The Failure Trace Archive: Enabling the Comparison of Failure Measurements and Models of Distributed Systems

Bahman Javadi[a,*], Derrick Kondo[b], Alexandru Iosup[c,**], Dick Epema[c]

[a]*School of Computing, Engineering and Mathematics, University of Western Sydney, Australia*
[b]*INRIA, France*
[c]*Faculty of Engineering, Mathematics and Computer Science TU Delft, The Netherlands*

Abstract

With the increasing presence, scale, and complexity of distributed systems, resource failures are becoming an important and practical topic of computer science research. While numerous failure models and failure-aware algorithms exist, their comparison has been hampered by the lack of public failure data sets and data processing tools. To facilitate the design, validation, and comparison of fault-tolerant models and algorithms, we have created the Failure Trace Archive (FTA)—an online, public repository of failure traces collected from diverse parallel and distributed systems. In this work, we first describe the design of the archive, in particular of the standard FTA data format, and the design of a toolbox that facilitates automated analysis of trace data sets. We also discuss the use of the FTA for various current and future purposes. Second, after applying the toolbox to nine failure traces collected from distributed systems used in various application domains (e.g., HPC, Internet operation, and various online applications), we present a comparative analysis of failures in various distributed systems. Our analysis presents various statistical insights and typical statistical modeling results for the availability of individual resources in various distributed systems. The analysis results underline the need for public availability of trace data from different distributed systems. Last, we show how different interpretations of the meaning of failure data can result in different conclusions for failure modeling and job scheduling in distributed systems. Our results for different interpretations show evidence that there may be a need for further revisiting existing failure-aware algorithms, when applied for general rather than for domain-specific distributed systems.

1. Introduction

As a consequence of increasing presence, complexity, and scale of distributed systems, resource failures have become inevitable. Failures can have serious consequences for applications running on these systems: performance degradation and loss of useful work for scientific applications, corruption of data,

*Corresponding author. Telephone: +61-2-9685 9181; Fax: +61-2-9685 9245
**Corresponding author. Telephone: +31-15-2784433; Fax: +31-15-2786632
Email addresses: `b.javadi@uws.edu.au` (Bahman Javadi), `A.Iosup@tudelft.nl` (Alexandru Iosup)

violation of service-level agreements, and even large losses of customers and revenue [1]. Although many models and algorithms exist for analyzing, predicting, and resolving failures [2, 3, 4, 5, 6, 7, 8], these models and algorithms are validated using failure traces of a single or a very limited number of systems. Moreover, for the few studies that use failure traces collected from multiple systems, the data sets are rarely publicly available. Thus, the field of failure models and fault-tolerant algorithms is severely fragmented, and the comparison and cross-validation of proposed models is difficult if not impossible. To remedy this situation, we have created, and we present in this work the Failure Trace Archive (**FTA**), which comprises publicly available traces of many different types of parallel and distributed systems, along with public tools for their analysis.

There are numerous causes that can lead to failures in real-world distributed systems, broadly derived from increasing system functionality, complexity, and scale. When system functionality increases, the immaturity of the software stack, and various security threats and attacks, can lead to unmaskable failures. When the complexity of the system grows, system misconfiguration and even scheduled downtime to update the system become regular sources of failure. When systems are expanded, system overload and even natural disasters affecting one part of the system may trigger cascading failures that can bring the entire system down. For example, between 2008 and 2010, both Facebook and Twitter experienced repeatedly downtime when overloaded [9, 10]. Overloads were also the cause of downtime for the Microsoft email service Hotmail, at the end of 2010 [11]. Even Goodreads, a popular social network for book readers, became unusable on Aug 20, 2012, due to overloads.

We create the FTA as a community archive, an approach that has been recognized as useful for sharing data and that has been employed by several communities in the computing domain. For example, the parallel computing community has built the Parallel Workloads Archive [12], the grid computing community has created the Grid Workloads Archive [13], etc. Efforts such as the Repository of Availability Traces [14], the Computer Failure Data Repository [15], and the Desktop Grid Failure Traces [4] have led to making failure-related data public, but did not establish the premise of a community archive for distributed computing systems. In particular, they did not build a common format for storing failure-related data, did not provide a working toolbox for uniformly processing and interpreting failure-related data, and did not publish a sufficient number of data sets to encompass a large variety and number of distributed systems. In contrast to these early efforts, our main contributions are as follows:

1. We survey the presence and impact of failures in real-world distributed systems (Section 2).
2. We design a public failure trace archive, creating a standard format for failure traces, a toolbox for uniformly processing and interpreting failure-related data, and a simulator that facilitates comparative trace analysis (Section 3). Currently, the archive includes 20 traces across 8 classes of distributed systems. We also present in this section our experience with numerous use cases of the FTA and our predictions regarding the applicability of the FTA for future distributed systems.
3. Using the toolbox, we uniformly analyze and model two failure characteristics across several types of distributed systems (Section 4).
4. We show that differences in the interpretation of failure-related data can change significantly the analysis and modeling results based on derived from the data (Section 5).
5. Using the simulator, we evaluate the effect of differences in the interpretation of failure traces on the job scheduling in two distributed systems (Section 6).

We have introduced the FTA in a preliminary conference paper [16], which we expand upon in this work with more context, new traces, more in-depth analysis, and new simulation results. In particular, we add more examples of failures in real-world distributed systems and several new traces in the archive, both in the survey in Section 2 and in the significantly increased amount of traces currently shared through the FTA. We also provide an overview of the use of the FTA in different aspects including design, testing, and procurement of distributed systems. Last, we provide a new simulator to enable researchers to study the impact of failures in their proposed fault tolerance models and algorithms, along with a case study to show the effect of trace interpretation on job scheduling in distributed systems.

The remainder of the work is organized as follows. In Section 2, we recall the terminology concerning failures in distributed systems and we survey many failures that have occurred in operational systems over the years. We describe the Failure Trace Archive structure, and past and future use, in Section 3. In Section 4, we present a statistical analysis of several types of distributed systems, based on a selection of data sets from the FTA. We then present the difference of interpretation for different data sets in Section 5. We evaluate the effect of trace interpretation on the job scheduling in two different distributed systems using trace-based simulations in Section 6. We describe related work in Section 7. Finally, we summarize our findings and present future directions in Section 8.

2. Background on Failures in Large-Scale Systems

We introduce in this section the terminology on failures used throughout this work. We also provide more motivation for our work on failures, through a selective survey of the presence and impact of failures in large-scale distributed systems.

2.1. Terminology

Throughout this work, we follow the basic concepts and definitions associated with system dependability as summarized by Avizienis et al. [17]. We also recommend the topical survey of Salfner et al. [18]. The basic threats to reliability are failures, errors, and faults occurring in a system. A *failure* is an event that makes a system fail to operate according to its specifications. A failure is observed as a deviation from the correct state of the system. We call the continuous period of a service outage due to a failure an *unavailability interval*. A continuous period of availability is called an *availability interval*. An *error* is a part of the system state that may lead to a failure. Some errors may not be visible from outside of the system, that is, they may not reach the external state of the system and thus cause failures; such errors are said to be *dormant*. Errors that do cause failures are said to be *active*. The root cause of an error is a *fault*.

2.2. Failures in Real-World Distributed Systems

Although we expect distributed systems to be highly available and reliable, the reality is that unmaskable failures occur often and with important consequences. In this section, we survey chronologically the presence and impact of failures in large-scale distributed systems. This survey focuses on a selection of exemplary failures affecting a large number of people or large-scale services based on distributed systems. Moreover, the survey is an important motivation for our work and has guided the selection of traces investigated in this work (see Section 3.4).

Our selective survey, albeit not comprehensive, covers in fifteen examples a variety of application domains in which failures occurring in distributed systems cause significant service issues. Our examples cover High-Performance Computing (HPC), Internet-based file-sharing and content distribution, Internet-based online gaming, general IT infrastructure, computer science research, social networking, online retail, and online stock trading. We also cover numerous causes that can lead to failures in real-world distributed systems: scheduled downtime, system overload, system misconfiguration, immaturity of the software stack, security threats and attacks, natural disasters, etc.

The production multi-cluster grid Grid3 [19] (now the Open Science Grid) was experiencing a job failure rate of about 30% in 2003 where 90% of failures were due to site problem (e.g., disk failure) [19, 20]. One of the European ICT infrastructures servicing hundreds of scientists, Grid'5000, suffered between 2005 and 2007 from cascading and catastrophic failures involving hundreds of computers [21]. Among the causes, immaturity of the software stack is a reasonable explanation that is supported by later evidence into the quality of grid middleware over the years [22]. Another possible cause is the abuse of the scheduling system by redundant submissions of batch jobs [23].

Overloads affect even peer-to-peer systems, although they are by design scalable. For example, since 2003, the BitTorrent file-sharing system [24] can experience poor performance during severe overloads (*flashcrowds*). Intuitively, because in peer-to-peer systems the users provide additional service capacity while being online, a (long) flashcrowd can lead to a beneficial accumulation of (bandwidth) capacity rather than to poor performance. However, even through 2009 and 2010 the performance of BitTorrent users during flashcrowds could be up to an order of magnitude lower than the performance observed in normal conditions [25].

The operators of World of Warcraft, a massively multiplayer online gaming service, have scheduled since 2003 periods of downtime of several hours weekly, for updating and managing their world-wide pool of over 200 clusters. Assuming an average of 4 hours of downtime, the maximum availability of World of Warcraft was and still is under 97%, which affects negatively their players. Scheduled downtime is currently the de-facto standard for the online gaming industry.

Natural disasters struck USA and Italy in 2003 [26, 27], causing severe blackouts and thus failure of the IT infrastructure. It is doubtful that redundant capacity and operational protocols could have masked failures of this magnitude.

The Akamai content distribution services were unavailable in May 2004 for over an hour, due to large-scale denial-of-service attacks [28]. Several major websites, including eBay, Yahoo!, and Google, which were relying on Akamai's distributed infrastructure for content distribution, also suffered downtime.

For shared infrstructure, a large fraction (about 20%) of the PlanetLab resources were unavailable to researchers several times during 2004 [29]. Moreover, the performance of the system could drop significantly during overload periods.

Between 2008 and 2010, both Facebook and Twitter experienced repeatedly downtime when overloaded [9, 10]. Overloads were also the cause of downtime for the Microsoft email service Hotmail, at the end of 2010 [11]. Even the Goodreads social network for book readers became overloaded on Aug 20, 2012.

Network misconfiguration led to downtime of several Amazon services in April 2011 [30]. Both BATS (March 2012) and Nasdaq (May 2012, during the Facebook IPO launch) failed due to algorithmic problems but also misconfiguration to respond to request overload [31]. The duration of the matchmaking algorithm used for trading exceeded the maximal duration of validity of requests, which allowed the distributed requests to be updated and ultimately triggered a loop in the process.

The cluster architecture of CCP's EVE Online, an online massively multiplayer online game, has crashed repeatedly between 2011 and 2013 [32]. Although both the hardware and the distributed middleware, and even the application design were upgraded periodically, the most important failures continued to be caused by player flashcrowds. As a consequence, many players lost assets that took years to develop.

A natural disaster that struck India in 2012 emphasizes the difficulty of masking large-scale (correlated) failures. As a consequence of lack of rain, the Indian power grid collapsed [33]. Pre-established fail-overs to different parts of the system failed in cascade, due to overload. Over 300 million people were left without access to the power and IT infrastructure for days or more.

