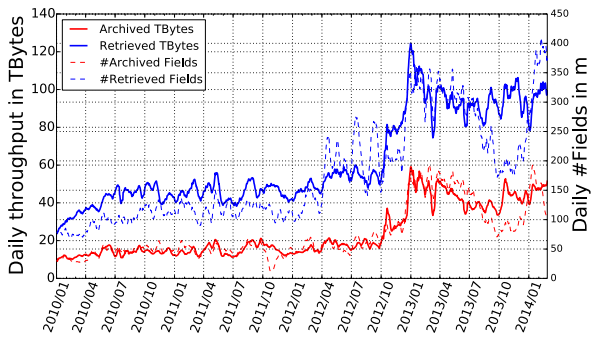

Figure 10: Throughput and number of accessed fields.

Table 6 presents some key characteristics of the considered 50 months. A total of 1.2 billion read requests were executed that fetched 269 billion fields which accounted for 91.6 PB of data, while 115 million requests created 114.7 billion new fields which account for 35.9 PB data. The logs breakdown each request into the number of fields, their summed up sizes and source of the returned fields. In total, 80.7% of all requests can be fully served by the field database (FDB), 17.6% of the requests also require data from the MARS server's disk drives and only 2.2% of the requests include data from tapes. 1.4% of the retrieve requests are fully served from tape without any cached data from disk. Considering the number of retrieved fields, MARS achieves a 95.1% cache hit ratio with a 1:38 disk-to-tape ratio on the MARS servers and a similar but not concretizable ratio on the FDB.

Total retrieved bytes (fields)	91.6 PB (269 bil.)
- from FDB bytes (fields)	54.2 PB (212 bil.)
- from MARS/disk bytes (fields)	29.4 PB (43.3 bil.)
- from HPSS/tape bytes (fields)	8 PB (13.3 bil.)
Total retrieve requests	1.2 bil.
- including FDB	992 mil. (85.3 %)
- from FDB only	938.9 mil. (80.7%)
- including MARS/disk	204.9 mil. (17.6%)
- from MARS/disk only	151.3 mil. (13%)
- including HPSS/tape	25.3 mil. (2.2%)
- from HPSS/tape only	16 mil. (1.4%)
Total archive requests	115 mil.
Total archived bytes (fields)	35.9 PB (114.7 bil.)

Table 6: Characterization of MARS workload 2010/01/01-2014/02/27.

Figure 11 characterizes the most active user ids in terms of retrieved and archived bytes and the according requests. It shows that the number of users who created content is significantly smaller than the number of users who retrieved data. A huge amount of traffic was actually generated by only two user ids. Some ids are shared by multi-purpose users, behind which multiple real user ids hide. Therefore, these very active users have to be considered as outliers.

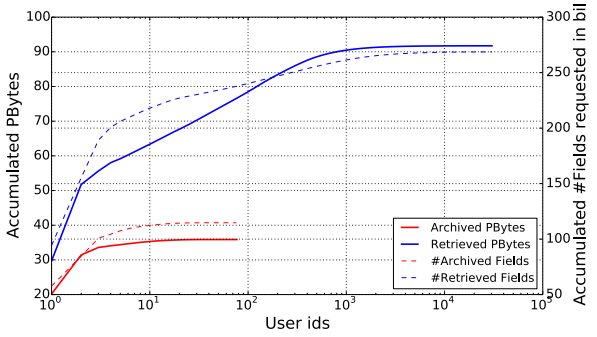

Figure 11: Traffic per unique user id.

6 Tape Mount Logs

Both ECFS and MARS use a HPSS powered tape archive as the final destination and primary copy of files. De-

spite the strong caching infrastructure, the tape robots are heavily used. Figure 12 illustrates the life cycle of

Figure 12: Tape states

one tape use in a simplified state diagram. To read data, a request is sent to the HPSS system that *loads* the tape into a drive. The tape is *mounted* as a volume and can be accessed. After a fixed timespan or on request, the volume is *unmounted*, but the tape can remain in the drive. If the tape is requested again while still being loaded, it can be *remounted*, or in the worst case *reloaded* into another drive. Eventually the tape is *unloaded* from the drive and put back to the library. Each tape has a unique identifier that indicates its type (STK T10k-B/C/D) and is assigned to ECFS or a MARS project. We use the identifiers to track the usage of the cartridges and map them to either ECFS or MARS. During the complete log period we saw a total of 231 different drive identifiers and 9,594 unique tape ids for ECFS and 23,118 for MARS.

