Making Sense of File Systems Through Provenance and Rich Metadata

Technical Report UCSC-SSRC-12-01
March 2012

Aleatha Parker-Wood
aleatha@cs.ucsc.edu

Storage Systems Research Center
Baskin School of Engineering
University of California, Santa Cruz
Santa Cruz, CA 95064
http://www.ssrc.ucsc.edu/

Proposal for Advancement to Candidacy

UNIVERSITY OF CALIFORNIA
SANTA CRUZ

MAKING SENSE OF FILE SYSTEMS THROUGH PROVENANCE AND RICH METADATA

Proposal for Advancement to Candidacy

in

COMPUTER SCIENCE

by

Aleatha Parker-Wood

February 2012

The proposal of Aleatha Parker-Wood
is approved:

Ethan L. Miller, Chair

Darrell D.E. Long

Margo Seltzer

Daniel Tunkelang

Table of Contents

List of Figures

Abstract

Modern high end computing systems store hundreds of petabytes of data and have billions of files, as many files as the internet of only a few years ago. Even modern personal computers store numbers of files that would be massive for the largest mainframe computers of 40 years ago. The quantities of data in modern computing have long since overwhelmed anyone's ability to manage it manually, and the 40 year old tools currently in use for file finding and management are reaching the limits of scale. In an environment like this, secure, effective, and efficient search algorithms and automatic file management become a necessity, not a nicety.

Our proposal addresses the question of how users can quickly find and manage files, without burdening the file system with expensive brute force searches, or requiring the user to become an expert in query languages. We propose a number of algorithms to improve file management in a large scale scientific computing environment. By collecting new metadata, including file system provenance, we propose to provide new ranking algorithms which are efficient and effective on large multi-user file systems. We intend to reduce the burden of file naming, allowing the system to generate expressive, unique file names on the fly; we have identified a statistical property of data that is likely to select meaningful attributes for file names. And since security is a concern on many large scientific computing systems, we intend to analyze the security properties of the proposed ranking algorithms, and demonstrate how our ranking algorithm degrades gracefully from the ideal ranking when applied in a setting with restrictive security permissions. We will validate our results using real world scientific data, and provide statistical analyses of rich metadata and provenance from this data. And we will validate our ranking and naming algorithms through a series of *in situ* user studies.

Modern data management must be automatic and scalable, allowing users and file systems to focus on what each does best. By exploiting patterns of human behavior, the system can provide faster searches and more interpretable interfaces to the file system. Data growth is not expected to level off anytime soon, and file systems must be ready to handle the load.

Chapter 1

Introduction

Don't keep a man guessing too long—he's sure to find the answer somewhere else.

Mae West

High end scientific computing (*SciHEC*) systems face a crisis. While the computing and storage capacity of computing systems has grown by orders of magnitude, allowing users to create and store ever larger amounts of data, tools for finding and managing data have not kept up. Scientists are still manually naming files, sorting files into directories, and searching by navigating or using brute force tools. Manual file organization leads to millions of files in a single directory, overwhelming the traditional inode structure [43]. Scientists are still using `grep` and `find`, `awk` and `du` to find files and directories. These tools are brute force and slow, frustrating the user, and consuming valuable computing resources. Worst of all, the ineffectiveness of manual file organization and naming is resulting in users circumventing the file system all together, and tracking information outside it. Users embed metadata in file names and directories, or use PowerPoint, Excel, Notepad, and three-ring binders to track their files and the relationships between them [47]. Hours of scientists' time are wasted organizing valuable metadata outside the system, where no one else can use it, and computing systems are being overwhelmed by the burden of search requests. This metadata could be managed efficiently and robustly by the file system and used to help users find their files more effectively. Improving the quality of file management on SciHEC systems not only benefits the scientists who use the system, it benefits the system itself, freeing up computing resources for more important tasks. However, to improve file management, a number of advances are needed in the areas of ranking and file naming. In this proposal, we will describe research in ranking and file naming which will use new forms of metadata to make sense of file systems, and support secure, effective, and efficient search algorithms and automatic file management.

1.1 Ranking

Ranked search has a variety of advantages. It can improve the user experience by displaying important results early. It can also reduce the load on computing systems. By not requiring the system to produce a complete list of results for every query, computing power can be saved at query time. Additionally, if the user can find their files with the first query, rather than going through a lengthy query refining process, the computing system does not need to retrieve results for multiple queries. Ranked search allows the user to issue exploratory queries to investigate the contents of the file system without going through an expensive query process. Research on the web has shown that users who are *re-finding* items that they already know to exist can issue shorter queries which rank the desired item higher than in their initial search [48], which simplifies search for the user. If this also holds true on the file system, it can further save user effort. Finally, ranking provides a mechanism for determining which files are relevant to queries. Files which are unlikely to be relevant to any query may not be likely to be accessed soon, and so we will investigate if ranking can provide additional guidance for archival policies and caching policies.

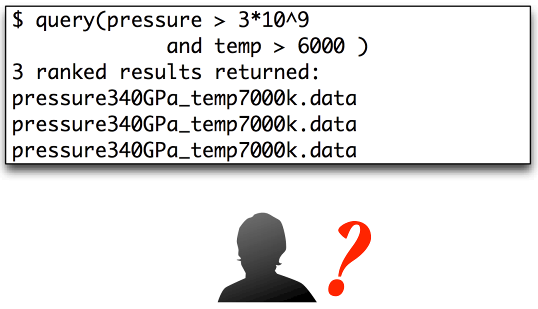
More sophisticated search tools are being proposed for SciHEC, which would allow users to access a database-like searchable file system [6, 11, 31]. These file systems use indexes to speed up search, and allow users to construct keyword or structured queries over metadata and content. But we are still left with a conundrum. When the user searches a billion files, how precise must their query be to find only the files they need, without sifting through tens of thousands of results? Modern scientists are not database administrators, or often even computer scientists [47], and are likely to issue vague queries which return too many results. They need the ability to issue simple queries and get back useful results immediately.