3. Overview of the Failure Trace Archive

The Failure Trace Archive (FTA) can be used in many ways. First, the FTA allows the comparison and cross-validation of a fault-tolerant model or algorithm across identical trace data sets. Second, it allows the evaluation of the generality of a model or algorithm across different types of resources (in terms of reliability or user base, for example). Third, it allows for the evaluation of the generality of a failure trace, i.e., to determine whether measurements are biased to a particular platform or middleware. Fourth, it allows for the determination of which trace data set is most interesting or applicable for a given algorithm or model. Fifth, it allows for the analysis of the evolution of availability in different systems across long timescales. Sixth, it allows for the integration of failure models with other types of models (such as workloads). Seventh, it facilitates the incorporation of traces with a common format into fault simulators or emulators for model or algorithm evaluation.

3.1. Archive Format

In our experience, the majority of time in measurement and modeling studies is spent in parsing and interpreting the measurements. To accelerate this processing and analysis for others, we have parsed and interpreted 20 diverse distributed systems in a standard format. Here we describe the rationale of the format.

The majority of our collection of traces record times of failures for *resources*, and contain an alternating time series of availability and unavailability intervals. As such, our format is resource-centric (versus job-centric or user-centric) with respect to failures of individual nodes or components of nodes, such as memory, CPU, or hard disks. We believe the format is also applicable to failures of services deployed on top of resources. However, our format does not explicitly describe higher-level failures, such as job failures, though potentially the FTA format could be extended for this type of failure or perhaps combined with the Grid Workload Archive format [13]. Measuring and understanding the relationship between lower-level failures (for example, of nodes or components) to higher-level failures (for example, jobs) is an area for future research.

The trace format is organized hierarchically as follows: Platform $\rightarrow$ Node $\rightarrow$ Component $\rightarrow$ Event Trace. Figure 1 depicts the structure of the FTA, where boxes represent database tables. We summarize the meaning of each table below. Table names are shown in bold.

- A **platform** contains a set of nodes. Examples of a platform include desktop PC's at Microsoft, or nodes in the LANL[1] clusters.

[1]Los Alamos National Laboratory

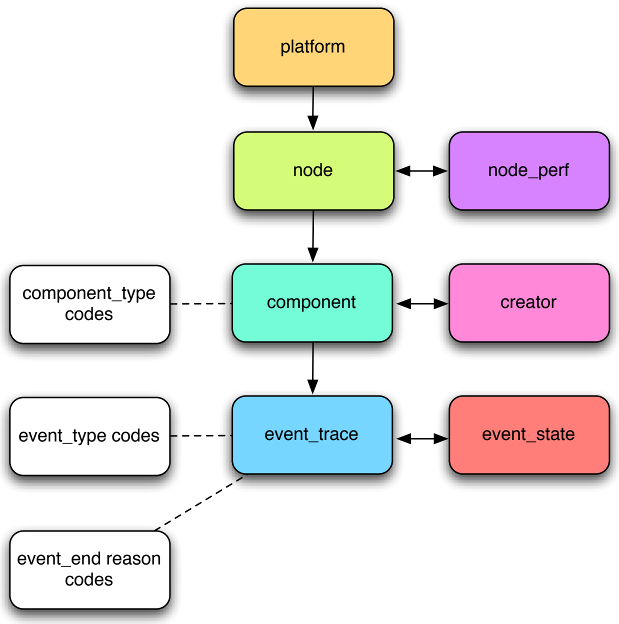


Figure 1. Overview of the FTA structure.

- A **node** contains a set of components, which is a software module or hardware resource of the node. Each node can have several components (e.g., CPU speed, available memory, client availability), each of which has a corresponding trace.
- The **node_perf** describes the node performance, as measured through benchmarks, for example.
- A **component** describes attributes of a software module or hardware resource of a node.
- A **creator** is the person responsible for the trace data set. This table stores details about data copyright, and about projects and published material that use the data.
- An **event_trace** is the trace of an event, with all of corresponding timing information (e.g., start and end times).
- The **event_state** is the state corresponding to an **event_trace**. For example, for CPU availability, the event_state could be the idleness of the CPU. For host availability, it could be the monitoring information associated with the event.

In addition, we have codes that correspond to different types of components (for example, memory, CPU, hard disk), events (for example, availability or unavailability), and event reason codes (for example, disk crash and CPU overheating).

The best test of a format is its application to real data sets for different types of systems with different types of failures measured in different ways. We applied this format to nine systems ranging from desktops on the Internet to supercomputing clusters. The types of failures included host, CPU, and even service failures. These failures were measured using a variety of methods, such as periodic probing, event notification, load measurement, and even human observation. Given that all of these data sets

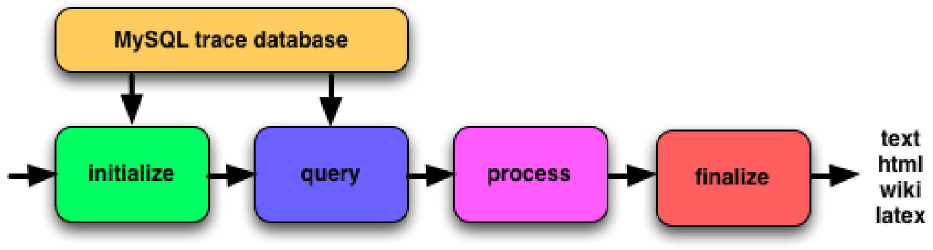


Figure 2. FTA Toolbox Design.

could be presented in this format with ease, we believe the format is good first step towards a standard. To anticipate future extensions of the format, we have several generic tables with double and string field that can contain additional new information should it arise.

3.2. FTA Toolbox

We implemented a FTA toolbox to facilitate the comparative analysis of failure traces (see Figure 2). The toolbox is implemented in Matlab, and uses several open-source Matlab packages, such as the Mysql and DataTable packages.

The toolbox takes as input four functions for initializing, querying, processing, and finalizing the data analysis. The initialization and query stages allow one to extract the necessary data from traces located in a MySQL database into Matlab in-memory data structures. By contrast to loading entire data sets into memory from large files, this method allows one to extract into memory only the data that is needed for processing.

Initialization and querying is separated from processing to allow expensive initialization queries to be conducted only once, after which any amount of processing can be done. Also, this separation allows the same initialization and queries to be used for many different processing functions. This facilitates code reuse.

The results of initializing and querying are then passed to the processing function. This function is run across each of those results. The processing output is then fed into the finalize function, which produces tables in latex, HTML, text, and wiki formats using the DataTable module. All graphs and tables in Sections 4 and 5 were produced using the FTA toolbox.

3.3. FTA Simulator

The GridSim is a framework which allows modeling and simulation of entities in parallel and distributed computing systems for performance evaluation purposes [34]. Although the processing nodes (i.e., a machine) within a resource in GridSim can be heterogeneous in terms of processing capability and configuration, there is no support to simulate resources in the presence of failures. In order to have this feature, we developed a set of packages for GridSim simulator to generate a list of failures based on the FTA format. To do this, we equipped GridSim with a failure injection mechanism that is able to simulate failure events collected in the FTA data sets in various distributed systems. It basically reads the **event_trace** tabbed file and generates a list of events that show the availability/unavailability patterns for each machine for a given resource (i.e., a resource may have more than one machine). The list of resources also is taken from the **node** tabbed file. In the case of resource failure, the target node will

Table 1. Summary of the data sets in the Failure Trace Archive.

Trace	Type	# of Nodes	Target Component	Period	Year
lanl05	SMP, HPC Clusters	4,750	host	9 years	1996-2005
g5k06	Grid	1,288	host	1.5 years	2005-2006
microsoft99	Desktop	51,663	host	35 days	1999
websites02	Web servers	131	host	8 months	2001-2002
pl05	P2P	692	host	1.5 year	2004-2005
ldns04	DNS servers	62,201	host	2 weeks	2004
overnet03	P2P	3,000	host	2 weeks	2003
nd07cpu	Desktop Grid	700	CPU, host	6 months	2007
skype06	P2P	2,081	host	1 month	2005
sat09	Desktop Grid	226,208	CPU	1.5 years	2007-2009
pnnl07	HPC Cluster	980	host, network	4 years	2003-2007
ucb94	Desktop Grid	80	CPU	46 days	1994
sdsc03	Desktop Grid	275	CPU	1 month	2003
lri05	Desktop Grid	40	CPU	1 month	2005
deug05	Desktop Grid	40	CPU	1 month	2005
cae06	Grid	686	host	35 days	2006
cs06	Grid	725	host	35 days	2006
glow06	Grid	715	host	33 days	2006
teragrid06	Grid	1001	host	8 months	2006-2007

stop working for the duration of unavailability interval and start working again for the given availability interval.

The failures are simulated in the node level (i.e., the same level as GridSim simulator) where we provided some fault-tolerance algorithms such as checkpointing mechanisms to analyze the effect of failures on job scheduling. In Section 6, we use this simulator with a perfect checkpointing mechanism to study the impact of the resource failures on job scheduling.

3.4. FTA Traces

The FTA currently has 20 formatted data sets, which are listed in Table 1, and 6 others currently with raw data only. Overall, we study in this work traces coming from distributed systems used in various application domains: HPC, Internet operation, Internet-based file-sharing, various other online applications. The data represent 8 types of distributed systems, including multi-cluster grids, HPC clusters, and large-scale P2P systems. The FTA traces represent a collection that covers significantly more application domains and system types, in comparison with the related archives described in Section 7; in particular, we have more than doubled the number of traces shared through the FTA in 2010 [16]. In the remainder of this section, we describe each formatted data set and the measurement method used for its collection. We further study in this work, in Sections 5 and Section 6, how the interpretation of the meaning of data collected with these methods can result in different conclusions for failure modeling and job scheduling in distributed systems, respectively.

lanl05 is a data set of 22 HPC systems at Los Alamos National Laboratory. It contains a record for every failure that happened in these systems as well as the root cause [3]. The **g5k06** data set is a trace of a computational grid platform in France (i.e., Grid'5000) which consists of 9 sites, 15 clusters and more than 2,500 processors [5]. The data was collected by periodic inspection and logging of each node's status through the grid middleware called OAR. The **microsoft99** data set contains log files of 51,663 desktops PCs at Microsoft Corporation where their reachability was determined with a ping every hour [2]. The data set of **websites02** was derived from probe-based measurements where a single machine at Carnegie Mellon sent a HTTP file request to web servers periodically every 10 minutes [35]. **pl05** consists of trace data measured between all pairs of PlanetLab nodes using pings every 15 minutes [36]. The **ldns04** data set includes the probe results of 62,201 local DNS servers where the inter-arrival time of the probes followed an exponential distribution with mean of one hour [37].

The **overnet03** data set is a probe-based measurement conducted over the Overnet peer-to-peer file-sharing system [38]. In this data set, the availability of 3,000 hosts was checked every 20 minutes. The **nd07cpu** data set contains traces recorded by Condor from the desktop systems at the University of Notre Dame [6]. The data set is comprised of time-stamped CPU load and idle times of each system, recorded every 16 minutes. The **skype06** data set is collected by application-level pings of nodes in the Skype superpeer network, every 30 minutes [39]. **sat09** is the data set of the SETI@Home project where the BOINC client [40] is instrumented to collect CPU availability traces from more than 200,000 hosts over the Internet [41]. We define CPU availability to be a binary value indicating whether the CPU was free or not. The traces record the time when CPU starts to be free and stops.