Tape mount frequencies							
System	**#Tapes**	**P05**	**P25**	**P50**	**mean (+-95%)**	**P95**	**P99**
MARS	23,118	0	2	46	291.22 (± 9.70)	1,106	3,351
ECFS	9,594	1	12	85	296.64 (± 11.18)	1,408	2,470
Tape mount latencies in seconds							
System	**#Mounts**	**P05**	**P25**	**P50**	**mean (+-95%)**	**P95**	**P99**
MARS	6,730,218	26	30	35	54.35 (± 0.06)	155	262
ECFS	2,845,154	25	28	32	48.19 (± 0.07)	138	257

Table 7: Tape mount statistics

Figure 13 presents the access frequencies of tapes. Sorted by the most often accessed tapes, the total number of loads is summed up. The graph shows that MARS is accountable for 6.7 million and ECFS for 2.8 million tape loads. The right graph of the figure compares MARS and ECFS and reveals a similar distribution pattern. About 20% of all tapes are accountable for 80% of all mounts and more than 50% of the tapes are accessed in less than 5% of the loads.

Next we investigate the behavior of the tape system

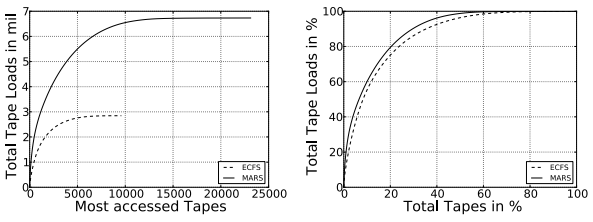

Figure 13: CDF over load requests per tape cartridge. Left: Absolute. Right: Normalized.

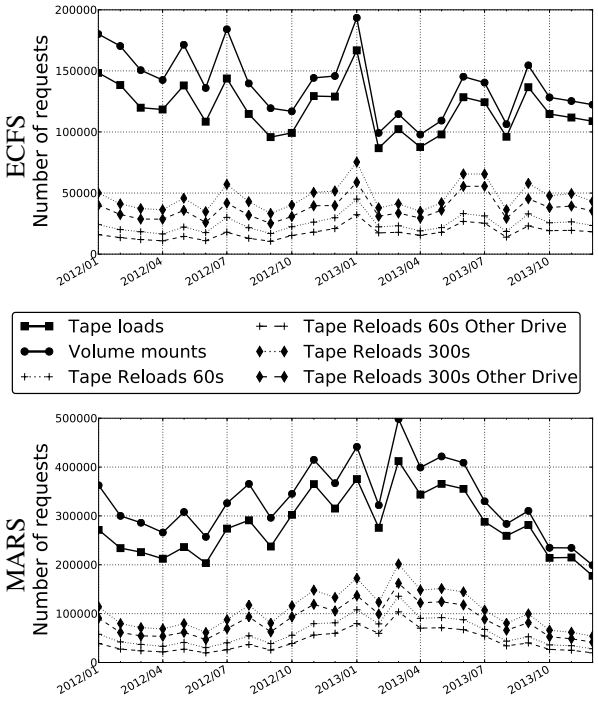

Figure 14: Mount Details for ECFS and MARS

over time. Both, the robot mount logs and the WHPSS logs were used for the analysis. Unfortunately, 6 days of the WHPSS logs were erroneous. We used the logs to feed finite state machines to track the cartridge states. Table 7 presents statistics of the time till a requested tape is available for work (volume mounted) and the number of load requests.

The ratio of 6.7 million MARS loads to the 2.8 million ECFS loads perfectly reflects the ratio of the stored data of 35.9 PB to 14.8 PB. The tape load times for the two categories are very similar, which leads to the assumption of equal or shared hardware in the backend. Although the median waiting time is only 35 (32) seconds, the mean is much higher due to some very long waiting times. More than 5% of all tape loads take more than 2 minutes and 1% of the loads take longer than 4 minutes.

