The Purge Threat: Scientists' Thoughts on Peta-Scale Usability

Alexandra Holloway
Jack Baskin School of Engineering
University of California, Santa Cruz
1156 High Street · Santa Cruz, CA 95064
fire@soe.ucsc.edu

ABSTRACT

In high-performance scientific computing, users output millions of files per project or simulation, resulting in petabytes of information. Little is known about how users make sense of it all, and what the major usability issues are in interacting with a file system at scale. We conducted interviews with scientists at national laboratories to identify common practices and issues with current peta-scale file system usage. The major usability problem encountered in the interviews was the purge threat, triggered when the parallel file system reaches capacity, and warning users about impending data loss. We show that the threat is not communicated to the users of the system in a meaningful way. We present three methods scientists used to address the purge threat—analysis, automation, and subversion—and discuss how subversion of the purging system is a clear indication of its lack of utility and indicative of its cognitive complexity. We define reactionary and cautionary archiving and draw a parallel between archiving methods and data production paradigms. Finally, we propose two non-hierarchical file and directory representation models to address the purge threat.

Categories and Subject Descriptors

H.5.0 [**Information interfaces and presentation**]: General

General Terms

Human Factors

Keywords

Interaction, high-performance computing, peta-scale, file system, data purge

1. INTRODUCTION

The growing size of peta-scale file systems has introduced usability problems to the domain of high-performance computing (HPC). Little is known about the usability factors which scientists consider important, and what the human-computer interaction trend is in high-performance parallel scientific computing.

In the system studied through interviews, users interface the parallel peta-scale file system remotely, often from a primary remote location, such as an NFS server. The peta-scale scratch file system is not backed up, though the much smaller NFS system has nightly or more frequent backups. Data are written in parallel to the peta-scale file system and can be read in parallel or sequentially. Most often, data are transferred from the parallel system to the smaller serial system for analysis, and to tape for archival.

We conducted a series of interviews to consider the current state of interaction with the system, including usability concerns, and asked scientists how they would propose to modify the system to have a better interactive experience with it. The research questions with which we conducted interviews were as follows.

RQ1. How do scientists interact with the parallel file system currently? Users and developers mainly interact with the system through the command line, with scripting and automation a major part of the interaction paradigm. The use of graphical interfaces among developers of high-performance computing applications is rare.

RQ2. What are the biggest usability problems of the peta-scale file system? Peta-scale file systems introduced problems related to decision-making and usability surrounding file system purges and associated data loss, which we call the purge threat. The way that the gravity and the scope of the purge threat are communicated to the user are lacking in efficacy; participants discuss user interface enhancements including two alternate file system views (time- and space-oriented), presented in Section 5. Other issues are file lookup problems, system performance, and trusting the system. Finally, the command-line user interface was not seen as a problem, but as a control-affording benefit of interacting with the system.

RQ3. How do scientists address the major usability concerns? Three ways scientists deal with the purge threat are *a*) analysis, in which users archive manually with significant thought and attention paid to the affected files; *b*) automation, when users archive automatically via a script; and *c*) subversion, when users perform specific actions to place their own data out of risk at the expense of other users' data and system integrity. System subversion is a keen sign that the archival process lacks utility. Users' archiving patterns are reactionary or cautionary in nature, paralleling data creation paradigms. Reactionary archiving is used more heav-

PDSW'11, November 13, 2011, Seattle, Washington, USA.

ily than cautionary because of the length of time that jobs run, and the cognitive load on analyzing, moving, and remembering files.

RQ4. What are the valuable usability features of the peta-scale file system? We found that scientists value an intimate knowledge of their own state in the system (especially in relation to their files' integrity), command-line compatibility, remote access, speed and robustness, and user control and freedom.

2. RELATED WORK

As file systems grow and expand to accommodate petabytes of data generated by scientific computing and simulation, usability problems emerge relating to the number of files and how they are accessed [1]. Gibson's and Norman's methods are helpful in understanding users' perceptions of a system. Gibson's affordances are defined as an action possibility or offering – that is, interface elements that communicate to the user the intended design of the element. Norman's affordances suggest actions within a system, intended or otherwise, and are perceived as suggestions. There is a distinction between system utility and usability: affordances relate to the usefulness (or utility) of a system, while the information the system presents specifies its usability [2]. In this paper, we wish to focus on the usability of the current peta-scale file system and methods of user interaction.