The **pnnl07** is traces of hardware failures on the HPC system with 980 nodes including dual Itanium-2 processors at Pacific Northwest National Laboratory (PNNL) [42]. For each hardware failure, the data set includes a time-stamp, a hardware identifier, the component that failed, a description of the failure, and the repair action taken. **ucb94** is the data set of the a workstation cluster used by UC Berkeley CAD group where information about CPU, memory, disk, keyboard and mouse activity were logged every two seconds [43]. A host was considered available in this measurement, if the average CPU usage over the past minute was less than 5%, and there had been no keyboard/mouse activity during that time.

The **sdsc04** data set consists of availability traces of 275 hosts at the San Diego Supercomputer Center (SDSC) while running Entropia's DCGrid software [44]. **lri05** and **deug05** are traces from desktop PC's at the University of Paris South, and ran the open source XtremWeb [45] desktop grid software [4]. They only difference between two data sets is the type of users where the `lri05` was a cluster used by a computer science research group for running parallel applications and benchmarks, while `deug05` consisted of desktop PC's in classrooms used by first-year undergraduates. The **cae06**, **cs06**, and **glow06** data sets [46] have been collected by the Condor Team from the test (the former two) and production (the latter) Condor pools at University of Wisconsin-Madison; the latter Condor pool was part of an international Grid system working for the CMS experiment at CERN. Finally, the **teragrid06** data set [46] includes failure traces collected from the NCSA Linux cluster, which was ranked as the best site of the TeraGrid system by the NSF Cyberinfrastructure User Survey 2005. They have crawled with a sampling interval of 5 minutes the online status page of the NCSA Linux cluster, containing general, job, and node use and availability information.

3.5. Discussion: On the Current and Future Use of the FTA

In this section, we discuss current and future use of the FTA. For current use, we discuss impact on research, practice, and education. For future use, we discuss the extension of FTA with data collected from several upcoming, non-traditional distributed systems.

Research impact. To assess the research impact of the FTA, we have followed the citation record of our FTA article [16] since the publication of our first report about the FTA at the CCGrid conference, in May 2010. Overall, the FTA article has attracted over 65 citations, as reported by Google Scholar in March, 2013. Google Scholar indicates that the FTA has been used for work published in conferences such as IEEE IPDPS, ACM HPDC, and ACM SC; and in journals such as CCPE, IEEE Internet Computing, and JPDC. Thus, we conclude that the FTA has been useful for a large part of the communities focusing on theoretical and applied research in distributed systems. For the future, we would like to promote the use of FTA by two other communities, related to systems reliability and distributed systems (USENIX-related).

Practical impact: use in system design and operation. The content of the FTA restates that many large-scale distributed systems experience failures. System designers and administrators can use FTA data to design, validate, and evaluate new algorithms, methods, and practical deployments. FTA data and tools are particularly relevant for work in topics such as backup and checkpointing, replication, prediction for proactive and reactive resource management [47], scheduling in general, security [48], storage management [49], and data availability and durability in general.

FTA traces have been used for work on checkpointing for parallel jobs [50], where the failure of nodes needs to be proactively compensated through periodic saving of application state and other checkpointing strategies. Using FTA traces, the authors perform extensive simulations of various checkpointing strategies.

FTA data can be used for various scheduling problems [51, 48, 52]. Scheduling in systems with volatile resource, such as volunteer computing environments, may not be able to use the same approaches developed for more stable systems. For example, a study of scheduling strategies for volunteer computing uses FTA traces to show that "the commonly used BOINC scheduling algorithms are unable to enforce fairness and project isolation" [51]. They also show that lack of coordination in the exploitation of shared resources can be inefficient and socially unfair.

Practical impact: use in system testing and procurement. System testing and procurement of large-scale distributed systems, which is an area that can still see many improvement in both theory and practice, can use FTA data directly for testing and dimensioning in general, including in tools for fault injection, testing with varying node availability, simulation of what-if scenarios, etc. Moreover, both system testing and procurement can use failure models derived from FTA data.

Among the simulators that use FTA data [53, 54], SimGrid was extended to support volunteer computing platforms [53]. The main challenge encountered by the designers of SimGrid is the scale of the simulated environments; they showed through experiments using FTA traces that their proposed scalability mechanisms perform well in practice.

The FTA traces have been instrumental in the development of various failure models [46, 55, 56]. In a recent study [46], we analyzed and model the time-varying behavior of failures in large-scale distributed systems. We used nineteen failure traces from the FTA, concerning production large-scale distributed systems, including grids, P2P systems, DNS servers, web servers, and desktop grids. We showed that time-correlated failures occur often and have an important impact on the performance and availability of such systems. We also proposed a statistical model that characterizes the duration of peaks, the peak inter-arrival time, the inter-arrival time of failures during the peaks, and the duration of failures during peaks. We showed that, for the systems we studied, our model characterizes over 50% and up to 95% of the downtime of these systems.

Practical impact: use in education. The lack of professionals who can work in distributed systems has become increasingly visible in the US and EU markets in the recent years, coupled with the growth in the number and size of data centers. As a consequence, (large-scale) distributed systems play a prominent role in the updated Computer Science Curricula 2013 (CS2013, Strawman draft) published by the ACM and IEEE Computer Society at the beginning of 2012 [57]. We expect most of the universities with strong ties to the ACM and IEEE to adopt this new curriculum guidelines and, already, many universities have already started having undergraduate and graduate-level courses in large-scale distributed systems [58, 59, 60]. Thus, we believe the FTA can become an important resource for computer science curricula in academic education.

We target with the FTA university-level courses that teach the use of grids and Clouds, large-scale distributed systems, performance analysis through simulation, and performance modeling. The reports, the tools, and the data included in the FTA can greatly help the instructors of such courses. Specifically, the data in the repository can be used for in-class demonstrations and student assignments, the detailed analysis can be used to illustrate concepts related to system operation, etc. The tools may be used to build new analysis and simulation tools.

We have used material from the FTA in several undergraduate and graduate-level courses at TU Delft. For example, we have used FTA analysis results in the first-year B.Sc. course Computer Organization[2], to exemplify the concept of system reliability. Similarly, we have used FTA analysis results in the M.Sc. course Cloud Computing[3], to explain the concept of cascading failures and to exemplify potential problems induced by multi-tenancy.

Future use of the FTA: support for upcoming distributed systems. Although the direction in which distributed systems is difficult to predict, our study [61] of the past decade's trends in scientific workloads has identified trends such as a shortening of job durations and loose coupling of jobs. These trends, coupled with the rise to prominence of the "data deluge" (Big Data) [62], indicate several possible directions in which distributed systems may evolve.

First, systems may become increasingly parallel, which means incorporating in distributed systems highly-parallel GPUs such as NVIDIA's Tesla, heterogeneous multi-cores such as Intel MIC, and even generic architectures such as the FPGA/MISD machines from Maxeller. Moreover, distributed systems may even start to incorporate hybrid processing elements, such as the CPU and GPU devices in Nvidia's Project Denver. The current FTA format could accommodate data collected from such architectures through its **component** and **component_type** elements. Important challenges here will be making the **node_perf** information meaningful across multiple classes of machines and facilitating the grouping of various members of the same resource family.

Second, systems operated by small companies or systems that exhibit high workload variation may increasingly rely on external resources, perhaps acquired from Cloud computing infrastructure such as Amazon EC2. Such hybrid systems will be heterogeneous and distributed in nature, and may raise new challenges in recording failure information about the Internet and about the acquired machines. The

[2]TU Delft course TI1400 Computer Organization.
[3]TU Delft course IN4392 Cloud Computing.

Table 2. Statistics of availability intervals for different data sets (values in hours.)

Trace	Mean	TrMean	Median	Std	CV	IQR	Max	Min	Skewness	Kurtosis	No.
lanl05	1779.99	1208.09	280.28	3462.33	1.95	1593.37	34480.23	0.02	3.09	14.29	19874
g5k06	32.41	18.41	7.09	94.24	2.91	24.07	10157.73	0.00	15.06	695.83	294318
microsoft99	67.01	40.39	10.00	138.47	2.07	55.00	840.00	1.00	3.40	15.80	526078
websites02	11.85	5.17	0.83	40.10	3.38	5.17	1196.55	0.00	9.02	135.89	47843
pl05	159.48	71.42	1.71	475.61	2.98	35.60	6051.49	0.00	4.91	34.26	24928
ldns04	140.93	125.79	28.29	193.39	1.37	213.47	559.27	0.00	1.24	2.97	223596
overnet03	2.29	1.48	1.33	4.63	2.02	1.00	120.11	0.00	8.03	113.34	33443
nd07cpu	13.73	5.46	1.07	60.05	4.37	7.11	3783.57	0.00	25.49	1228.74	134176
skype06	16.27	10.12	5.11	34.57	2.12	11.87	465.95	0.00	4.81	34.38	29217

current FTA format could accommodate machine heterogeneity, through its **component** and **component_type** elements. However, information about these machines may only be available through a view from the user, where, similarly to the view provided by desktop grids, details outside the shared resources (e.g., of the physical machine) are hidden from the user. Although obtaining failure information from such distributed systems raises important challenges, early solutions already exist, for example as test suites [63] or as large-scale observations [64].

Third, a resource of future interest will be storage. We have not yet included traces collected from traditional multi-tier distributed storage systems into the FTA, although the `overnet03` traces represent systems with single tier, heterogeneous, P2P, file-based storage. Extending the FTA format for the latter is a topic of future work.

4. Analysis of FTA Traces

In this section, we analyze the first nine data sets of FTA in two steps[4]. First, we inspect the basic statistics for two failure characteristics, duration of availability and unavailability intervals. Second, we fit distributions for modeling failures in terms of probability distributions of availability and unavailability intervals. Third, we present a qualitative comparison of failure characteristics in distributed systems.

4.1. Basic Statistics

We focus in this section on various statistics for the availability and unavailability intervals: the mean and the trimmed mean (defined as the mean value after discarding the top 10% of the values), the median, the standard deviation (std), the coefficient of variation (CV), the interquartile range (IQR), the maximum and minimum, the skewness (the third moment), the kurtosis (the fourth moment), and the number of intervals. Tables 2 and 3 summarize the results obtained for each data set for availability and unavailability intervals, respectively. These tables contain three types of descriptive statistics. Statistics of the first type (mean, median, and trimmed mean) reflect the central tendency of the distributions; statistics of the second type (CV, IQR, minimum, maximum) measure the spread of the distribution; and statistics of the third type (skewness, kurtosis) reflect the shape of the distribution.

[4]These nine data sets cover all the distributed system types currently represented in the FTA.

Table 3. Statistics of the unavailability intervals for different data sets (values in hours.)