Figure 14 visualizes the HPSS behavior for ECFS and MARS over time. The graphs show more volume mounts than tape loads, which shows the fraction of remounts without tape movements. The bottom of the graphs present the number of tape reloads and volume remounts within 60 and 300 seconds. A *tape reload 60s* means that after the tape was unloaded, within 60 seconds another mount request was issued. Over the observed time frame ECFS and MARS show a total of 21% of reloads within 60 seconds and 39% of reloads within 300 seconds. The significant finding is that in total 14.8% of all loaded tapes were unloaded from another drive less than 60 seconds ago.

Time slot (s)	MARS	ECFS
(0, 60]	73.922	8,740
(60, 120]	48,580	8,270
(120, 300]	107,276	23,126
(300, 600]	129,861	31,145
(0, 600]	359,639	71,281
(600, 1200]	189,691	46,792
(1200, 1800]	146,249	36,733
total success	695,569	154,806
[0, -1800]	849,229	162,965
(1800, fail]	8,061,777	1,364,171
Positive hit rate in %	7.24	9.20
Neutral hit rate in %	8.84	9.69
Total misses in %	83.92	81,11

Table 8: Prefetching hits based on the correlation analysis

6.1 Tape Prefetching

The average loading time could be improved by prefetching the tapes which are likely to be read next. In order to identify such tapes, we performed a correlation analysis on the tape mount logs using the Pearson correlation. In the following we describe the procedure and estimate the potential for improvement.

For any combination of two tapes x and y with a correlation coefficient of at least 0.8, we analyzed all load requests of x and measured the time difference until y was requested. Assuming the system would prefetch y once it sees a request for x, then this delay indicates the time y occupies a drive until it is requested. Table 8 shows the number of load requests within different time slots for the ECFS and MARS tapes.

We consider access delays of more than 1,800 seconds as prefetching failures, because with a high probability the tape would be evicted before being accessed. Furthermore, the interval [0, -1,800] shows the number of operations that did not result in a hit, but saw a load of y within the preceding 30 minutes. This is the case if x and y are requested at the same time, if y is already loaded or y was requested prior to x. In this case x would be the prefetching result of y's load request and therefore, should not issue a prefetching event itself. This time slot neither generates any profit, nor induces any costs, which is why we call these events neutral hit and do not consider them as misses. Misses are load requests of x that never see a corresponding load event of y during the following 1,800 seconds.

The MARS and ECFS logs show a total of 695,569 and 154,806 prefetching hits which on average would have resulted in a hit every 74 seconds. Considering the mean latency of 54.35 and 48.19 seconds per load request, the latency of these operations would have accumulated to 1.20 and 0.24 years and could be saved by prefetching. The total load request latency of all load operation of the two projects is 11.60 and 4.35 years, respectively. The above mentioned reduction of 1.20 and 0.24 years could reduce these by 10.3% and 5.5%. This is a theoretical upper limit, since the 9.4 million misses would nearly double the amount of tape loads and clearly is unsuitable.

To design a prefetching strategy, possible candidates have to be identified. Furthermore, it has to be verified that the robots and drives have enough spare resources to process the prefetching load operations without impairing operational use. The logs show an average of 546.33 ($\pm$ 3.65) load operations per hour and during the busiest 5% of hours, more than 894 operations were performed. During the peak 1% utilization, more than 1,046 load operations were executed per hour. In the absence of such peak loads, the robots should be able to handle additional loads induced by prefetching misses. Finally, prefetching would not be applicable if the drives are constantly busy. We consider a drive to be idle if a tape is loaded but not mounted. Since mounted tapes are always loaded, the ratio of the volume mount time to the tape loaded time is a good metric for the drives utilization. We calculated this ratio for every hour of the observed time frame and on average see that tapes are accessed 82% of the time they are loaded. For the 0.01 and 0.99 percentiles we see a usage of 65% and 94%, respectively. This shows that the drives are highly utilized, but offer idle time to apply prefetching strategies.