3. PARTICIPANTS AND METHOD

Thirteen participants were recruited from Los Alamos National Lab (LANL) and affiliated projects in July 2011. Four of the participants were interviewed in a group (the research team together); the others were interviewed one at a time. Hence, there were 10 interview sessions at LANL. Two of the participants were developers; nine of the participants were users; one participant had a dual role of both user and developer; and one participant was a file systems researcher (neither user nor developer) and is marked as an affiliate.

Four more participants—a consultant, two users, and a person in a unique role of being a user of his own development tools—were interviewed at Lawrence Livermore National Laboratory (LLNL) in September 2011.[1] In the text below, the participants' real names were replaced with pseudonyms, with the four-person research team represented by one name. One participant (Erin) was female (though her teammates were male), and all other participants were male. The summary of the participants is found in Table 1.

One to four participants at a time were brought in to a room for about an hour each. The LANL interviews were conducted by four researchers in person and one by phone; the LLNL interviews had one interviewer.

Participants were interviewed in a semi-structured setting, with certain themes and topics discussed at length as the opportunity allowed. Questions included describing the home directory structure and locations of certain key files on the different file systems, strategies of when to back up (archive) or purge documents, naming schemes, and sharing and permissions. Participants were asked about their opinions of how the knowledge of their interactions with the file system will be passed on to other scientists. Finally, we asked participants to describe properties of the ideal file system. As tape recording was not permitted, themes were extracted from written notes.

[1]We received IRB exemption for these studies. Exemption HS1671.

Pseudonym	Org.	Role
Aaron	LANL	Developer
Bruce	LANL	Developer
Charlie	LANL	User
Donald	LANL	User
Erin's team	LANL	User Team
Farhad	Affiliate	Researcher
Grisham	LANL	User
Harry	LANL	User
Ian	LANL	User
Jake	LANL	User, Developer
Kelsey	LLNL	Consultant
Leslie	LLNL	User
Mark	LLNL	User
Nate	LLNL	User, Developer

Table 1: Participants and their roles

4. USABILITY CONCERNS

The major usability issue in using large-scale file systems is how to create, store, and access the files created by the user and by running experiments on an HPC system. When an experiment runs, it potentially creates millions of files. Restart files, which are like save points or checkpoints generated at certain intervals to allow the user to re-start a simulation from a later point in the experiment, make up the majority (75%) of the data written to disk and can be the same size as the job's memory size. Restart files are persistent and, as we will show, are kept in great numbers as a strategy. Visualization dumps are much smaller, with 1% to 10% of the size of a restart dump and comprising the bulk of *productive I/O* – data the user needs to perform analyses and draw conclusions [3]. Other files include time history files, parallel output data, as well as any data the software outputs directly (i.e., standard out) (Nate). The exact number of files created by an experiment depends on the time steps and the number of processors, leading to potentially millions of files associated with a single project (Kelsey). Moreover, some apps use a single file with all results and data values aggregated into it; others use multiple input files. One participant suggested that 90% of these data are never used after creation (Kelsey). Our interview results confirm previous work in this area [1].

There are two reasons so much data, much of it intermediate, and the majority of it unread, are kept. First, keeping frequent checkpoints is important to prevent loss of time. In the event of a file system crash (the parallel system is not backed up), or if an experiment is progressing abnormally, a scientist must roll back to the last "good" checkpoint without wasting a lot of compute time. Second, scientists must save all data that led to a decision – that is, they must be able to reproduce all results or show proof that results were obtained through deterministic means [4]. This pre-emptive saving of data is referred to as *defensive I/O* [3]. It is not until much later that a user knows whether or not the data are worth keeping. In some cases, a checkpoint can be deleted after then next subsequent checkpoint is created; in other cases, scientists wish to keep checkpoints for longer so that an earlier time in the simulation can be accessed (Nate). Figure 1

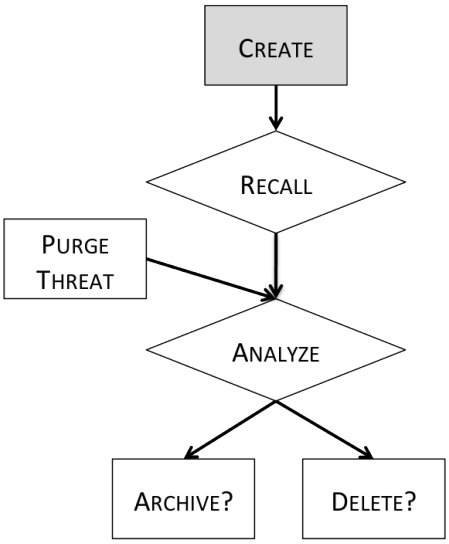

Figure 1: Create-analyze-purge work flow diagram

shows the work flow diagram of creating data (a side-effect of running an experiment), recalling the data, analyzing the data, and deciding between archiving and deleting it.