Trace	Mean	TrMean	Median	Std	CV	IQR	Max	Min	Skewness	Kurtosis	No.
lanl05	5.88	1.67	0.97	78.39	13.32	1.98	5325.70	0.00	43.96	2289.91	23451
g5k06	7.41	0.94	0.05	60.24	8.13	0.19	6314.95	0.00	26.26	1237.26	294145
microsoft99	16.49	9.15	2.00	46.50	2.82	14.00	840.00	1.00	8.52	105.12	493687
websites02	1.18	0.49	0.17	22.92	19.46	0.34	3311.51	0.00	111.03	14311.32	47714
pl05	49.61	12.86	0.50	269.90	5.44	6.36	9329.47	0.00	15.10	340.33	24236
ldns04	8.61	5.47	2.28	20.68	2.40	7.82	533.22	0.00	8.62	123.06	161395
overnet03	11.98	4.00	0.33	36.82	3.07	1.67	167.83	0.00	3.66	15.11	35449
nd07cpu	4.25	0.47	0.27	62.83	14.77	0.36	3616.70	0.04	33.72	1307.29	134026
skype06	14.31	9.45	6.16	30.23	2.11	14.30	596.03	0.02	6.26	62.72	27136

The results reveal that the ratios between the mean and the median for the availability and unavailability intervals are quite different across the data sets. This indicates that single-parameter distributions might not be a good option for a failure model. This can be confirmed by the skewness and kurtosis values that indicate that both availability and unavailability intervals are well modeled by distributions that are right-skewed and long-tailed. Moreover, the results indicate that unavailability distributions are more highly right-skewed and have longer tails than the availability distributions.

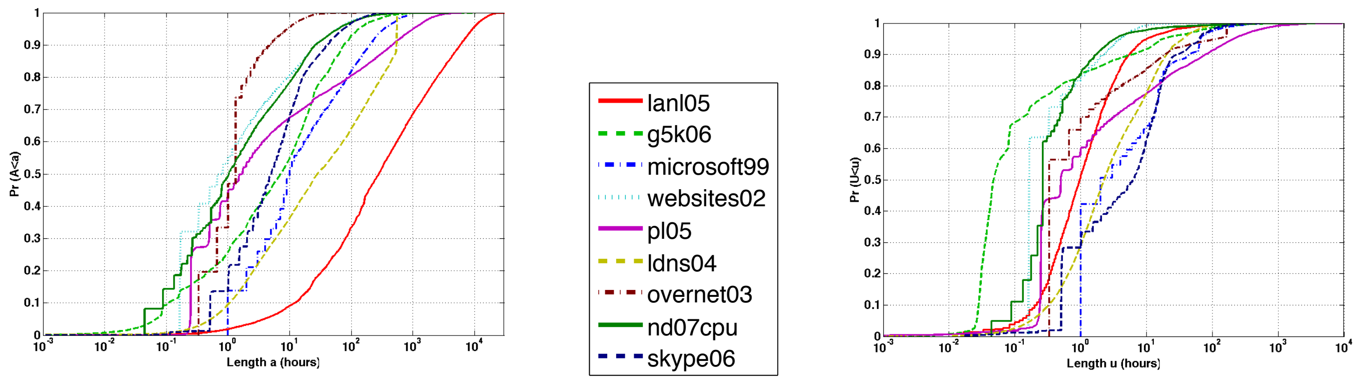
The unavailability intervals are more variable than the availability intervals, as indicated by the higher values of the CV and the lower values of the trimmed mean (the cut 10% of the data accounts for much of the difference between the mean and the trimmed mean). This further emphasizes the need for distributions with more degrees of freedom, e.g., phase-type distributions, to model unavailability intervals for these data sets.

4.2. Toward Failure Models

In this section, we refer to the distribution of availability and unavailability intervals as the failure model. The cumulative distribution functions (CDFs) of the availability and unavailability intervals are plotted in Figures 3(a) and 3(c), respectively. We find that the data sets differ significantly in scale and shape of these distributions.

We have conducted parameter fitting for various distributions, namely the Exponential, Weibull, Pareto, Log-normal, and Gamma distributions. The fitting was done using maximum likelihood estimation (MLE). We adopted two goodness of fit (GOF) tests, the Kolmogorov-Smirnov (KS) and the Anderson-Darling (AD) tests, to evaluate the distribution fits. The results of both tests are reported in terms of the p-values for availability and unavailability distributions in Tables 4 and 5, respectively. The p-value we report is the average of 1000 p-values, each of which was computed by randomly selecting 30 samples from a data set—this is a standard method [65, 41] for computing p-values when the number of samples is high.

The exponential function seems to be far from the underlying distributions. However, it could be a good fit for the availability distributions of `microsoft99`, `overnet03` and `skype06` and the unavailability distributions of `microsoft99`, `ldns04`, and `skype06` as well. So, the `skype06` data set with the exponential failure model is a good candidate to evaluate Markov models for prediction of host availability/unavailability. However, the resolution of the measurement method could have caused the exponential distribution to be a good fit. For example, the `overnet03`, `skype06`, and



(a) Availability (b) Legend (c) Unavailability

Figure 3. The cumulative distribution functions of the availability and unavailability intervals.

Table 4. P-values resulting from KS and AD tests for availability. A gray box denotes p-value above significance level of 0.05.

Trace	Exponential	Weibull	Pareto	Log-Normal	Gamma
lanl05	0.005 0.025	0.416 0.571	0.002 0.010	0.475 0.611	0.345 0.488
g5k06	0.012 0.038	0.472 0.597	0.003 0.018	0.394 0.564	0.409 0.507
microsoft99	0.005 0.084	0.294 0.546	0.000 0.049	0.371 0.611	0.198 0.418
websites02	0.000 0.006	0.079 0.354	0.000 0.027	0.188 0.401	0.055 0.182
pl05	0.000 0.000	0.080 0.245	0.002 0.016	0.168 0.321	0.043 0.131
ldns04	0.009 0.042	0.316 0.510	0.002 0.010	0.357 0.527	0.287 0.472
overnet03	0.045 0.460	0.068 0.532	0.000 0.013	0.160 0.660	0.052 0.481
nd07cpu	0.001 0.011	0.348 0.526	0.002 0.063	0.408 0.596	0.167 0.284
skype06	0.048 0.105	0.373 0.493	0.000 0.002	0.452 0.581	0.257 0.375

Table 5. P-values resulting from KS and AD tests for unavailability. A gray box denotes p-value above significance level of 0.05.

Trace	Exponential	Weibull	Pareto	Log-Normal	Gamma
lanl05	0.000 0.004	0.196 0.346	0.000 0.001	0.481 0.607	0.042 0.095
g5k06	0.000 0.000	0.008 0.073	0.000 0.000	0.037 0.144	0.003 0.022
microsoft99	0.004 0.180	0.048 0.529	0.000 0.376	0.076 0.611	0.052 0.368
websites02	0.000 0.023	0.001 0.150	0.000 0.002	0.005 0.209	0.003 0.090
pl05	0.000 0.000	0.035 0.178	0.000 0.004	0.081 0.274	0.019 0.079
ldns04	0.035 0.112	0.404 0.538	0.000 0.001	0.464 0.607	0.277 0.411
overnet03	0.000 0.040	0.003 0.305	0.000 0.204	0.011 0.389	0.005 0.118
nd07cpu	0.000 0.004	0.028 0.219	0.000 0.031	0.126 0.559	0.003 0.032
skype06	0.071 0.191	0.288 0.478	0.002 0.015	0.182 0.449	0.267 0.408

Table 6. Parameters of distributions for availability (left) and unavailability (right). Statistical terms: mean: μ, std: σ, shape: k, scale: λ.

	Availability intervals				Unavailability intervals			
Trace	**Exp(μ)**	**Wbl(k, λ)**	**LogN(μ, σ)**	**Gam(k, λ)**	**Exp(μ)**	**Wbl(k, λ)**	**LogN(μ, σ)**	**Gam(k, λ)**
lanl05	1779.99	0.48 816.60	5.56 2.39	0.35 5102.71	5.92	0.58 2.18	0.05 1.42	0.38 15.44
g5k06	32.41	0.48 14.37	1.51 2.42	0.34 94.35	7.41	0.35 0.47	-2.00 2.20	0.19 39.92
microsoft99	67.01	0.55 35.30	2.62 1.84	0.41 162.19	16.49	0.60 9.34	1.42 1.54	0.46 35.52
websites02	11.85	0.46 3.68	0.23 2.02	0.31 38.67	1.18	0.65 0.61	-1.12 1.13	0.50 2.37
pl05	159.49	0.33 19.35	1.44 2.86	0.20 788.03	49.61	0.36 5.59	0.40 2.45	0.21 237.65
ldns04	141.06	0.51 79.30	3.25 2.33	0.39 362.43	8.61	0.63 5.62	0.91 1.64	0.51 16.87
overnet03	2.29	0.85 2.04	0.19 0.98	0.91 2.53	12.00	0.44 2.98	0.08 1.80	0.29 41.64
nd07cpu	13.73	0.45 4.16	0.30 2.20	0.30 46.16	4.25	0.51 0.74	-1.02 1.27	0.28 15.07
skype06	16.27	0.64 10.86	1.60 1.57	0.53 30.79	14.31	0.63 9.48	1.40 1.73	0.50 28.53

microsoft99 systems were measured using probes with periods of 20 minutes, 30 minutes, and 1 hour, respectively. As a consequence, there are no (un)availability intervals taking less than this length, and in the CDFs shown in Figure 3, there are spikes at those period lengths. Further investigation of the usability of exponential distributions for failure characteristics in distributed systems is needed (and enabled by the FTA); the implications of the findings can affect a large number of theoretical and practical studies.

Our results reveal that for the availability/unavailability distributions heavy-tailed distributions do not give a good fit—the p-values for Pareto are very low. The only exceptions are the distributions of unavailability of overnet03 and microsoft99, which are close to being heavy-tailed. It is worth noting that the AD test is more sensitive to the tail than the KS test. This explains the difference between p-values of the two GOF tests, especially when we have a heavy-tailed data set.

For all data sets, the Gamma distribution is a good candidate for the failure model as the p-values are relatively high. This distribution function is very flexible and can also be adopted for analytical work based on Markov models [66]. Additionally, the results of the GOF tests show that the best fit for all data sets are either the Weibull or the Log-Normal distribution. As expected from our statistical analysis, the failure model tends to be a long-tailed distribution. However, several data sets, such as g5k06 and pl05, show imperfect fits for the unavailability distribution. A possible explanation is the relation of the model with the system architecture. For example, the g5k06 platform has 15 clusters in 9 geographically distributed sites, and each cluster could have its own separate model, as proposed by Iosup et al. [5]. Moreover, as mentioned before, we need distributions with more degrees of freedom such as the hyper-exponential to model the unavailability distributions.

We conclude our study towards a failure model for distributed systems with a summary of parameter values obtained for various distributions, when fit to empirical data. Table 6 summarizes these values for the availability and unavailability intervals of all data sets under study. For the availability distributions, we analyze the *hazard rate*, i.e., the probability of the next failure with respect to time from the last failure. For the data sets for which the Weibull or Gamma distributions are a good fit, the hazard rate is decreasing. Recall that for such distributions if the shape parameter is less than one, i.e., $k < 1$, then we have a decreasing hazard rate. That means that if the systems do not have any failure for a long time (longer availability duration) the probability of a failure occurring in the near future decreases. In other words, a decreasing hazard rate can be interpreted as more stability of resources over time. The only hazard rate that is alarming is with overnet03, where the shape parameter is close to one.

4.3. Qualitative Comparison of Failure Characteristics in Distributed Systems

In this section we create a qualitative comparison of failure characteristics in distributed systems. We approach this comparison through a conceptual framework in which each data set in the FTA can be characterized through various qualitative parameters, then compared qualitatively with other data sets. The main benefit of this comparison is making the selection of traces easier for the non-expert user. The qualitative comparison is summarized in Table 7; the meaning of the values is explained in the following.