7 Cache Simulation

Using a newly designed cache evaluation environment, different cache eviction strategies have been analyzed running the ECFS trace files (which were described in Section 4.2). We reuse the file size categorization of Table 1 and investigate the cache efficiency for different caching strategies and cache sizes. All GET, PUT, DEL, RENAME operations of the trace are replayed to a simulator that mimics a simple disk cache. A GET request on a non-existent file triggers a cache miss and the file is loaded to the cache. Also all PUT requests load the file into the cache, which might lead to an eviction. We evaluate the following cache eviction strategies using the ECFS traces that cover a timespan from 2012/01 till 2014/05 and are visualized in Figure 4.

LRU Data is evicted on a Least-Recently-Used strategy.

MRU Data is evicted on a Most-Recently-Used strategy.

FIFO Queue based eviction.

RANDOM A random cache entry is chosen for eviction. The presented graphs show the average results over 10 runs.

ARC Adaptive Replacement Cache that keeps track of both frequently and recently used files with an eviction history [22].

Bélády Adaption of the Bélády algorithm [6] which evicts those files that will not be needed for the longest time in the future. This algorithm would only be optimal if all files had the same size, but nevertheless we use this almost perfect cache as a baseline. The construction of an optimal cache is NP hard [9].

The results are visualized in Figure 15. For every file size category used at ECMWF (see Table 1) we present a graph that analyzes the cache hit ratio for the different caching strategies and multiple cache sizes. The last row shows the relative difference for a single *combined* cache for all files against the combined hit ratio over the sum of all hits and misses of the 6 subcaches. Its capacity steps are the sum of the same step of the other six caches. A negative result means that a single huge cache has a better hit ratio in terms of requests. We used the full year of 2012 to warm up the caches and present the total cache hit ratio for the period from 2013/01 to 2014/05.

Only the ECMWF baseline for *Tiny* files achieves a 100% hit ratio, as all files are always held on disk. The first GET request on a file that had no previous PUT request in the observed trace will result in a cache miss. Therefore, our results cannot reach 100%. The rightmost capacity of the graphs present a cache with unlimited size which never evict files because it can hold all of them. Therefore, this point presents the theoretical maximum cache hit ratio for the observed time frame.

The graphs show that strategies like *MRU* and *FIFO* are not usable at all. Only for very large caches, they yield an acceptable hit ratio. Due to the high number of re-GET requests, the most recently used files should be cached and not evicted, which explains the bad hit rates for *MRU*. Also the *FIFO* reveals a bad performance because it neglects the popularity of files. The constant writes of new files will evict files independent of their usage patterns.

While *MRU* and *FIFO* show a harmful behavior for cache efficiency, the *Random* strategy provides an unoptimized baseline. The *LRU* strategy accommodates the observations of the user sessions and the hot files presented in Figure 5 as it does not evict recently used files. The *ARC* cache competes as an improved LRU and in general slightly outperforms the *LRU* strategy.

The *Bélády* provides the best results, but as a theoretical construct that requires knowledge of the future we can only use it as an upper limit. Even for small cache sizes, this strategy often reaches the maximum hit ratio. This observation creates the assumption that an anticipatory eviction strategy that learns from the past might out-