4.1 Addressing the Purge Threat

When the parallel scratch file system fills up, the system must decide which files to keep and which to discard – thus purging old files. Data are scheduled to be purged from with a least-recently-accessed policy. The list of files that will be affected are published in a file on the file system, and the system users are responsible for checking if any of their files are affected by the purge. If they are, the users must decide between archiving the data or deleting it. Thus, the decision is transferred to the users, and poses a real usability concern. If users do nothing, the system deletes the least accessed files, regardless of their relative location on the system and importance to the user. This threat of data loss is what we call the *purge threat*. The users' decision-making strategies surrounding the purge threat on the peta-scale file system are the focus of the following sections.

Participants named three ways of addressing the purge threat: analysis, automation, and subversion. Interestingly, not one participant named the fourth method—doing nothing and letting the files perish—as a valid way of dealing with the purge threat.

1. *Analysis:* Considering the affected files and moving them manually into archival storage (i.e., to tape).

2. *Automation:* Archiving some or all files affected by the purge with an automated script to move files regardless of contents (thus circumventing the analysis step in Figure 1).

3. *Subversion:* Using the touch command to refresh the access date on the files, thus removing the threat to one's own files and passing the threat on to a colleague.

The latter point was particularly noteworthy because it showed that the system was not working for the users, and users were deliberately subverting the system due to its lack of utility. This has an interesting parallel to security systems: despite traditional or novel security and permission measures, a system's security may be threatened if the sharing system is not easy to use [5].

Nate said he manually moved affected files, sorting through them carefully: if the size of a data set was large, for him, that translated to "real money in tapes." He said it forced a conscious decision to migrate the data. Other participants had a more disaffected approach to space and set up cron jobs to migrate the data automatically, with the feeling that when a file is archived, it is gone forever, never to be accessed again – though preserved in the unlikely scenario it is needed in the future. Charlie, Leslie, and Mark were pained by the latency to archive data saying it can take days, but Jake had no such concerns because his archival jobs completed overnight.

4.2 Archiving With the Purge Threat

From the conversations with scientists we gleaned that there were two archiving methods for data (i.e., reasons for moving data from the parallel system to long-term storage on tape).

- Cautionary archiving was meant to protect against system crashes or other sources of unanticipated data loss, as the parallel storage system had no backup source.

- Reactionary archiving was a result of a system purge threat and was done to guarantee that no data loss would occur for the reason of a scheduled purge.

One reason archiving was seen as the most pressing issue on the HPC system was that, once objects have been archived, accessing them is slow and difficult. Hence, reactionary archiving was mentioned in interviews more often than cautionary archiving. The overall feeling from the participants about archived data was that documents in the archive were gone forever though could be recovered with great pains if necessary. Mark related the work flow for archiving data (shown in Figure 2):

1. Knowing the oldest files will be purged, the user must determine the affected files and their size.

2. The user must decide whether the files are meant to be kept.

3. The user must determine where to put these files and how he or she will remember where they were placed

4. The user must move the files and subsequently (often, days later) verify that all files were successfully moved.

Mark and all other participants recounted the inefficient and complicated procedure surrounding archiving.

4.3 Other Issues

Participants mentioned several other issues as contributors to poor utility.

Usability and file lookup. As a scientist creates more and more files, experiments, and runs, it becomes increasingly difficult to recall file locations in the system, especially across multiple systems. Aaron said, dismissively, "Bookkeeping and file management is the user's problem," and Nate agreed, adding that each scientist has to come up with his or her own system over time. "I had lost a lot of data early on," he