This very problem has been thoroughly explored on the Web for over a decade, which has trillions of URLs and billions of untrained users. When faced with the queries of unsophisticated users, the clear web search winner is ranked search. Rather than requiring users to learn an complex and opaque query syntax, and asking them for information they may not remember, ranked search allows users to issue a query with high *recall* (returning many results which match), and then uses heuristics to decide what the user is most likely to be looking for. Techniques such as PageRank [41], recommender systems, and personalization use the flood of data on the web to their advantage, leveraging it to make predictions about the average desires of users.

SciHEC, however, despite being similar in scale, has lagged in the development of search. In part this is because, despite the huge amounts of data produced, file systems capture a different kind of structure from the web, so traditional web ideas of importance could not be directly applied. Additionally, unlike *informational* tasks, which account for approximately 80% of web searches [26] and where any correct answer will often suffice, a file system user is often looking for a specific file and no other will do, similar to *navigational* [26] or *re-finding* tasks on the web [2, 48]. Where the web has a rich selection of links between pages to indicate what individuals think is important, file systems have stuck to simple POSIX metadata, such as directory hierarchies, file access time, file creation time, and file system links. Hierarchies tell us about relationships between files, but nothing about importance, and current directory management techniques require the same manual organization we are trying to save users from. While file access time gives us some measure of importance by capturing recency, it has no notion of access frequency. It cannot distinguish between one recent access and a thousand, and access time is often inaccurate, or not captured. File system links are rarely used by the users, and they give little indication of importance. What we really need is something which captures the notion of importance on a file system, the way links capture the notion of importance on the web.

Fortunately, there are likely candidates, if we are willing to capture additional metadata. By tracking what files people actually use, we can make better predictions about what they're likely to be looking for. One approach we propose in this work is time-decayed, probabilistic access counters. Small, easy to update, and forgiving of update inconsistency, these can tell us what files are both recently and often accessed. A second approach we will propose uses file system level *provenance*—tracking the data flows between files and processes—to measure file usage, both globally and on an individual basis. Both of these approaches can be used to determine file popularity over time, finding recently popular files and files which are accessed often over a long period. It can also be used to generate an "inverse popularity" ranking, where the user is looking for things which are infrequently used (perhaps looking for underused data sets to explore, or candidates for archival.) In addition, provenance capture allows us to track things on a user by user basis, allowing powerful algorithms for personalization and recommendation, and enabling identification of "working sets" which are often accessed together, as well as providing a new class of useful queryable data for the user.

File systems have other differences from the web. Where the default behavior of the web is to make everything available to everyone, all the time, file systems have a much more complex security model. The contents of files can only be made visible to certain people. The *names* of files can only be made visible to certain people. It's been demonstrated that ranked search can leak information about file contents and metadata [13]. Our research [47] and that of others [12] suggest there is a wide range of sharing in SciHEC, from fully open research to highly classified.



```
$ query(pressure > 3*10^9
            and temp > 6000 )
3 ranked results returned:
pressure340GPa_temp7000k.data
pressure340GPa_temp7000k.data
pressure340GPa_temp7000k.data
```

Figure 1.1: Manually named files using metadata can result in confusion at search time.

Therefore any ranking algorithm proposed for a SciHEC file system must address concerns about information leakage. Our previous work has already shown one mechanism for producing secure file system search [42]. We will show how this is applicable to our ranking algorithms, and suggest a refinement which also obscures usage information.

Ranked search in an indexed file system has the ability to significantly reduce the burdens of users and systems. However, it is only a piece of the file management puzzle. Next, we discuss another major problem users face, effectively naming files.

1.2 Naming

File naming is a burden that is often taken for granted. Users spend time agonizing over what words and directory structure will help them (or worse yet, others) find a given file or directory again in the future. File names and locations must be chosen in advance, and convey context and meaning to a future self who may or may not remember the circumstances leading to the file's creation and contents. To combat the problem of losing data, users create *ad hoc* ontologies and tags, building elaborate directory structures and embedding every possible piece of metadata into file names, hoping that at least one piece of the chosen metadata will be the item they need to jog their future memory. This often results in very long file names, which create portability problems as users move files between file systems with different character limits [47]. These naming strategies are error prone, as users make mistakes, swap field names, or omit fields they will later need [47].

Faster search can help with part of the file finding problem, by allowing the user to specify criteria and quickly find files anywhere in the file system, rather than limiting the user to brute force searching within a directory structure. However, users must still name files by hand, with the associated effort and potential for error. When files are returned from a search, they must still be uniquely distinguishable from each other or risk name collisions as shown in Figure 1.1. Even in a directory structure created by hand, it may still not be obvious what is unique about a given file, and current automatically generated directory names are often painfully opaque. Finally, uniqueness alone is not enough to generate a good file name. Consider the inode number. This is unique, but not at all meaningful to the user. How can we create a file name that will help the user distinguish the file, and remind them of its contents?

In order to help users distinguish their files more effectively, we propose a scheme for dynamically naming a file, either at creation time, or at search time. By analyzing the statistical properties of metadata on a file type by file type basis, we hope to select attributes which are both unique and meaningful to the user. In order to do so, we will establish and exploit the following properties of metadata:

- Metadata produced by natural processes, such as human activity, is likely to follow a Zipfian distribution [51]. Metadata created by computers for computers, is commonly uniform by design, such as inodes.
- Metadata produced by humans is likely to have a vocabulary size described by Heaps' law [23].
- Empirical data shows that the intersection of two uncorrelated Zipfian distributions of metadata is near uniform [35].