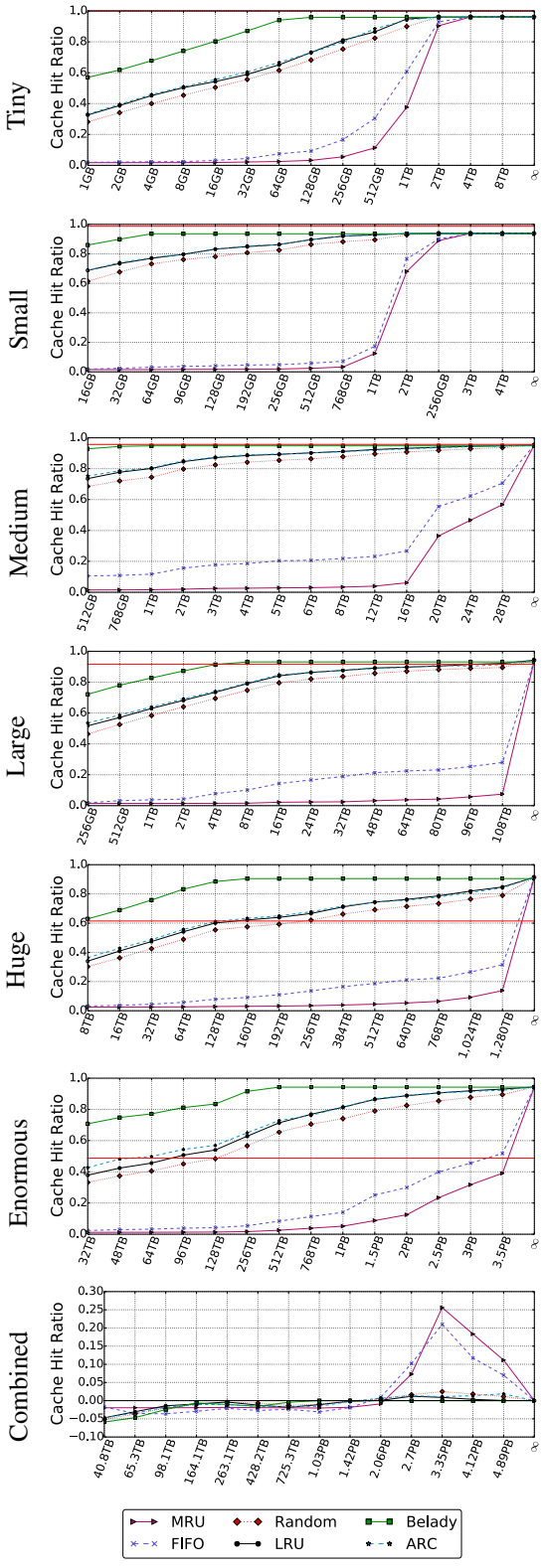

Figure 15: ECFS cache hit ratio evaluation. 2012 is used for cache warm up. Ratios are measured for 2013/2014. Horizontal red line marks ECMWF's hit ratio.

perform the other strategies. Domain knowledge as observable from the user sessions presented in Section 4.3 is available and could be fed to the caches.

Interestingly, the analysis of a big combined cache reveals that up to a cache size of about 2 PB, the single big instance in all cases provides a better hit rate. For a total cache size larger than 2 PB, the 6 subcaches provide better results for all cache strategies. All visualized hit ratios follow an upward trend for more capacity. The simulator can help to identify the achievable improvement of the hit ratio for extra cache capacity.

8 Discussion & Conclusion

This work analyzed log files and database snapshots to understand the behavior of ECFS and MARS, the two main archival systems at ECMWF, including the tape libraries that form the storage backbone of the two systems. The ECFS system resembles a typical archive and our findings underline the characterizations of previously studied systems [1, 21]. We analyzed the caching infrastructure of ECFS and provide a model and simulator to compare caches with different strategies and capacities. While ECMWF uses a lot of domain-specific knowledge which cannot be described algorithmically, we used the simulator to test basic strategies. It turned out that the *Adaptive Replacement Cache* (ARC) which evicts the *least recently used* and *least frequently used* files from the cache performs best. This conforms to our observation that files are often retrieved again after a short while (usually in the same user session). Coupling our test results with the knowledge of ECMWF will help in the future to further improve their cache hit rates.

Unlike ECFS, MARS opens the uncharted category of active archives that has not been thoroughly investigated until now. The object database uses a three-tiered architecture and a custom query language. All stored fields are subject to be read by computational models or experiments at any time. The investigated log files do not provide all the information necessary to fully characterize the user traffic. It is, for example, not possible to deduce the exact fields returned, although we know the queries. Nevertheless, it was possible to roughly describe the traffic and to assess the cache efficiency and the usage of the tape backend.

For ECFS, we saw read accesses on only 9% of the file corpus with a disk cache hit ratio of 86.7% during the observed timespan. Only 26,3% of the files on tape were ever read. The MARS logs do not reveal deeper information about which files and fields were accessed, but show that 95.1% of the requested fields were served from disk caches where only 2.2% of all requests included accesses to one or multiple tapes. Despite of this strong caching infrastructure, we have observed more than 9.5 million tape loads over a period of two years with 5% of the tapes being loaded more than 1,000 times.