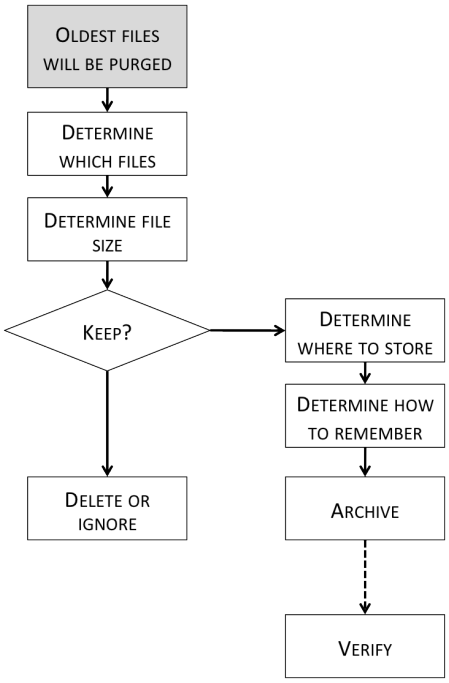

Figure 2: Deciding to archive work flow diagram

said. The methods used by the participants to keep track of the experiment particulars and resulting simulations' output files varied widely and included directory structure or hierarchy to describe experiment parameters, electronic notebook or `README` file, explicit file names containing meta-information about the experiment, and paper notebooks [6]. Having to keep track of file names and their relative locations was considered time-consuming and cumbersome.

System performance. The overabundance of files, numbering in the hundreds of millions of files in the user base (Kelsey), generate usability problems in the areas of space and I/O performance. All participants had experienced a loss of performance related to space considerations (such as running out of space and hard directory limitations). Performance loss manifested itself as severe latency when listing a directory, for example, to an inability to complete the command when reaching wildcard limitations.

Trusting the system. Trust (i.e., integrity and availability) was an important aspect of a file system (Donald, Mark, Nate). Participants said the storage system must be trusted against crashing, data corruption, and file system downtime, and system contents should be visible for verification whenever desired, with minimal latency.

5. PROPOSED SOLUTIONS

Participants had some ideas about solutions to the usability concerns mentioned in Section 4. Usability enhancements participants considered included alternate representations of the directory structure and graphical or more user-focused interfaces.

5.1 The Purge Threat Has No Solution

Kelsey warned about the purging problem, citing it as the biggest danger at the peta scale. The system purge associated with a file system filled to near capacity was a major topic of discourse, yet no real solution was proposed to it. Participants mentioned some usability enhancements to make the purge threat less threatening.

> At the next level of scale [in the ideal system], walking through the file system and directory structure, we've thought about how we do our clean-up... The next generation may be the breaking point from "barely doable" to "what do we do next?" (Kelsey)

It is clear that archiving is a performance issue with a bottleneck at I/O. But is archiving a usability issue as well? One of the problems with the purge threat is that the user is not notified well. A system purge is important because potentially, files are permanently deleted – files that the user created either through reactionary or cautionary means. The following are among the reasons the purge threat is a usability problem in HPC.

1. The user must retrieve the information about affected files, rather than have the list delivered.
2. The user may not understand the gravity or seriousness of the threat (that files will be deleted unless user action is taken).
3. The user may not understand the scope of the treat (which files are at risk).

In the section below, we discuss two participant-proposed representations of the file system to help users address the purge threat.

Time-oriented file structure. When the system reaches capacity, the system administrators publish a list of files that will be affected at the next system purge. It is up to the user to check if any of his or her files are on the list. Nate argued that the list should be "in your face:" it should be pushed to the user and the user should be notified, rather than leaving it up to the user to retrieve the information. Furthermore, the list should be of an acceptable granularity so as not to overwhelm the user with pages of file names that are irrelevant or repetitive. Kelsey proposed a policy-driven time-oriented view of data, that would display documents "without having to walk the directory structure," avoiding the bottleneck. Thus, the pre-indexed and cached time-oriented directory structure would display the files in chronological (last access) order and would highlight the files that are affected by the next upcoming purge. The user could then browse the files in any chosen granularity.

Space-oriented file structure. Automatic migration would be a benefit to some users, though others feel that the process of selecting files for archival or deletion, thus separating "important" from "unimportant" results and data, is a necessary part of the job. Nate proposed a tool that would quickly display each directory's size, which would be useful in determining the larger files one is storing on the parallel system. Removing the largest file may help to alleviate the purge threat. One can use `ls -laSh` to sort files in decreasing order, but the bottleneck is in walking the directory structure. This tool would pre-index and cache, creating a sort of linked list of files arranged by size. When looking for the largest (or smallest) directory or file, to delete or archive, the tool would answer the question, "How far down (in the hierarchy) must I go to reach a directory with at least the specified size?"