The first two properties allow us to suggest metadata which is human-like, and potentially meaningful. The final property allows us to select attributes which will result in a unique name with only a small number of attributes. In combination, they make it possible to create short, unique filenames using metadata which will convey meaningful information to the user. By reducing the manual nature of file naming, we hope to further ease the data management burden of the user, reduce file portability problems, and make it easier to re-find files in the future, rather than losing files in a sea of similarly named items.

In summary, the volume of data in modern SciHEC file systems has overwhelmed the user's ability to manage it effectively. Searchable file systems fill part of the gap, but we are still left with the problems of presenting the user useful results quickly, and helping the user distinguish between files. In the following sections, we will describe how we plan to solve these problems (Section 2), discuss what work has been done in these areas already (Section 3), show some preparation and preliminary results in the area (Section 4), lay out our expected timeline for completing the necessary research tasks (Section 5), and conclude (Section 6).

Chapter 2

Proposed Work

I aim to misbehave.

Malcolm Reynolds, *Firefly*

This section describes our proposed work, focused on two key areas, ranking and file naming. These two areas are both crucial to finding and managing files in a large file system. We describe two novel ranking algorithms, one based on provenance, and one based on time decayed access counters. We also describe a novel security threat these two algorithms introduce, and describe our mitigation strategy. Finally, we conclude with our algorithm for dynamically generating file names, both at file creation time, and as a way of disambiguating search results at query time. This section focuses on the design and implementation aspects of the proposal, but a detailed timeline of tasks is also described in section 5.

2.1 Ranking

Ranking files effectively is the key to providing a good search experience. A good ranking algorithm allows the users to spend less time writing precise queries, less time sifting through a mountain of results, and allows the file system to dedicate less resources to answering queries. Below we detail two proposed ranking algorithms designed with SciHEC file systems in mind. Each requires additional metadata which is not currently collected by file systems. In addition, we detail the expected tradeoffs of each algorithm, and a proposed implementation strategy.

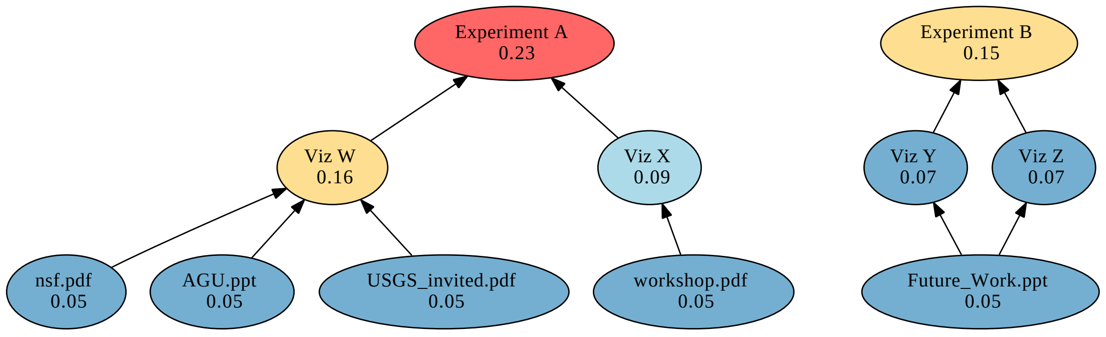


Figure 2.1: Applying PageRank to a provenance graph results in overemphasis on the roots.

2.1.1 Ranking via provenance

Provenance, as captured at the file system level [39, 44], is a rich source of information about the behavior of users within the file system. We propose to leverage provenance to provide a ranking of files within the file system by popularity, using an algorithm called Time and Provenance Informed Ranking (TaPIR).

Provenance consists of a directed acyclic graph (DAG) containing nodes which represent files and processes, and edges which describe the data flow between them. The *roots* of the provenance graph are the nodes which all other nodes are derived from. For instance, if provenance has been collected since the beginning of a system's creation, roots might be things such as the source code files which comprise the kernel. The *leaf* nodes are the most recent nodes, nodes which have not yet had anything else derived from them.

Much like links in the web graph do for the web, the provenance graph provides us with *endorsements* of files, proof that users use these files and find them important. We can use these endorsements to provide a likelihood that this file will contain the information a user is looking for, similarly to the way PageRank [41] provides a likelihood that users will find a given file useful.

Like PageRank, TaPIR uses the stationary distribution of the graph's Markov chain to determine the popularity of a given item. However, PageRank has some limitations which make it unsuitable for direct application. For instance, PageRank was designed with a graph in mind which can (and often does) have cycles, is constantly adding and removing links, and has very few nodes with either an in-degree or out-degree of zero. By contrast, the provenance graph is a directed acyclic graph with leaves and roots, and is largely static after creation. Links may be added, but are never removed.

Applying PageRank directly to a provenance graph, as shown in Figure 2.1, results in

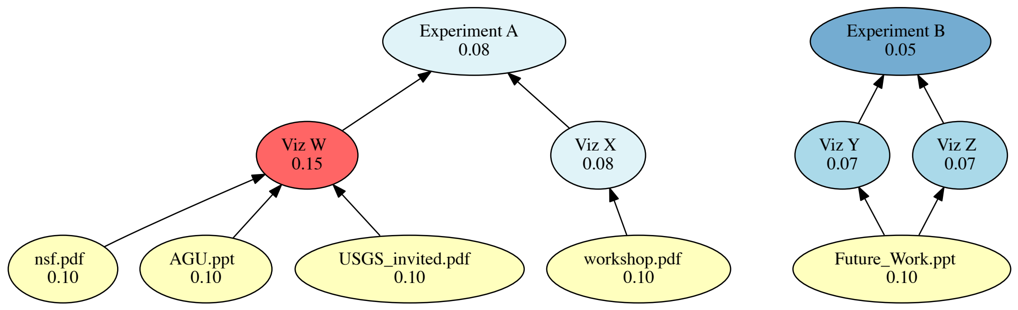


Figure 2.2: TaPIR favors newer popular files, and assigns a value to leaves, unlike PageRank's emphasis on the roots, which are often ubiquitous files and processes such as the kernel source.

commonly used roots being ranked as most important. However, previous research in desktop ranking, such as *Stuff I've Seen* [15], suggests that recently used files are the ones most commonly accessed. We surmise that an over-emphasis on older files is less useful than focusing on recently used files. In addition, since provenance is a DAG, there are a large number of leaves, nodes with in-degree of zero. While this situation occurs often on the web, in that context, files that are not linked to are often assumed to be uninteresting. By contrast, in a file system they are the files the users were most recently accessing, and are therefore potentially the most interesting.