The archival system's latency is not the primary bottleneck for the computations, as most operations run in batch and wait until the requested data is available. Nevertheless, since extensive queries that involve tapes can take several minutes, hours or even days, it is worthwhile to improve the average loading time of the tapes. Although the system is already well-configured, the 60 second tape reload rate of 14.8% and the prefetching analysis expose further potential for optimization.

The quality of weather forecasts constantly improves due to faster computers and improved computational models. At the same time, the requirements for storage infrastructure grow as well. Kryder's Law states that the areal density of bits on disk platters roughly doubles every two years [27]. While this was true for the last three decades, Rosenthal et al. forecast that the improvements in storage cost per bit for disk and tape will be much slower [24]. ECMWF faced a CAGR of 45% over the last years, which lately increased to 53% with the latest supercomputer. While this growth was maintainable with a constant budget during the last years, it might lead to the economical threats for long-term digital storage described by Baker et al. [5]. The consequence has to be an adjustment of scenario planning or implementing means to grow slower.

Due to a lack of studies, it is difficult to compare or generalize our results to other archival storage environments. Nevertheless, we believe that they are also relevant for other existing or future systems and that they can help to make the right design decisions. The reason is the observation that many large systems share at least some essential properties like the small read-to-write ratio and the overall architecture combining a large tape library with a relatively small disk-based cache. We developed a generic, extensible set of tools that can be applied to analyze workloads of archives and data centers. The cache simulation, for instance, helps to evaluate new caching strategies and to explore the impact of different eviction strategies and cache bucket sizes.

9 Closing Remarks

We are very grateful to the European Centre for Medium-Range Weather Forecasts for the opportunity to browse and analyze their log files and to share their valuable knowledge on building large scale archival systems.

The analyzed traces are stored at ECMWF and will be made available upon individual requests. The cache simulator, part of the scripts, and links to the trace files are available at https://github.com/zdvresearch/fast15-paper-extras.

References

[1] ADAMS, I., STORER, M., AND MILLER, E. Analysis of Workload Behavior in Scientific and Historical Long-Term Data Repositories. *ACM Transactions on Storage (TOS)* (May 2012).

[2] AGRAWAL, N., BOLOSKY, W., DOUCEUR, J., AND LORCH, J. A Five-year Study of File-system Metadata. *ACM Transactions on Storage (TOS)* (Oct. 2007).

[3] ATIKOGLU, B., XU, Y., FRACHTENBERG, E., JIANG, S., AND PALECZNY, M. Workload Analysis of a Large-scale Key-value Store. In *Proc. of the 12th ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)* (2012).

[4] BAKER, M., KEETON, K., AND MARTIN, S. Why Traditional Storage Systems Don't Help Us Save Stuff Forever. In *Proc. of the 1st Workshop on Hot Topics in System Dependability (HotDep)* (2005).

[5] BAKER, M., SHAH, M., ROSENTHAL, D., ROUSSOPOULOS, M., MANIATIS, P., GIULI, T., AND BUNGALE, P. A Fresh Look at the Reliability of Long-term Digital Storage. In *Proc. of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems (EuroSys)* (2006).

[6] BÉLÁDY, L. A Study of Replacement Algorithms for a Virtual-storage Computer. *IBM Systems Journal* (June 1966).

[7] BRESLAU, L., CAO, P., FAN, L., PHILLIPS, G., AND SHENKER, S. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proc. of the 18th IEEE International Conference on Computer and Communications (INFOCOM)* (1999).

[8] CHEN, Y., SRINIVASAN, K., GOODSON, G., AND KATZ, R. Design Implications for Enterprise Storage Systems via Multi-Dimensional Trace Analysis. In *Proc. of the 23rd ACM Symposium on Operating Systems Principles (SOSP)* (2011).

[9] CHROBAK, M., WOEGINGER, G., MAKINO, K., AND XU, H. Caching Is Hard – Even in the Fault Model. In *Algorithms – ESA 2010*, M. de Berg and U. Meyer, Eds., vol. 6346 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2010.

[10] COLARELLI, D., AND GRUNWALD, D. Massive Arrays of Idle Disks for Storage Archives. In *Proc. of the ACM/IEEE conference on Supercomputing (SC)* (2002).