Our comparison framework uses four parameters, two based on data collected at node-level, and two based on data collected at system-level. The volatility (V) is dependent on the failure rate of each node in the system. The availability (A) is the percentage of time that a node is working properly. The measurement duration (M) and scale (S) are the duration of measurement and scale of the entire system, respectively. For each item we assign three different levels as described in Table 8. For the

Table 7. Qualitative comparison of nine data sets in the FTA. H:High, M:Medium, L:Low. For V, A, M, and S, see Table 8 and associated text.

Trace	V	A	M	S	Failure model
lanl05	L	H	H	H	Long-tailed/Long-tailed
g5k06	M	M	H	M	Long-tailed/Long-tailed
microsoft99	M	M	L	H	Short-tailed/Heavy-tailed
websites02	H	H	M	L	Long-tailed/Long-tailed
pl05	L	M	H	L	Long-tailed/Long-tailed
ldns04	L	H	L	H	Long-tailed/Short-tailed
overnet03	H	L	L	H	Short-tailed/Heavy-tailed
nd07cpu	H	M	M	L	Heavy-tailed/Long-tailed
skype06	H	L	L	H	Short-tailed/Short-tailed

Table 8. Parameters in the qualitative comparison (see text).

Parameter observation	Parameter name	Parameter Level Low	High	Parameter unit
Node-level	V: Volatility	$V < 50$	$V > 100$	hour
Node-level	A: Availability	$A < 60$	$A > 90$	%
System-level	M: Measurement	$M < 6$	$M > 12$	month
System-level	S: Scale	$S < 1$	$S > 2$	10^3 nodes

qualitative comparison, we also look at the types of models that fit well the availability and unavailability. Specifically, for each trace we use the tail behavior of its availability and unavailability distributions.

For the failure model, we have observed in Section 4.2 that all best-fits are long-tailed distributions. However, for the qualitative comparison we have applied another classification, one which is based on the p-values of the KS and AD tests with a significance level of 0.05. Specifically, if a data set has acceptable p-values for Pareto or Exponential distribution, the failure model would be heavy-tailed or short-tailed, respectively. Otherwise, the failure model could be classified as long-tailed (for more details about tail behavior, see [66]).

5. Differences of Interpretation

To emphasize the critical need for public data and analysis methods, we give three examples of where differences of trace interpretation result in differences in the derived failure models. In particular, we show that differences of interpretation can change significantly the distribution of failures in terms of passing statistical goodness-of-fit tests and the fitted distributions. Overall, we show significant differences for both empirical and fitted distribution. This emphasizes the need for public data sets and for a general framework for data interpretation, expressed in methods and tools.

We choose three data sets, namely `lanl05`, `g5k06`, and `nd07cpu`, where the time of failures can be interpreted differently. On close examination of the `lanl05` trace, we found that there are overlapping unavailability intervals. This overlapping of intervals was especially evident in System 16 of this trace, which is a cluster of 16 NUMA-based nodes, each of which has 128 processors and 4 NICs.

In some cases, one failure interval completely subsumed another. In other cases, the start time of a failure interval A was greater than the start time of another interval B but less than the stop time of interval B. Moreover, the stop time of A was greater than the stop time B. We believe these intervals might be the result of human error, as the data was manually recorded.

The authors that first described this data set [3] did not detail the cause of these intervals nor how or why these intervals were interpreted in a certain way. Comparing our statistics of the failures with those previously reported by Schroeder and Gibson [3], we "reversed engineered" the interpretation, and found that the authors used the *union* of failures intervals having ambiguity. For comparison, we interpret the failure intervals in System 16 differently and optimistically using their *intersection*, calling the resulting post-processed data set **lanl0516B**.

We also found different possible interpretations for the `g5k06` trace. In the raw trace, the states of nodes are given as `available`, `unavailable`, `suspected`, or `dead`. `Suspected` is a state (assigned mostly automatically) in which a node does not behave well according to OAR, the Grid'5000 node manager. The "bad" behavior is detected through many tools, such as the node monitor *finaud*, the jobs monitor *sarko*, and the internal OAR state manager *NodeChangeState*. Pessimistic trace processing would interpret the `suspected` state as a failure, and assume unavailability. An optimistic trace processing would interpret the `suspected` state as a fault but not a failure, and assume availability. The former interpretation is used in the `g5k06` trace described in previous sections and by Iosup et al. [5]. We denote the latter interpretation as **g5k06B**.

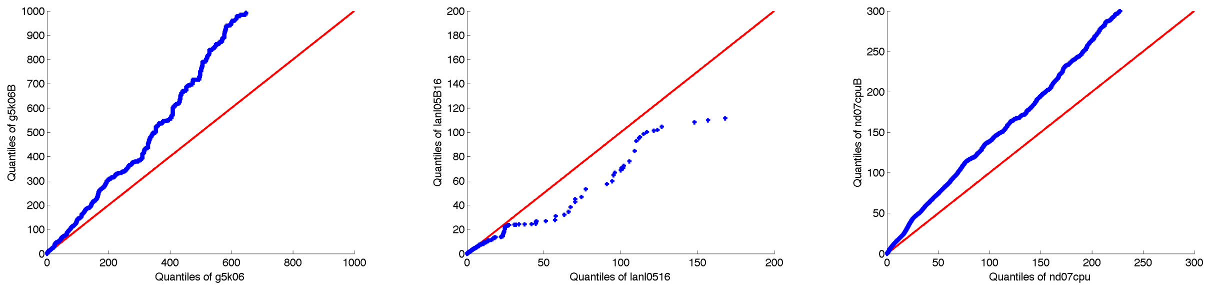
The `nd07cpu` trace is the third data set for which we draw attention about various possible interpretations. The trace is comprised of host idle times and CPU loads. Defined by Rood and Lewis [6], CPUs are available when the host is idle without any user for more than 15 minutes and the CPU load (which could be independent of the user) is less than 50%. We relaxed this condition to lengthen the CPU availability time by including the time when a user is present (which, in turn, would cause zero idle time) and the CPU load is less than 10%. This is a reasonable definition of CPU availability, as a guest job could still run on the host without interfering significantly with local jobs. The data set with this latter interpretation is referred to as **nd07cpuB**.

In the following, we present the analysis of different failure interpretations for the aforementioned data sets. First, we compare the empirical distributions graphically. Second, we fit several distributions to each of the data sets, and compare the fitted distributions for each pair of data sets. We compare the fitted distributions statistically with p-values, and then graphically with qq-plots.

5.1. Differences of Empirical Distributions

We now investigate the statistical properties of availability and unavailability intervals, under different assumptions. Overall, we find that the impact of different interpretations is significant. Specifically, we find that a different interpretation can lead to increase, decrease, or no change in the characteristics of either availability and unavailability intervals. Moreover, we find that the characteristics of availability and unavailability intervals can change independently of each other, e.g., one can increase while the other can decrease.

Figure 4 shows the quantiles of the empirical distributions for each pair of data sets. If the two data sets have the same distribution, their qq-plot will match the line $y = x$, which is plotted in solid red as a reference. We only show representative results: qq-plots for `g5k06`'s availability, `lanl0516`'s



(a) g5k06, availability (b) lanl0516, unavailability (c) nd07cpu, availability

Figure 4. Quantile-Quantile plots of empirical data for ambiguous data sets.

unavailability, and `nd07cpu`'s availability.

For `g5k06` (Figure 4(a)), `g5k06B` has longer availability intervals; it also has shorter unavailability intervals (not depicted here). This is due to the optimistic interpretation of the `suspected` state. The deviation is greatest at the quantile at 1000 hours of `g5k06B`, which corresponds to the the quantile of 600 hours of `g5k06`. Also, the mean availability in `g5k06B` was increased by a factor of 1.50 (!) because of the difference of interpretation. The mean unavailability in `g5k06B` was decreased by a factor of 1.13 due to the decrease in the number of failures.

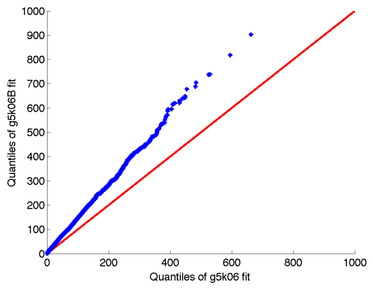
For `lanl0516`, we have observed little differences of interpretation for availability. This is due to System 16 being highly available over a long period of time, which means that changes in unavailability periods may not also be reflected in availability periods. However, there are clear differences in the distribution of unavailability, as shown in Figure 4(b). Specifically, the unavailability intervals for `lanl0516B` are statistically much shorter than for `lanl0516`.

For `nd07cpu`, we find that `nd07cpuB` has statistically longer availability and unavailability intervals than `nd07cpu`. In particular, the mean lengths of availability and unavailability intervals have increased by a factor of 1.47 and 1.35, respectively. While the total amount of unavailability decreased in `nd07cpuB`, small unavailability intervals were interpreted as availability intervals, after the optimistic processing of the traces. Thus, both the mean lengths of availability and unavailability were increased.

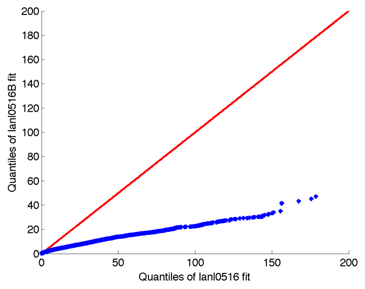
5.2. Differences of Fitted Distributions

We now show how the differences of interpretation affect the statistical goodness-of-fit tests of fitted distribution and their fitted parameters. Overall, similarly to Section 5.1 we find that the impact of different interpretations is significant.

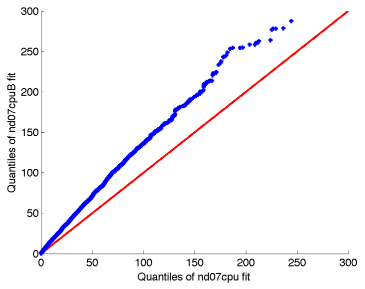
Typically, a significance value of 0.05 or 0.10 is used as a threshold for p-values to determine whether to reject the NULL hypothesis that the fitted distribution represents the empirical. We found several cases where the p-values for different interpretations would result in conflicting conclusions, i.e., rejection for one interpretation and and failure of rejection for another. For example, the AD-test for the Weibull distribution fitted to `g5k06`'s unavailability intervals resulted in a p-value of 0.07, whereas the p-value corresponding to `g5k06B` was 0.035. Similarly, for the AD test for the Log-Normal distribution the p-value is 0.148 for `g5k06` versus 0.057 for `g5k06B`. Thus, for a significance level of 0.05, we find that the Weibull distribution would not be rejected as a good fit for `g5k06`'s unavailability distribution, but would be rejected `g5k06B`'s unavailability distribution. For a threshold of 0.10, the Log-Normal



(a) g5k06, availability



(b) lanl0516, unavailability



(c) nd07cpu, availability

Figure 5. Quantile-Quantile plots of fitted distributions for ambiguous data sets.

Table 9. Parameters of distributions for availability (left) and unavailability (right) for ambiguous data sets (mean: μ, std: σ, shape: k, scale: λ). A grey box indicates a significant difference of parameters between data sets.