Fortunately, both of these problems can be solved with a simple modification. For PageRank to compute the stationary distribution of the web, modifications must be made to the web graph, in order to make it a valid Markov chain (irreducible, aperiodic, and positive recurrent.) In other words, one must be able to get, through some number of clicks, from any page to any other page. To make this possible, PageRank introduces a *teleport function*, which serves as a virtual link from every page to any other page, and occurs some fraction of the time (traditionally 15%), rather than following a real link out of a page. In the standard implementation of PageRank, the teleport function has a uniform probability of transitioning from any page to any page.

In TaPIR, we utilize the teleport function differently, in order to solve the problems created by applying PageRank to a DAG. Instead of applying a uniform teleport function, the probability of teleporting is proportional to the current node's distance from a leaf node, and favors teleportation to leaf nodes. This has two benefits, as shown in Figure 2.2. Firstly, it prevents a strong emphasis on the roots. Secondly, it allows us to assign a higher value to leaves, despite not having anything deriving from them. Where two leaves have identical rank under this scheme, we can use the modification time as a tie-breaker statistic, favoring newer

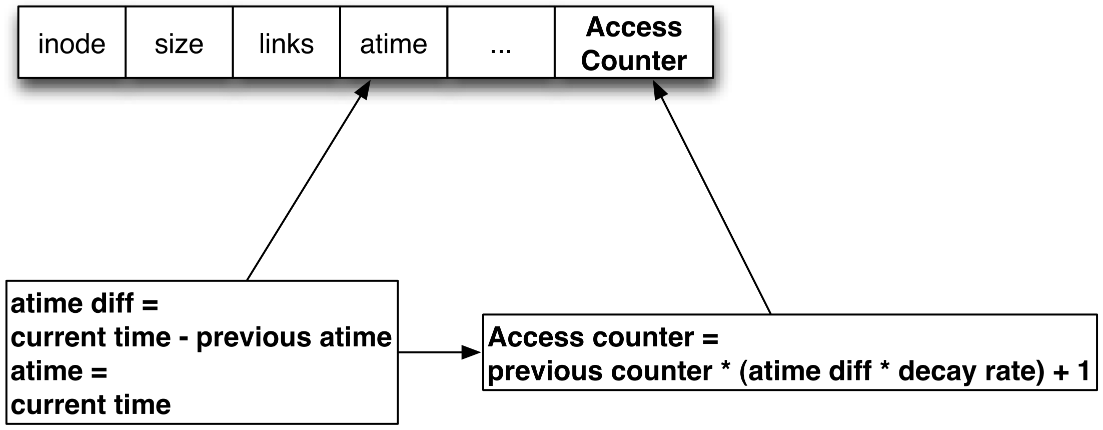


Figure 2.3: Access metadata is updated on access to the file. AccessRank uses best effort updates to both `atime` and `acounter`, but does not require consistency, unlike a typical implementation of atime. Access time is decayed by multiplying the time differential between `now` and the previous access by a decay factor, α.

files. While not all leaf nodes are guaranteed to be the newest files in the system, they are guaranteed to be newer than anything they inherit from, making height in the tree a proxy statistic for age.

This ranking algorithm gives us a elegant mechanism for ranking files, with a proven theoretical foundation, based on previous research about web ranking and what users are likely to use on file systems. First, users often look for recently used files, which this algorithm favors. Second, users often look for the same files over and over again. This scheme ranks frequently used files higher.

In addition to providing ranking at a global scale, this ranking scheme gives us additional information, which can be used to customize ranking and recommend files. We have enough information to focus on files primarily used by the current user, and the provenance graph can be used to improve recall, by suggesting files used by users with similar file access history.

2.1.2 Ranking via time-decayed access counters

While TaPIR has potential extensions for personalization and security, and provenance provides a rich interface to the file system, not all systems may be willing to incur the time and space overhead of provenance collection. As an alternative, we will present a light-weight ranking algorithm based on caching algorithms using time decayed access counters (AccessRank), as shown in Figure 2.3. Rather than using the provenance graph to determine file popularity, this tracks file accesses over time, emphasizing files that have been accessed frequently and recently. Unlike TaPIR, which focuses on *write usage*, files which have been the source of data for other files, AccessRank focuses on *read usage*, files which have been read, regardless of whether that read resulted in a write or not.

To implement this ranking algorithm, a new piece of metadata is added to the inode, consisting of a single integer, which we refer to as `acounter`. `acounter` is updated on a best effort basis. Each time the file is accessed, `acounter` is first decayed, using the following calculation:

$$\text{decayed counter} = \text{current counter} \times (\text{now} - \texttt{atime}) \times \alpha$$

where α is the decay rate. We will test on a range of α values, which will decay the counter by half at time spans between 1 week and 6 months. Once the counter has been decayed, it is incremented and written back. At periodic intervals, the system is scrubbed for recently updated atimes, and a global index containing `atime` and `acounter` is updated to reflect the new ranking order. Files are ranked according to the decayed value of `acounter` at the time of the query.