5.2 Addressing User Experience, Expertise

At LANL, most participants seemed confused when asked about the user interface: The UNIX-style command-line interface (CLI) was seen as an integral part of the experience. It was so integral that users of the system tended to forget all about it, even when using it for multiple hours a day. At LLNL, the discussion about the dream system was much more focused and all participants admitted to having spent some time thinking about the issue previously.

Despite the feeling that the user interface is an immutable part of the computing experience, participants brainstormed changes they felt would be beneficial.

We considered the topics of how user experience and expertise contributed to the participants' feelings about system utility. Expertise was loosely defined by the amount of code written, from a user that writes only small scripts to a developer that contributes to a project with thousands of lines of code. Donald, Jake, and Nate suggested that scientists at the labs are intimately familiar with the current system and would need a very compelling reason to use something otherwise. Jake suggested that there is no such thing as an ideal system, saying, with a smirk, that the dream has already been achieved. Donald said that what he uses now works, and he—or anyone—would need a compelling reason to switch: "I would use a GUI to a limited degree, but I don't see the reason." Jake said, "There's nothing I can't make work for me. I'm not bothered by doing things manually." He was referring to the system being able to work around the oriented user – that is, the user that is able and willing to learn a particular system and to use it for its strengths. Participants preferred the current command-line interface to the system over any new or "improved" interface because they would rather have robustness and limited features over a feature-rich system that performs poorly (Jake, Leslie, Mark).

User interface. In Nate's group, the rule is to show and replicate all GUI-based instructions on the command-line so that the tool becomes a crutch and can be discarded with practice. GUIs were considered slower to use, offer less control, and are more cumbersome to use remotely (Donald, Erin, Nate); however, GUIs are useful for monitoring experiments and viewing the summary of results (Erin). Erin's team and Nate noted that the CLI is the preferred way to interact with the system because "a GUI slows me down" (Nate). Donald said he is used to using a command line and feels more control than when using a GUI. He said that GUIs are "nice" with limited use of the system, but with advanced users that interact with the system daily for extended periods of time, GUIs are not useful. Kelsey, who uses a GUI called Hopper that aids in file transfer and management by making the underlying protocol transparent to the user, said he has seen both novices and experts use it. Bruce suggested that a graphical interface would be useful, but only for those who are less experienced or tech-savvy – the "manager button," for managers that have less computing experience than their team members.

Fast context- and content-aware search. When we asked questions not directly related to the user interface, participants at LANL had a different, and more explicit, answer. At LANL, where scientists use Macs as their primary desktop machines, participants thought Spotlight-like search would make an excellent contribution to the user experience (Erin's team, Grisham, Ian, Jake). Spotlight is a virtually-indexed system-wide desktop search for Mac OS X, using system metadata, extended metadata, and file contents. Grisham and Jake both used their Macs specifically because of fast searches by context and contents – in fact, when a job completed, both moved the results from the parallel file system to the home directory, and then to the Mac desktop, as an easily-searchable archival space, for inspection. Ian also mentioned the discontinued Google Desktop model as useful and spoke of it interchangeably with Spotlight search. At LLNL, where the primary desktop was a Windows-based PC, speculation around Spotlight was replaced with the Google search paradigm but still echoed the concerns of the LANL participants. Mark said: "The Google [search] approach opens up a world of possibilities in terms of non-hierarchical file management."

6. CONCLUSION

On interviewing the participants, we considered how the kinds of interactions that scientists have with their file systems on a daily basis informs the storage interfaces that they use. We found that most participants claim to be satisfied with their interactions with the file system, but at least three participants admitted to moving a small subset of their data to their laptops or desktops to do the "real" interactions. Interestingly, interactions, perceived problems, and solutions to problems at the peta scale were similar across participants from two national labs.

The most pressing concern was the *purge threat*, fear of data loss in a mass deletion of least-accessed files triggered when the non-backed-up parallel file system fills up. Participants identified three ways to address the threat, including *a*) analysis and manual archiving, *b*) automation by archiving through scripts, and *c*) subversion through passing the responsibility on to a colleague by changing the access time on files; and we indicated that the means of addressing the purge threat do not meet participants' usability demands. We identified two paradigms of decision influences on archiving: namely, *reactionary* archiving and *cautionary* archiving and showed heavier bias towards the former due to inefficient task breakdown surrounding archiving.

We showed that participants view the purge threat as poorly-communicated to the users, and identified three reasons for its poor usability: *a*) The user must retrieve the information, rather than have the information delivered; *b*) the user may not understand the severity of the situation and the necessity for user action; and *c*) the user may not understand the scope of the threat. The threat should be presented to the user clearly and concisely, and the severity of the effect of the purge should be conveyed.