System	Exp(μ)	Wbl(k, λ)	LogN(μ, σ)	Gam(k, λ)	Exp(μ)	Wbl(k, λ)	LogN(μ, σ)	Gam(k, λ)
g5k06	32.41	0.48 14.37	1.51 2.42	0.34 94.35	7.41	0.35 0.47	-2.00 2.20	0.19 39.92
g5k06B	48.61	0.52 22.66	2.08 2.21	0.37 131.78	6.54	0.35 0.31	-2.36 2.07	0.18 37.00
lanl05	1779.99	0.48 816.60	5.56 2.39	0.35 5102.71	5.92	0.58 2.18	0.05 1.42	0.38 15.44
lanl05B	1774.21	0.48 812.98	5.55 2.39	0.35 5087.60	5.06	0.59 2.12	0.03 1.40	0.41 12.28
nd07cpu	13.73	0.45 4.16	0.30 2.20	0.30 46.16	4.25	0.51 0.74	-1.02 1.27	0.28 15.07
nd07cpuB	20.12	0.48 7.21	0.91 2.07	0.33 61.74	5.75	0.49 0.83	-0.91 1.21	0.26 21.72

distribution would not be rejected for `g5k06`, but would be rejected for `g6k06B`'s unavailability distribution.

We found similar cases for `lanl0516` and `lanl0516B`, and for `nd07cpu` and `nd07cpuB`. For `lanl0516`, the Gamma distribution is rejected for `lanl0516`'s unavailability intervals but not rejected for `lanl0516B` according to the p-values resulting from the KS test (0.046 versus 0.056). For `nd07cpu`, the Log-Normal distribution is rejected for `nd07cpuB`'s unavailability intervals, but not rejected for `nd07cpu` according to the p-value for the KS test (0.14 versus 0.01).

In addition to quantitative contradictions, we also depict contradictions graphically, in Figure 5. We plot the quantiles for the fitted Gamma distributions of pairs of data sets. We choose the Gamma distribution as it is analytically easy to use and has a relatively high p-value.

Figure 5 depicts qq-plots of the Gamma distributions for `g5k06`'s availability, `lanl0516`'s unavailability, and `nd07cpu` availability distributions. We observe from the qq-plots that the distributions fitted to different interpretations of the same data set are significantly different. For example, for `lanl0516`, we see that the quantile of 40 hours for `lanl0516B` corresponds to the quantile of 180 hours for `lanl516`.

Furthermore, the impact on the distribution parameters is significant as shown in Table 9. Significant differences in parameters are highlighted in grey. For example, the mean of the exponential distribution for `g5k06B`'s availability is a factor of 1.50 times greater than `g5k06`. Moreover, different interpretations can significantly affect the scale parameter of the Gamma. For example, the scale parameter of the gamma distribution for `g5k06B`'s availability is factor of 1.39 times greater than `g5k06`.

6. Performance Evaluation

In this section, we evaluate the effect of resource failures on the job scheduling for two different case studies. To do this, we consider a set of parallel jobs to be scheduled on a parallel system in the presence of resource failures. We also use the results of trace interpretation explained in Section 5 as the failure traces. The FTA simulator based on GridSim [34] is used to simulate the considered case studies.

6.1. Performance Metrics

The performance metrics that we consider in all simulation scenarios are the Average Weighted Response Time (AWRT) [67] and the Bounded Slowdown (BS) [68]. The AWRT for N given jobs is defined by:

$$AWRT = \frac{\sum_{j=1}^{N} d_j \cdot v_j \cdot (c_j - s_j)}{\sum_{j=1}^{N} d_j \cdot v_j}, \tag{1}$$

where d_j is the execution time of the job, v_j is the number of nodes used by job j, c_j is the time of completion of the job, and s_j is its submission time. The resource consumption $d_j \cdot v_j$ of a job j is used as its weight in the denominator. AWRT measures the average time that users must wait to have their jobs completed weighted by the weights of the jobs. The BS for a set of N jobs is defined as follows:

$$BS = \frac{1}{N} \sum_{j=1}^{N} \frac{W_j + \max(d_j, b)}{\max(d_j, b)}, \tag{2}$$

where W_j is the waiting time of job j and b is a lower bound on the runtimes of jobs that is used to eliminate the effect of very short jobs [68]. We set b to 10 seconds.

6.2. Experimental Setup

The workload model for evaluation scenarios is obtained from the Grid Workload Archive [13]. We use the parallel job model of the DAS-2 multi-cluster Grid [69]. Based on the workload characterization, the inter-arrival time, the job size, and the job duration follow Weibull, two-stage Loguniform, and Lognormal distributions, respectively. These distributions with their parameters are listed in Table 10. It should be noted that the number of nodes in the job can be scaled to the system size (e.g., M nodes) by setting $h = log_2 M$. Based on the workload model, P_{one} and P_{pow2} are the probabilities of occurrence in the workload of jobs that run on one node and of jobs that run on a number of nodes that is a power of two, respectively. In order to generate different synthetic workloads, we modify the second parameter of the Weibull distribution (the *shape* parameter β) as shown in Table 10 to change the inter-arrival time of the jobs.

The failure traces used for the experiments are `g5k06` and `lanl05`. We use the failure trace of a cluster in Grid'5000 with 64 nodes (i.e., Cluster number 2) as well as the system number 16 from the LANL system with 16 nodes and 128 processors. The characteristics of the failure traces are listed in Table 11. For each failure trace, we use the same interpretations as explained in Section 5. As one can

Table 10. Input parameters for the workload model.

Input Parameters	Distribution/Value
Inter-arrival time	Weibull ($\alpha = 23.375, 0.2 \leq \beta \leq 0.3$)
No. of nodes	Loguniform ($l = 0.8, m, h, q = 0.9$)
Job duration	Lognormal ($\mu = 4.5, \sigma = 2.0$)
P_{one}	0.024
P_{pow2}	0.788

Table 11. Characteristics of the failure traces used in the experiments.

Traces	No. of events	Avg. events/node	Avg. availability (hrs)	Avg. unavailability (hrs)
g5k06C2	50924	795	22.256	10.223
g5k06C2B	33310	520	37.699	11.948
lanl0516	5092	318	228.806	9.328
lanl0516B	5208	325	229.846	3.006

see, the average availability of `g5k06C2B` is significantly increased where the average unavailability slightly increased. In contrast, the average unavailability of `lanl0516B` is decreased where the average availability is marginal increased.

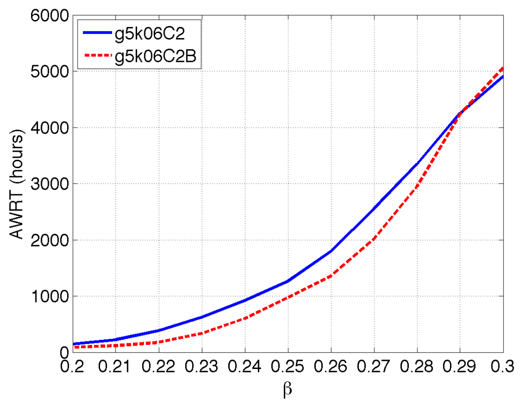
We configure the simulator so that it simulates systems similar to the ones on which the traces were collected. We use *selective* backfilling [70] as the job scheduler, which grants a reservation to a job when its expected slowdown exceeds a threshold, i.e., when the job has waited long enough in the queue. The expected slowdown of a given job, sometimes also called its *eXpansion Factor* (XF), is defined as $XF = (W_i + d_i)/d_i$, where W_i and d_i are the waiting time and the execution time of job i, respectively. We use the Selective-Differential-Adaptive scheme proposed in [70], which sets the threshold for XF to be the average slowdown of previously completed jobs. It has been shown that selective backfilling outperforms other types of backfilling algorithms [70], which is the reason why we use it in our experimentations.

After submitting a job to the scheduler, each job runs on nodes that are available. In case of resource (node) failure during the execution of a job, we assume *perfect* checkpointing so that the job is started from where it left off when the node becomes available again. Checkpointing issues are beyond the scope of this paper, and we refer interested readers to [71] to see how checkpoint overheads and periods can be computed bases on the associated failure model.

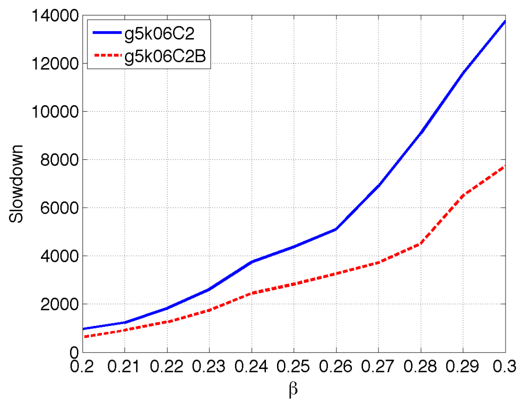
For each simulation experiment, we gathered statistics for a two-month period of the DAS-2 workloads. The first week of workloads during the *warm-up* phase were ignored to avoid bias before the system reach steady state. For the experiments, each data point is the average of 50 simulation rounds to have a better confidence on the performance metrics.

6.3. Results and Discussion

In Figure 6 we show the simulation results in terms of AWRT and BS versus the arrival rate for two traces of `g5k06`. As can be seen, the response time and slowdown for the `g5k06C2B` trace is much lower than for the `g5k06C2`. The main reason for this is the higher average availability of `g5k06C2B` as illustrated in Table 11. The same results for the `lanl05` traces are plotted in Figure 7. While the AWRT of the `lanl0516B` trace is a bit higher than that of `lanl0516`, the slowdown shows no obvious difference for the two traces. One possible reason is the high availability of both `lanl05` traces (see Table 11).

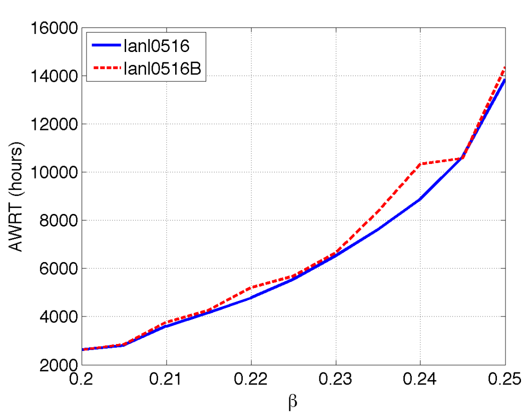


(a) AWRT

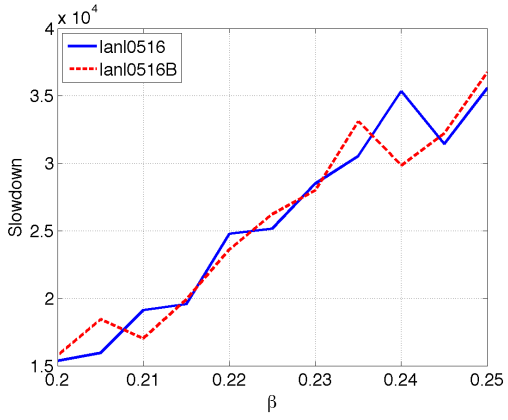


(b) BS

Figure 6. The AWRT and the BS versus the arrival rate for two traces of g5k06.