[11] DOUCEUR, J., AND BOLOSKY, W. A large-scale study of file-system contents. In *Proc. of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)* (1999).

[12] ECMWF. ECMWF Data Handling System. http://www.ecmwf.int/en/computing/our-facilities/data-handling-system, Sept. 2014.

[13] EVANS, K., AND KUENNING, G. A Study of Irregularities in File-Size Distributions. In *Proc. of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)* (2002).

[14] FRANK, J., MILLER, E., AND ROSENTHAL, I. A. D. Evolutionary trends in a supercomputing tertiary storage environment. In *Proc. of the 20th IEEE International Symposium on Modeling, Analysis, and Simulation (MASCOTS)* (2012).

[15] GILL, P., ARLITT, M., LI, Z., AND MAHANTI, A. Youtube Traffic Characterization: A View from the Edge. In *Proc. of the 7th ACM SIGCOMM Conference on Internet Measurement (IMC)* (2007).

[16] GRAWINKEL, M., BEST, G., SPLIETKER, M., AND BRINKMANN, A. LoneStar Stack: Architecture of a Disk-Based Archival System. In *Proc. of the 9th IEEE International Conference on Networking, Architecture, and Storage (NAS)* (2014).

[17] HUANG, Q., BIRMAN, K., VAN RENESSE, R., LLOYD, W., KUMAR, S., AND LI, H. C. An Analysis of Facebook Photo Caching. In *Proc. of the 24th ACM Symposium on Operating Systems Principles (SOSP)* (2013).

[18] JAFFE, E., AND KIRKPATRICK, S. Architecture of the Internet Archive. In *Proc. of the ACM Israeli Experimental Systems Conference (SYSTOR)* (2009).

[19] LEUNG, A., PASUPATHY, S., AND MILLER, G. G. E. Measurement and Analysis of Large-scale Network File System Workloads. In *Proc. of the Annual Technical Conference (ATC)* (2008).

[20] LOS ALAMOS NATIONAL LABORATORY. Archive Data to Support and Enable Computer Science Research. http://institutes.lanl.gov/data/archive-data/, 2014.

[21] MADDEN, B., ADAMS, I., FRANK, J., AND MILLER, E. Analyzing User Behavior: A Trace Analysis of the NCAR Archival Storage System. Tech. Rep. UCSC-SSRC-ssrctr-12-02, University of California, Santa Cruz, Mar. 2012.

[22] MEGIDDO, N., AND MODHA, D. ARC: A Self-Tuning, Low Overhead Replacement Cache. In *Proc. of the USENIX Conference on File and Storage Technologies (FAST)* (2003).

[23] MEISTER, D., KAISER, J., BRINKMANN, A., CORTES, T., KUHN, M., AND KUNKEL, J. A study on data deduplication in hpc storage systems. In *Proc. of the ACM/IEEE Conference on Supercomputing (SC)* (2012).

[24] ROSENTHAL, D., ROSENTHAL, D., MILLER, E., ADAMS, I., STORER, M., AND ZADOK, E. The Economics of Long-Term Digital Storage. In *The Memory of the World in the Digital age: Digitization and Preservation* (2012), United Nations Educational, Scientific and Cultural Organization (UNESCO).

[25] SAROIU, S., GUMMADI, P., DUNN, R., GRIBBLE, S., AND LEVY, H. An Analysis of Internet Content Delivery Systems. In *Proc. of the 5th Conference on Operating Systems Design and Implementation (OSDI)* (2002).

[26] STORER, M. W., GREENAN, K. M., MILLER, E. L., AND VORUGANTI, K. Pergamum: Replacing Tape with Energy Efficient, Reliable, Disk-Based Archival Storage. In *Proc. of the 6th USENIX Conference on File and Storage Technologies (FAST)* (2008).

[27] WALTER, C. Kryder's Law. *Scientific American*, 293 (2005).

[28] WELCH, B., AND NOER, G. Optimizing a hybrid SSD/HDD HPC storage system based on file size distributions. In *Proc. of the 29th IEEE Conference on Mass Storage Systems and Technologies (MSST)* (2013).