We are aware that `atime` is a notoriously unreliable piece of metadata in large file systems. Having multiple readers or writers to a single file makes it difficult to determine what the most recent access time actually is, and contention to update the inode can slow disk throughput and the OS. Many systems disable `atime` all together, or use `relatime`, which only updates `atime` if it has been a certain time interval since the last update. Likewise, a perfectly consistent access counter will require every file reader to update the inode, leading to similar performance problems. Fortunately, since ranking is an ordering function, it only requires that the relative ranking of files be consistent, and approximately correct. This allows a weaker set of consistency guarantees to be applied to the counter and atime, which we will investigate the impact of. In addition to testing AccessRankon the collected `atime` data, we will simulate the use of the `relatime` model for a variety of time intervals, and investigate the variation in ranking it would incur for our captured workloads. Additionally, since we do not require synchronous updates, updates to `atime` and `acounter` can be queued in a log and stored only in the ranking index, to prevent contention for the inode with other inode updates. We will also compare against similar caching algorithms such as Adaptive Replacement Cache (ARC) [37] and aggregating caches [4] to determine whether a caching algorithm alone is sufficient for ranking.

This ranking mechanism has the benefit of being simple to implement, and lightweight, while retaining the advantages of tracking time and frequency of usage. It does require enabling a weak version of `atime`, with the incurred penalties for performance. However, we believe that the performance will be faster than that of full provenance collection, and the weak consistency guarantees of `relatime` can be used to speed up performance.

2.1.3 Security

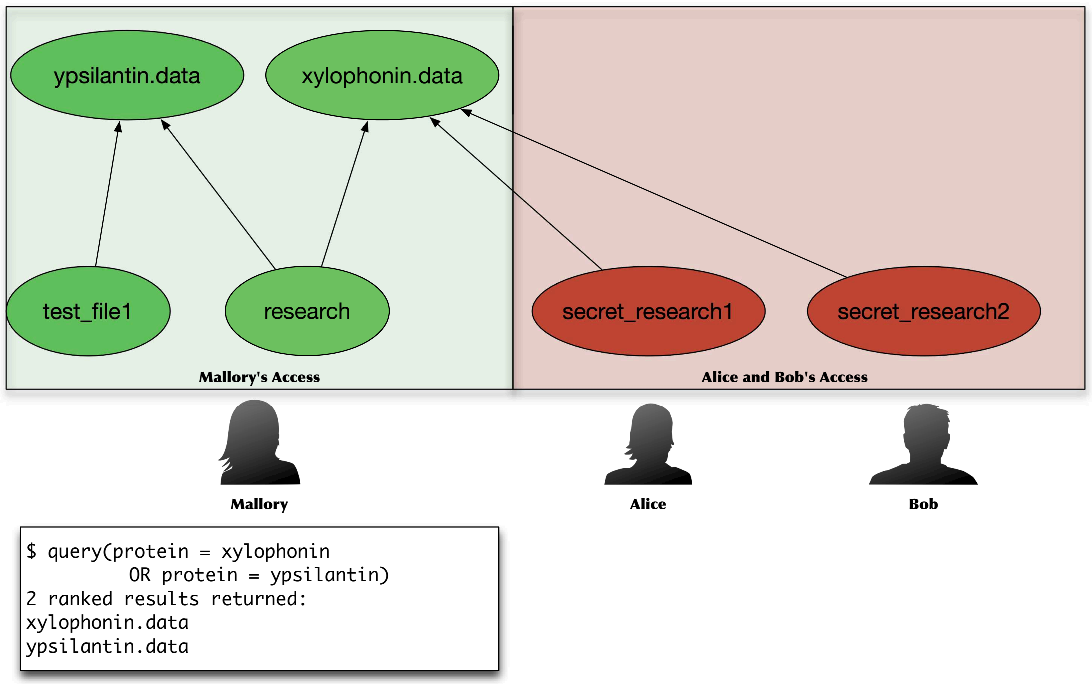
The security of ranked search is a known concern [13]. Usage based ranking is subject to two types of attacks, one previously known, which attacks the contents of files, and one novel, which collects information about the usage of files. We describe each, and our solutions.

Statistical manipulation of file contents can be used to extract information about the contents of files that are not visible to the searcher. We have previously demonstrated an index *partitioning* approach for hierarchical file systems, Security Aware Partitioning [42], which prevents this style of attack. In Security Aware Partitioning, the file system indexes are are partitioned into sub-indexes. This approach has been used in the past for speeding up search [35], by using in-memory Bloom filters to determine relevant partitions, without incurring the cost of seeking to disk to retrieve a full index. However, in Security Aware Partitioning, the partitions are chosen in order to meet security criteria. A partition has two properties. One, there is a distinct and known set of users and groups who can access it. Two, if a user can access any file in a partition, they can access every file in that partition. This means that security checks can be done at the partition level, before ever retrieving results. Each partition can maintain a rank sorted index without revealing information about other partitions and their files. Thus, Security Aware Partitioning prevents content-based statistical attacks on ranked search, while still allowing ranking to be calculated globally.

However, the unsolved threat in usage-based ranking is that of usage security. We describe a risk model for search, and discuss the implications for each ranking algorithm, then propose a solution for TaPIR which prevents usage information leakage. A solution for Access-Rank is out of the scope of this work.