(a) AWRT

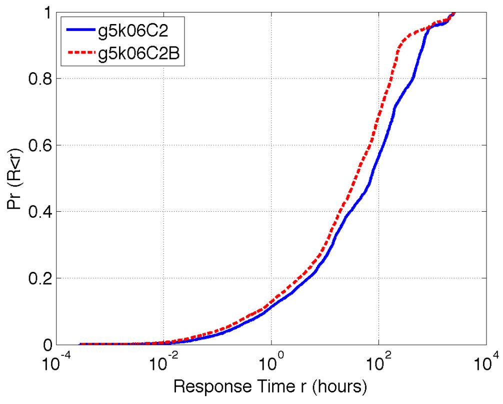


(b) BS

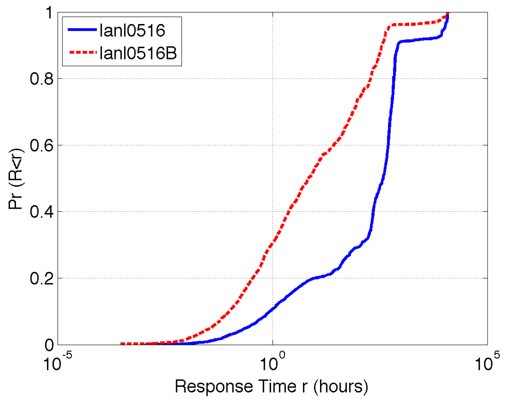
Figure 7. The AWRT and the BS versus the arrival rate for two traces of lanl05.

To be more precise in terms of comparing two interpretations of a failure trace, we also consider the Cumulative Distribution Function (CDF) of the job response time for both the `g5k06` and `lanl05` traces under a moderate workload. As one can see in Figure 8(a), the response time for `g5k06C2` is higher than that for `g5k06C2B`, similar to Figure 6. However, Figure 8(b) shows that in contrast to what we observe in Figure 7, the job response time with `lanl0516B` is lower than with `lanl0516`. This reveals that using average values can not always provide an accurate reflection of system performance in the presence of resource failures. In our case, since the difference of the two `g5k06` traces in terms of availability is really considerable, the AWRT and BS metrics can show the difference of the system performance. However, both `lanl05` traces have very close availability and the only difference is the lower unavailability in `lanl0516B`. In this case, the CDF is much more informative than the average metrics.

The last but not least, making both trace data and analytical methods publicly available is critical to apply the failure traces in the research and development of reliability algorithms for parallel and distributed systems.



(a) g5k06C2 ($\beta = 0.25$)



(b) lanl0516 ($\beta = 0.225$)

Figure 8. The cumulative distribution functions of the job response time for two traces under a moderate load.

7. Related Work

In this section, we compare our work with related work: other public-access archives and previous works on statistical modeling.

Differences between the FTA and other public-access archives: The FTA extends previous work in public-access archives in three main ways. First, the FTA defines a comprehensive unified format that facilitates use and comparison of the traces. In particular, the FTA data format can accommodate various types of components and events. None of the archives we survey in this section can accommodate information from all the distributed system types investigated in this work. Second, the FTA already hosts a much broader and deeper collection than all the public-access archives we survey here. The FTA contains over 20 data sets, covers 8 classes of distributed systems and representing over 10 application domains, and spans with its failure traces over 20 operational years. Moreover, the FTA also shares raw (!) data and various scripts for data parsing. Third, the FTA provides a public toolbox for failure trace analysis. None of the other archives we survey in this section provides such a toolbox; although such a toolbox is needed to reduce the interpretation differences when using different data acquisition processes (see also Sections 5 and 6), none of the other archives provides a comprehensive toolbox.

Open-access archives for failure data: The Computer Failure Data Repository [15] provides failure traces for supercomputers and clusters used for one application domain, HPC. However, no standard format is defined, and only raw data is shared. The Repository of Availability Traces [14] contains traces for 5 distributed systems in a common format, and scripts used for parsing the raw data. While a standard format is defined, we believe this format excludes critical information for capturing a range of failure types and systems. For example, the format does not contain the failure cause, the creator, and the component type. The Desktop Grid Trace Archive [4] is focused specifically on one type of distributed systems, desktop grids; moreover, a generic failure format is not provided. The Grid Observatory [72] provides numerous data sets, some of which could provide failure information. However, the repository is currently limited to a single system (i.e., EGEE) and only provides raw data. The Observatory does not provide a common data format, or scripts for parsing and analysis.

Other open-access archives for distributed systems, but without failure data: There are several trace archives that provides workload data set of parallel and distributed systems. The Parallel Workloads Archive [12] includes many workload traces of supercomputers and parallel machines. The Grid Workloads Archive [13] provides workload traces of clusters and grids. The P2P Trace Archive [73] shares many workload and operational traces collected from peer-to-peer systems. The Game Trace Archive [74] publishes operational and other traces representative to distributed systems supporting online gaming, typically massively multiplayer. Although all these archives have their own trace format, they do not focus on job, service, or resource failures.

Differences between this study and previous work on statistical modeling of failures in distributed systems: Overall, our work presents the first uniform and comprehensive statistical analysis and comparison of the failure characteristics for different types of distributed systems. Most previous studies, in particular [2, 36, 37, 38, 39], do not focus on modelling issues. A few other studies [3, 5] have also conducted modelling of various failure characteristics. However, this body of previous modelling work focuses on a particular system type or even on a single data set, and so the generality of the model for distributed systems was not yet confirmed.

In the study of Bakkaloglu et al. [35], the entity being modeled is different than ours. Bakkaloglu et al. model the *number* of machines available at some time point, considering correlated failures, in the context of a distributed storage system. In contrast, our study focuses on the the continuous *durations* of availability and failures. This latter characteristic is essential for stochastic scheduling algorithms that conduct task assignment based on the probability of task completion.

8. Conclusion

Despite the importance and impact of failures in (large-scale) distributed computing environments, few traces collected from real environments that contain information on failures are publicly available. To address this situation, which restricts the applicability of failure models and the development of failure-aware systems, our contribution in this work is threefold:

1. We have created the Failure Trace Archive for facilitating the comparative analysis of failures in distributed and parallel systems. We defined a standard trace format, and showed its suitability by converting traces of 20 diverse distributed systems into this format. Given traces in this format, we implemented a toolbox and a simulator that facilitates the comparison of failure statistics, models, and algorithms. Ultimately, we envision that scientists would use the toolbox as a repository of modeling and predictive methods.

2. Using the toolbox, we gave a uniform and global statistical analysis of failure in nine distributed systems. One key finding was that the Weibull, the Lognormal, and the Gamma distributions are often the best candidates for availability and unavailability distributions. Moreover, the hazard rate with respect to the Weibull distribution was decreasing in all systems. In some cases, the measurement method (in particular the resolution of probing) seemed to cause bias in the distribution of availability, and we identified these data sets with potential bias.
3. Finally, we have shown how differences of interpretation of trace data sets can result in significantly different failure models and statistics. Moreover, we observed that differences of interpretation have a major impact on the job scheduling in the presence of resource failures. This shows that it is critical to make both trace data and analytical methods publicly available.

As future research, we intend to discover the relationship between lower-level failures (for example, of nodes or components) and higher-level failures (for example, of jobs) in large-scale distributed systems.

9. Availability of FTA Data and Scripts

The Failure Trace Archive, including technical documentation on the data format, the toolbox. and the trace data sets are available online at: `http://fta.scem.uws.edu.au`.

Acknowledgements

We gratefully acknowledge the generous contributors to the Failure Trace Archive that made their trace data sets publicly available. We thank the INRIA ALEAE project directed by Emmanuel Jeannot, and the ANR SPADES (08-ANR-SEGI-025) project for supporting this collaborative work. We thank Eric Heien for creating the initial hierarchical version of the Failure Trace Archive format. We thank Cecile Germain for useful discussions with respect to resource versus job versus user-level failures. This work was also partially supported by the STW/NWO Veni grant @larGe (11881).

References

[1] Failure Rates at Google, `http://perspectives.mvdirona.com/2008/06/11/JeffDeanOnGoogleInfrastructure.aspx`.

[2] W. Bolosky, J. Douceur, D. Ely, M. Theimer, Feasibility of a Serverless Distributed file System Deployed on an Existing Set of Desktop PCs, in: Proceedings of SIGMETRICS, 2000.

[3] B. Schroeder, G. A. Gibson, A large scale study of failures in high-performance-computing systems, in: International Symposium on Dependable Systems and Networks (DSN), 2006.

[4] D. Kondo, G. Fedak, F. Cappello, A. A. Chien, H. Casanova, Characterizing resource availability in enterprise desktop grids, Future Generation Comp. Syst. 23 (7) (2007) 888–903.

[5] A. Iosup, M. Jan, O. Sonmez, D. H. Epema, On the dynamic resource availability in grids, in: 8th IEEE/ACM International Conference on Grid Computing, 2007.

[6] B. Rood, M. J. Lewis, Multi-state grid resource availability characterization, in: 8th Grid Computing Conference, 2007.

[7] A. Andrzejak, D. Kondo, D. P. Anderson, Ensuring collective availability in volatile resource pools via forecasting, in: DSOM, 2008, pp. 149–161.

[8] A. Andrzejak, P. Domingues, L. Silva, Predicting Machine Availabilities in Desktop Pools, in: IEEE/IFIP Network Operations and Management Symposium, 2006, pp. 225–234.

[9] P. Cashmore, Twitter down: Twitter doesn't know why, Mashable report. [Online] Available: `http://mashable.com/2009/08/06/twitter-down/` (Aug 2009).

[10] P. Cashmore, Facebook down. Twitter down. Social media meltdown., Mashable report. [Online] Available: `http://mashable.com/2009/08/06/facebook-down-3/` (Aug 2009).

[11] Various sources, Microsoft windows live hotmail, dec 30, 2010 to jan 2, 2011, More information: `http://windowsteamblog.com/windows_live/b/windowslive/archive/2011/01/03/hotmail-email-access-issue-now-resolved.aspx`, `http://www.infoworld.com/t/software-service/hotmail-fail-microsoft-lays-egg-in-the-cloud-616`. (Dec 2010).

[12] D. Feitelson, Parallel Workloads Archive, `http://www.cs.huji.ac.il/labs/parallel/workload/`.

[13] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, D. H. J. Epema, The grid workloads archive, Future Generation Comp. Syst. 24 (7) (2008) 672–686.

[14] B. Godfrey, Repository of Availability Traces, `http://www.eecs.berkeley.edu/~pbg/availability`.

[15] B. Schroeder, G. A. Gibson, The computer failure data repository., in: Workshop on Reliability Analysis of System Failure Data (RAF'07), 2007.

[16] D. Kondo, B. Javadi, A. Iosup, D. H. J. Epema, The failure trace archive: Enabling comparative analysis of failures in diverse distributed systems, in: CCGRID, 2010, pp. 398–407.

[17] A. Avizienis, J.-C. Laprie, B. Randell, C. E. Landwehr, Basic concepts and taxonomy of dependable and secure computing, IEEE Trans. Dependable Sec. Comput. 1 (1) (2004) 11–33.

[18] F. Salfner, M. Lenk, M. Malek, A survey of online failure prediction methods, ACM Comput. Surv. 42 (3).

[19] I. T. Foster et al., The grid2003 production grid: Principles and practice, in: HPDC, 2004, pp. 236–245.

[20] C. Dumitrescu, I. Raicu, I. T. Foster, Experiences in running workloads over grid3, in: GCC, 2005, pp. 274–286.