Although no solution to the purge threat was proposed, enhancements to the user interface were discussed: *a*) a time-oriented rather than hierarchical view of the file system, including a concise grouping of the affected files pushed to the user, and *b*) a size-centric view of the file system.

Other participant-identified concerns included file recall, such as remembering where files are, recalling them, and keeping track of them; and other issues such as trusting the system and system performance, including hard constraints such as wildcard limitations. File lookup and overall system utility was affected by user expertise. The more the participants contributed to code, whether from a development role or through scripting, the more the participants were set with respect to their interface and their unique systems

of storing file information for later retrieval. Participants with little coding experience were more likely to claim to be adaptable to a different interface; those with more experience were unwavering in their laud for the command line. Overall, scientists were found to be intimately familiar with the current system and would need a very compelling reason to switch. Unfortunately, analysts and visualization experts were not included in the interview set; these personnel may be more adept in (or prefer) a graphical user interface.

All but one participant were reluctant to be accepting or excited about a potential new user interface on their existing file system or even their ideal file system. The scientists were adamant about keeping the CLI a top priority for any project changes. The reluctance to change was explained by the scientists' priorities in their interactions with browsing and searching a file system: *a*) command-line interface compatibility; *b*) remote access; *c*) speed and robustness; and *d*) user control and freedom. However, the way users find files clearly needed improvement as users were found to augment, with pencil and paper, the file system to keep track of files. Better searching algorithms and interfaces, including context- and content-sensitive search common in popular search engines, were proposed. The user interface was not seen as a problem; instead, it was seen as a way to afford freedom of expression.

7. FUTURE WORK

We will study the topic of usability of peta-scale systems with more participants in different labs, and with survey. We will use a focus group following the Delphi method [7], which will provide depth of data about user experience and predict the future of scientific human-computer interaction. Additionally, surveys distributed to scientists both at LANL, LLNL, and at other facilities will provide a general idea of the current attitudes on interacting with peta-scale systems.

Acknowledgments

This material is based upon work supported in part by: the Department of Energy under Award Number DE-FC02-10ER26017/DE-SC0005417, the Department of Energy's Petascale Data Storage Institute (PDSI) under Award Number DE-FC02-06ER25768, and the National Science Foundation under awards CCF-0937938 and IIP-0934401 (I/UCRC Center for Research in Intelligent Storage).

We extend our gratitude to Meghan (Wingate) McClelland for her assistance in the project and arranging the interviews at LANL. We wish to thank the interview participants at the two national labs for their time and contributions. Special thanks to Alex Nelson for his valuable insights into this manuscript. Thanks to and Caitlin Sadowski and Jaeheon Yi for their suggestions.

8. REFERENCES

[1] P. Gunda, L. Ravindranath, C. Thekkath, Y. Yu, and L. Zhuang, "Nectar: automatic management of data and computation in datacenters," in *Proceedings of OSDI*, 2010.

[2] J. McGrenere and W. Ho, "Affordances: Clarifying and evolving a concept," in *Graphics Interface*. Citeseer, 2000, pp. 179–186.

[3] Lawrence Livermore National Laboratory (LLNL), "Draft Statement of Work: Advanced Simulation and Computing (ASC): B563020," https://asc.llnl.gov/sequoia/rfp/02_SequoiaSOW_V06.doc, Livermore, CA, May 2008.

[4] I. Adams, D. Long, E. Miller, S. Pasupathy, and M. Storer, "Maximizing efficiency by trading storage for computation," in *Proceedings of the 2009 conference on Hot topics in cloud computing*. USENIX Association, 2009, pp. 17–17.

[5] M. Johnson, S. Bellovin, R. Reeder, and S. Schechter, "Laissez-faire file sharing: access control designed for individuals at the endpoints," in *Proceedings of the 2009 workshop on New security paradigms workshop*. ACM, 2009, pp. 1–10.

[6] C. Strong, S. Jones, A. Parker-Wood, A. Holloway, and D. D. E. Long, "Los Alamos National Laboratory Interviews," University of California, Santa Cruz, Tech. Rep. UCSC-SSRC-11-06, Sep. 2011.

[7] H. Linstone and A. Turoff, *The Delphi Method: Techniques and Applications*. Addison Wesley Longman Publishing Co., 2002.