2.1.3.1 Threat Model

While ranking using Security Aware Partitioning hides all information about the contents of files, it is still possible for a user to infer things about the usage pattern of files using either of our proposed ranking strategies. Consider the scenario in Figure 2.4, in which we describe a theoretical biotechnology company using TaPIR. User Mallory has access to file `xylophonin.data`, which contains information on the properties of a certain protein, xylophonin. Mallory cannot see any files except her own that derive from file `xylophonin.data`, but by searching for criteria that matches both file `xylophonin.data` and a newly created file `ypsilantin.data` (protein = xylophonin OR protein = ypsilantin), and observing their relative



```
$ query(protein = xylophonin
         OR protein = ypsilantin)
2 ranked results returned:
xylophonin.data
ypsilantin.data
```

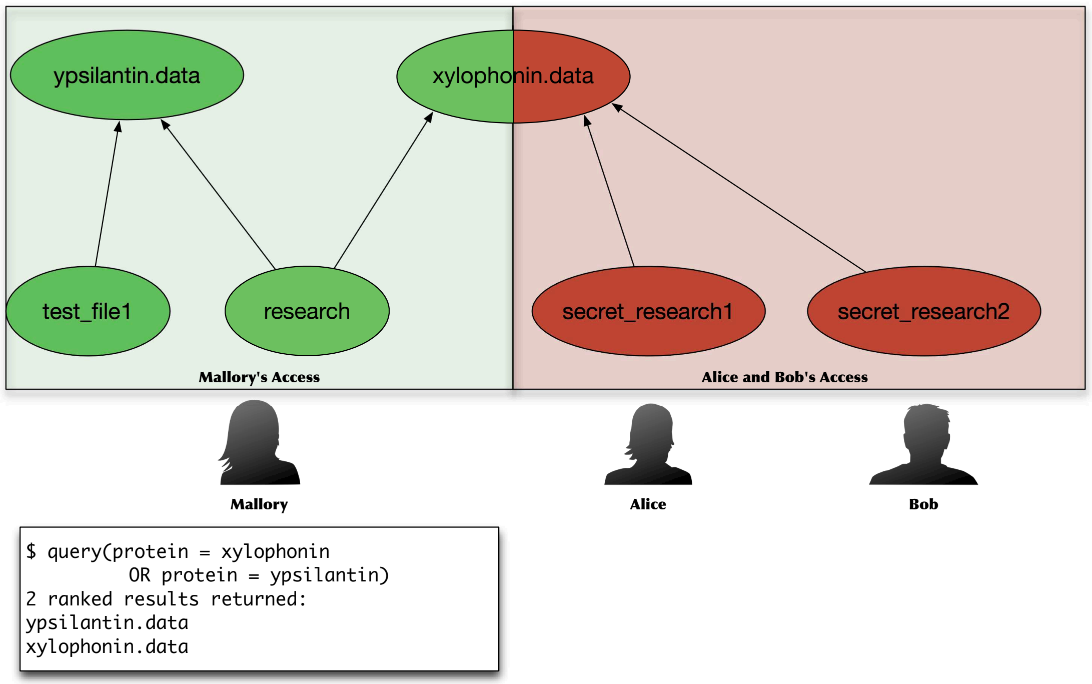
Figure 2.4: Provenance based ranking can provide information about the activities of other users that are outside of the current user's security scope. In this case, Mallory has learned that other users are doing research on a specific protein, despite not having access to their files.

rankings, Mallory can infer that another user is using file `xylophonin.data` more frequently than Mallory is using `ypsilantin.data`. With this information, Mallory may realize that there is additional research being done on the protein xylophonin, and perhaps even reverse engineer the nature of the research.

The threat model for AccessRank is similar, but focuses on read usage, rather than write usage. Mallory can again create a second file as a point of comparison to file `xylophonin.data`, and then access it repeatedly to drive its rank up. If `xylophonin.data` is ranked higher, she can infer that `xylophonin.data` has been accessed often and/or recently. In addition, by observing relative rankings over a period of time, she can infer whether the file is continuing to be used.

2.1.3.2 Solution and Evaluation

To prevent this for provenance ranking, we can partition the provenance graph itself, using Security Aware Partitioning on the provenance graph, rather than a file system graph, as shown in Figure 2.5. Each partition of the provenance graph is chosen such that if a user can access one node in the partition, based on their permissions, they can access every node,



```
$ query(protein = xylophonin
         OR protein = ypsilantin)
2 ranked results returned:
ypsilantin.data
xylophonin.data
```

Figure 2.5: By partitioning the provenance graph, and executing ranking only on the local partitions, we can prevent information leakage. In this case, we have duplicated the node for `xylophonin.data` in order to prevent usage information leakage.

and all the links between them. If they cannot access a given node, then all links from that node are hidden. Each connected tree comprises a separate partition. We can then perform TaPIR on the partitioned sub-graphs. How effective this ranking is depends on the distribution of security permissions on the provenance graph. As permissions grow increasingly restrictive, the ranking focuses ever more tightly on the usage of single users, and ever more nodes must be split. This has implications for the efficiency of the ranking algorithm, and for its effectiveness. Ranking calculations must be done on a larger number of smaller graphs, and we also wind up with duplication of nodes across sub-graphs, as some files will appear in the provenance of many files. However, as the size of sub-graphs decreases, it may be more efficient to perform the partitioning at query time, and calculate rank only on the relevant subgraph, reducing the amount of duplicate data that must be stored.

We intend to empirically study the expected efficiency of the ranking as a function of partition size, and to characterize the effect on file ranking. We will perform an analysis of security permission distributions in provenance graphs we will collect from a SciHEC file system. We will partition the provenance graph using Security Aware Partitioning [42], and examine the size of the generated subgraphs and their metadata entropy, to characterize search efficiency. We will utilize relevance judgements collected during the user study, and then evaluate the changes in mean average precision (MAP), and Discounted Cumulative Gain [27] which occur when TaPIR is applied to the partitioned subgraphs. This will allow us to quantify how partitioned provenance graphs affect search performance on a real world data set, and whether the degree of node duplication will be prohibitively expensive.