[21] A. Iosup, M. Jan, O. O. Sonmez, D. H. J. Epema, On the dynamic resource availability in grids, in: GRID, 2007, pp. 26–33.

[22] A. Iosup, D. H. J. Epema, Build-and-test workloads for grid middleware: Problem, analysis, and applications, in: CCGRID, 2007, pp. 205–213.

[23] H. Casanova, Benefits and drawbacks of redundant batch requests, J. Grid Comput. 5 (2) (2007) 235–250.

[24] B. Cohen, Incentives build robustness in bittorrent, in: Proceedings of The 2nd International Workshop on Peer-to-Peer Systems (IPTPS), 2003.

[25] B. Zhang, A. Iosup, J. A. Pouwelse, D. H. J. Epema, Identifying, analyzing, and modeling flashcrowds in bittorrent, in: Peer-to-Peer Computing, 2011, pp. 240–249.

[26] BBC, Blackouts cause n america chaos, BBC News. [Online] Available: `http://news.bbc.co.uk/2/hi/americas/3152451.stm` (Aug 2003).

[27] BBC, Huge blackout cripples Italy, BBC News. [Online] Available: `http://news.bbc.co.uk/2/hi/3146136.stm` (Sep 2003).

[28] M. Masnik, Akamai knocked out, major websites offline, TechDirt.com article. [Online] Available: `http://techdirt.com/articles/20040524/0923243.shtml` (May 2004).

[29] S. Nath, H. Yu, P. B. Gibbons, S. Seshan, Subtleties in tolerating correlated failures in wide-area storage systems, in: NSDI, 2006.

[30] Various sources, Amazon EC2 and RDS fail, april 21, 2011, from 12am to 12pm (12 hours), More information: `http://aws.amazon.com/message/65648/`, `http://www.zdnet.com/blog/projectfailures/cio-analysis-examining-amazons-cloud-failure/13152`, `http://bits.blogs.nytimes.com/2011/04/21/amazon-cloud-failure-takes-down-web-sites/`, and `http://www.economist.com/node/18620774/print`. (Dec 2010).

[31] Reuters, Knight capital was hardly the first: Timeline of market glitches, Reuters report. [Online] Available: `http://www.foxbusiness.com/industries/2012/08/02/knight-capital-was-hardly-first-timeline-market-glitches/` (Aug 2012).

[32] CCP, The battle of Asakai and Poinen must burn, by the numbers, CCP Developers report. [Online] Available: `http://community.eveonline.com/devblog.asp?a=blog&nbid=74226` (Jan 2013).

[33] J. Romero, Lack of rain a leading cause of Indian Grid collapse, IEEE Spectrum. [Online] Available: `http://spectrum.ieee.org/energywise/energy/the-smarter-grid/disappointing-monsoon-season-wreaks-havoc-with-indias-grid/` (Jul 2012).

[34] A. Sulistio, U. Cibej, S. Venugopal, B. Robic, R. Buyya, A toolkit for modelling and simulating data grids: an extension to GridSim, Concurrency and Computation: Practice and Experience 20 (13) (2008) 1591–1609.

[35] M. Bakkaloglu, J. J. Wylie, C. Wang, G. R. Ganger, On correlated failures in survivable storage systems, Technical Report MU-CS-02-129, Carnegie Mellon University (May 2002).

[36] J. Stribling, Planetlab all paris ping, `http://pdos.csail.mit.edu/~strib/pl_app/`. URL `http://infospect.planet-lab.org/pings`

[37] J. Pang, J. Hendricks, A. Akella, B. Maggs, R. D. Prisco, S. Seshan, Availability,usage, and deployment characteristics of the domain name system, in: Internet Measurement Conference (IMC), 2004.

[38] R. Bhagwan, S. Savage, G. Voelker, Understanding Availability, in: Proceedings of IPTPS'03, 2003.

[39] S. Guha, N. Daswani, , R. Jain, An experimental study of the skype peer-to-peer voip system, in: Proceedings of The 5th International Workshop on Peer-to-Peer Systems (IPTPS), 2006.

[40] D. Anderson, BOINC: A system for public-resource computing and storage, in: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, Pittsburgh, USA, 2004.

[41] B. Javadi, D. Kondo, J. Vincent, D. Anderson, Mining for statistical availability models in large-scale distributed systems: An empirical study of SETI@home, in: 17th IEEE/ACM International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2009.

[42] D. Brown, G. Smith, Mpp2 syslog data (2006-2008), Technical Report PNNL-SA-61371, Pacific Northwest National Laboratory (2008).

[43] R. Arpaci, A. Dusseau, A. Vahdat, L. Liu, T. Anderson, D. Patterson, The Interaction of Parallel and Sequential Workloads on a Network of Workstations, in: Proceedings of SIGMETRICS'95, 1995, pp. 267–278.

[44] D. Kondo, M. Taufer, C. L. B. III, H. Casanova, A. A. Chien, Characterizing and evaluating desktop grids: An empirical study, in: IPDPS, 2004.

[45] G. Fedak, C. Germain, V. N'eri, F. Cappello, XtremWeb: A Generic Global Computing System, in: Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CC-GRID'01), 2001.

[46] N. Yigitbasi, M. Gallet, D. Kondo, A. Iosup, D. Epema, Analysis and modeling of time-correlated failures in large-scale distributed systems, in: 8th IEEE/ACM International Conference on Grid Computing, 2010.

[47] D. L. Iglesias, D. Kondo, J. M. Marquès, Long-term availability prediction for groups of volunteer resources, J. Parallel Distrib. Comput. 72 (2) (2012) 281–296.

[48] L.-C. Canon, E. Jeannot, J. B. Weissman, A scheduling and certification algorithm for defeating collusion in desktop grids, in: ICDCS, 2011, pp. 343–352.

[49] Z. Huang, E. Biersack, Y. Peng, Reducing repair traffic in p2p backup systems: Exact regenerating codes on hierarchical codes, TOS 7 (3) (2011) 10.

[50] M. Bougeret, H. Casanova, M. Rabie, Y. Robert, F. Vivien, Checkpointing strategies for parallel jobs, in: SC, 2011, p. 33.

[51] B. Donassolo, A. Legrand, C. Geyer, Non-cooperative scheduling considered harmful in collaborative volunteer computing environments, in: CCGRID, 2011, pp. 144–153.

[52] B. Javadi, J. Abawajy, R. Buyya, Failure-aware resource provisioning for hybrid cloud infrastructure, Journal of Parallel and Distributed Computing 72 (10) (2012) 1318 – 1331.

[53] B. Donassolo, H. Casanova, A. Legrand, P. Velho, Fast and scalable simulation of volunteer computing systems using simgrid, in: HPDC, 2010, pp. 605–612.

[54] S. Ostermann, K. Plankensteiner, R. Prodan, Using a new event-based simulation framework for investigating resource provisioning in clouds, Scientific Programming 19 (2-3) (2011) 161–178.

[55] M. Gallet, N. Yigitbasi, B. Javadi, D. Kondo, A. Iosup, D. H. J. Epema, A model for space-correlated failures in large-scale distributed systems, in: Euro-Par (1), 2010, pp. 88–100.

[56] B. Javadi, D. Kondo, J.-M. Vincent, D. P. Anderson, Discovering statistical models of availability in large distributed systems: An empirical study of SETI@home, IEEE Trans. Parallel Distrib. Syst. 22 (11) (2011) 1896–1903.

[57] ACM/IEEE-CS Joint Task Force, Computer Science Curricula 2013 (CS2013), ACM and IEEE Computer Society joint development of a Computing Curricula volume on Computer Science. [Online] Available: `http://ai.stanford.edu/users/sahami/CS2013/`.

[58] A. Kimball, S. Michels-Slettvet, C. Bisciglia, Cluster computing for web-scale data processing, in: SIGCSE Technical Symposium on Computer Science Education (SIGCSE), 2008, pp. 116–120.

[59] R. A. Brown, Hadoop at home: large-scale computing at a small college, in: SIGCSE Technical Symposium on Computer Science Education (SIGCSE), 2009, pp. 106–110.

[60] A. Rabkin, C. Reiss, R. H. Katz, D. A. Patterson, Experiences teaching mapreduce in the cloud, in: SIGCSE Technical Symposium on Computer Science Education (SIGCSE), 2012, pp. 601–606.

[61] A. Iosup, D. H. J. Epema, Grid computing workloads, IEEE Internet Computing 15 (2) (2011) 19–26.

[62] A. J. Hey, A. Trefethen, Data Deluge: an e-science perspective.

[63] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, D. H. J. Epema, Performance analysis of cloud computing services for many-tasks scientific computing, IEEE Trans. Parallel Distrib. Syst. 22 (6) (2011) 931–945.

[64] I. Drago, M. Mellia, M. M. Munafò, A. Sperotto, R. Sadre, A. Pras, Inside dropbox: understanding personal cloud storage services, in: Internet Measurement Conference, 2012, pp. 481–494.

[65] D. Nurmi, J. Brevik, R. Wolski, Modeling Machine Availability in Enterprise and Wide-area Distributed Computing Environments , Tech. Rep. CS2003-28, Dept. of Computer Science and Engineering, University of California at Santa Barbara (2003).

[66] D. Feitelson, Workload Modeling for Computer Systems Performance Evaluation, e-Book, `http://www.cs.huji.ac.il/~feit/wlmod/`, 2009.

[67] C. Grimme, J. Lepping, A. Papaspyrou, Prospects of collaboration between compute providers by means of job interchange, in: 13th Job Scheduling Strategies for Parallel Processing, Vol. 4942 of Lecture Notes in Computer Science, Berlin / Heidelberg, 2008, pp. 132–151.

[68] D. G. Feitelson, L. Rudolph, U. Schwiegelshohn, K. C. Sevcik, P. Wong, Theory and practice in parallel job scheduling, in: Proceedings of the 3rd International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP '97), Springer-Verlag, London, Seattle, WA, 1997, pp. 1–34.

[69] H. Li, D. Groep, L. Wolters, Workload characteristics of a multi-cluster supercomputer, in: Proceedings of the 10th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP '04), Springer-Verlag, Berlin, New York, USA, 2004, pp. 176–193.

[70] S. Srinivasan, R. Kettimuthu, V. Subramani, P. Sadayappan, Selective reservation strategies for backfill job scheduling, in: 8th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP '02), Vol. 2537 of LNCS, London, UK, 2002, pp. 55–71.

[71] M. Bouguerra, T. Gautier, D. Trystram, J.-M. Vincent, A flexible checkpoint/restart model in distributed systems, in: 9th International Conference on Parallel Processing and Applied Mathematics, Vol. 6067 of LNCS, Springer, 2010, pp. 206–215.

[72] C. Loomis, The grid observatory, in: GMAC '09: Proceedings of the 6th international conference industry session on Grids meets autonomic computing, ACM, New York, NY, USA, 2009, pp. 41–42.

[73] B. Zhang, A. Iosup, J. Pouwelse, D. Epema, The peer-to-peer trace archive: design and comparative trace analysis, in: Proceedings of the ACM CoNEXT Student Workshop, CoNEXT '10 Student Workshop, ACM, New York, NY, USA, 2010, pp. 21:1–21:2.

[74] Y. Guo, A. Iosup, The Game Trace Archive, in: NETGAMES, 2012, pp. 1–6.