2.1.4 Data Collection

In order to evaluate the proposed ranking schemes, we need access counters and provenance traces from a HEC file system that is in current use. The data necessary for evaluation is similar to that captured by the Open Provenance Model (OPM) [38], with all edges annotated with time information, and with ownership defined as that of both the POSIX `user` and `group`. This is a subset of the information used by file system provenance capturing systems such as PASS [39]. While we will attempt to collect a fully PASS compliant trace for future research, it may not be feasible in all tracing environments, due to technical or security restrictions.

In order to capture a provenance graph we need the following pieces of information:

- A unique reference to each instance of an executable
- References to every input file used by an executable instance
- References to every output file created by an executable instance
- Timestamps when the output file was closed, process was run, or the input file was read.
- A unique identifier for the user and group which owns the process or file, in order to do security analysis.

This allows us to create a graph where each node has a unique label, and to track times in order to rank and break causality cycles.

Where possible, we will also attempt to collect:

- The command line and process parameters
- The process environment
- A complete description of the operating system and hardware
- Random seeds where applicable

Since no deployed file system currently collects provenance or access counters, it will be necessary to set up collection mechanisms. There are two possible scenarios. In the first scenario, computation is done on a central machine with its own storage, as in a typical SciHEC installation. In this instance, we only need to trace traffic at the central machine and to its file system, in order to collect all the needed data. In the second scenario, the users are connecting to a central file server, and then doing local computation which results in new content created on the file server. In this case, we must collect data both at the central file server, and at individual client machines. In order to do this, we propose two complementary mechanisms, central data collection, and a shim layer which will interpose between volunteer users and their file server. We will attempt to leverage existing provenance and usage collection code wherever possible, such as libraries from PASS [39] and Burrito [22].

Centralized computation

In the case where computation is done on a central machine, we propose to solicit volunteer users who will use a user space file system to communicate with the underlying file system. The user space file system can then collect all of the data listed above.

Central data collection

Where feasible, the easiest way to collect data is to directly instrument the file system itself, or capture packets at the file server. By intercepting process calls to `open` and `write` at the file server, we have visibility to every file read and write, and can associate them using the unique identifiers passed by the client to the file server. However, this does not permit us to collect a fully PASS compliant trace, since it cannot capture information about the process environment, the parameters to the process, the system libraries used, and so on. In order to capture these additional pieces of data, we still need to interpose at the client side. In addition, some protocols may not provide information about the user or process requesting the data, such as NFS. Therefore, we must use a shim on the client side to capture the necessary data.

Client-side shim

A shim is a small lightweight piece of code which interposes between the user and their file system calls. With a shim, we can capture much more detailed information about the user's computing environment, process parameters, and so on. However, it requires us to install code on individual users computing systems. This scenario is similar to the centralized computation model, but with the added complexity of heterogeneous computing environments.

2.1.5 User Studies and Evaluation

As this dissertation focuses on the interactions between users and the file system, user studies are necessary to validate the effectiveness of the algorithms. To evaluate a ranking algorithm, users must be given the ability to carry out search tasks on a data corpus, and then allowed to evaluate the relevance and ordering of results. We intend to collect provenance and usage data from a variety of installations, such as the UCSC astrophysics department, NASA, and potentially one of the National Laboratories. In addition, we will solicit users from each of these sites and from other locations, to carry out a user study that is appropriate for the type of scientific data collected. We will solicit astrophysicists to search astrophysics data, and so on.

Once the provenance and usage data has been collected, the files in the data set will have their metadata and content indexed into a database. Volunteer users will be asked to supply a set of queries, and list a set of relevant files. A set of query results will be obtained by running the user's query over the database and then ordering them by relevance and rank. We will use Okapi BM25 [29] for content relevance, and a binary match for metadata fields. The users will have the ability to do key-value queries over metadata (including range queries), and content queries.

The user presenting the query, and other users in the same field of research, will be asked to navigate to relevant files using existing tools (such as `grep`, `find`, and manual directory navigation) and the time it takes to find each file will be noted, as well as how many of the total are found. For each query, therefore, some percentage of users will be engaged in a discovery task, and some percentage in a re-finding task. The users will be then presented with a list of ranked file results. Users will be asked asked to identify all relevant files from the top N results. The top N results from each of the ranking algorithms will be merged, discarding duplicates, to generate a list of files. The ranked files will be presented in randomly interlaced order to prevent positional bias. So, for instance, if algorithm A presents file X at position 1, and algorithm B presents it at position 5, half the users will see X at position 1, and the other half will see it at position 5. If all ranking algorithms miss a file that the query creator deemed relevant, that file will also be added to the list of results, in the final position, in order to get relevance judgements from other users.

Once all users have offered relevance judgements, each algorithm will be scored on its ability to rank relevant files, using mean average precision (MAP), and Discounted Cumulative Gain [27], on the unmerged, ranked lists. Given relevance judgements, we can then also perform offline investigation of parameter tuning on the different algorithms, such as varying the α parameter for AccessRank and varying the teleport probability for TaPIR. Additionally, we will analyze the ranking algorithms for correlations, and investigate whether using the two in conjunction can result in further improved ranking. We will evaluate whether basic caching algorithms achieve reasonable performance for the same data sets. And finally, we will also use the relevance judgements to characterize the impact of provenance partitioning.

2.2 Naming

The problem of naming comes down to the following. Can the system present sufficient information to a user such that they can quickly and accurately identify if the file they are looking at is the file they want? In order to do this, the user must find *meaning* and *relevance* in the information that has been presented. Further, the name must be context dependent. If the user has searched for files with attributes $p = 1$ and $r = 5$, presenting three files named p1-r5.data will not assist them. In this case, the system should detect the problem and present additional context to help the user identify differences between the files. This problem is particularly relevant in the area of non-hierarchical file systems, since directories are no longer available to give context to a search result, and displaying all available metadata is prohibitively large.

We propose the following: a human is most likely to find meaning in data which is like data a human would have created. Human activity is well known to create data with Zipfian distributions [51], from the words in documents, to the distribution of populations in cities. Additionally, natural languages display a property known as Heaps' Law [23], which describes the size of an expected vocabulary given the size of a data set. We propose to present file names by choosing key-value pairs based on a selection of n attributes which are Zipfian in distribution and show a vocabulary size similar to that predicted by Heaps' Law (and are therefore human-like). We will favor attributes which are uncorrelated (such that their cross-product is uniformly distributed and unique.) This will result in unique, human-like file names. For example, at $n = 3$, on a test corpus of PDF documents, we might create a POSIX file name such the following: `author=aleatha-title=Dissertation Proposal-ModDate=2012-01-28T10:56:05-08:00.pdf`. Alternatively, we might present an ordered list at query time for disambiguation, such as `proposal.pdf (author = aleatha, title = Dissertation Proposal, ModDate = 2012-01-28T10:56:05-08:00)`.

We will experiment with analyzing distributions at different levels of granularity, from entire attributes, to analysis of items within attributes (such as analyzing words within a text field.) We will evaluate names based on these algorithms by presenting file/name pairs to a variety of users to determine whether the generated names are, in fact, meaningful. Additionally, during the evaluation, we will determine an empirical scoring function to account for the importance of distribution and statistical independence.

2.2.1 Metadata Analysis

In addition to conducting experiments on generating file names, we intend to do an analysis of the distributions of metadata for a variety of file formats, focusing on files popular in various scientific computing communities. (For instance, the FITS file format from astronomy.) This will allow us to analyze which types of metadata demonstrate Zipfian distributions, and how strongly correlated metadata is within a given context.

We will collect a variety of files which are known to have well structured metadata from multiple different disciplines, such as FITS files from NASA, geological surveys from USGS, and so on. We will also examine files from non-scientific data sources, such as email and MP3s. The data used will be a super-set of the data used for file naming experiments.

We will apply goodness of fit tests (*i.e.* Pearson's χ^2 test) to compare the distribution of metadata to a Zipfian distribution. Where the Zipfian distribution is a poor fit, we will evaluate other distributions as well, in order to develop an accurate picture of metadata distributions. Where metadata can be segmented (such as text fields), we will examine the distribution of the field both segmented and unsegmented. In the unsegmented analysis, a field value such as "Protein Folding" will be treated as a single value, whereas in segmented analysis, the frequencies will be updated for both "Protein" and "Folding". We will also examine the vocabulary size of each field, segmented and unsegmented, to evaluate how closely each field's distribution follows the predictions of Heaps' Law. To our knowledge, this is the first deep statistical analysis of this type of rich metadata. As well as benefiting research in file naming, this research can be used to inform the design of efficient indexes for searchable file systems. It can also be used for more sophisticated generation of synthetic file system and indexing workloads, extending the abilities of systems such as Impressions [3].

2.2.2 User Studies and Evaluation

File naming is a highly individual behavior, and manually generated file names are, without doubt, often superior to automatic ones. However, the purpose of this research is to offer a viable alternative to manual file naming that is less labor-intensive and error-prone. Therefore, the goal of evaluation is to confirm that the generated file names are meaningful to the user and uniquely identify the contents of the file. This can be evaluated with a simple series of yes/no judgements on the part of the user. As a control, we can also supply a file name generated by randomly choosing file attributes. Therefore, an evaluation will consist of some set of users, presented with a series of file names, and the ability to inspect the original file and its manually generated file name (if available.) The user will be presented with a generated file name, either randomly chosen or Zipfian, and asked if it is sufficient to identify the contents of the file. After answering yes or no, the user will be presented with a second file name, generated from the opposite algorithm, and asked if this name is also sufficient to identify the file. After answering yes or no, the user is presented with the next file and file name pair. Experiments will be done in short runs of ten to twenty files, to prevent decision fatigue. Users will be matched to files which are either in their field of expertise, or of general interest, such as MP3s.

2.3 Summary

In summary, we propose to improve file management in SciHEC systems by using properties of metadata to quickly find and manage files. We will focus in particular on ranking files and naming files. We propose two ranking algorithms, one which employs provenance to determine what files users find interesting, and one which uses light-weight access counters. We also propose a solution to security threats in ranked search, which hides usage information the user should not have access to. In addition, we propose a file naming algorithm which uses the statistical properties of metadata to discover attributes which are human-like in content.

In order to evaluate our proposed algorithms, we will collect real-world usage data on SciHEC file systems, and perform statistical analysis on provenance graphs, access counters, and rich metadata. We will use the results of this analysis to perform user studies on ranking and naming. And finally, we will use the collected data to characterize the expected performance and security properties of our ranking algorithms.

Chapter 3

Related Work

If you don't see the use of it, I certainly won't let you clear it away. Go away and think. Then, when you can come back and tell me that you do see the use of it, I may allow you to destroy it.

GK Chesterton

While ranking specifically for SciHEC file systems has seen little research, much work has been done on ranking for other types of search. Strategies for learning to rank, evaluating ranking, securing ranked search, and doing ranking efficiently will all be used to inform our work. In addition, research on faceted search and web snippets may have some applicability to the file naming problem.

3.1 Desktop search and desktop ranking

The problem of effective ranking for desktop search has been a subject of interest for some time now. Researchers have identified the lack of connections between files, and proposed to infer links, based on file contents, directory structures, or browser caches [10, 14]. While these do add additional context for search, these approaches are computing intensive, as every file's contents must be parsed. They require manually encoding types of file relationships, such as that of header files to source files. They work poorly for binary data and proprietary formats, requiring a different parser for each type of file. Perhaps for these reasons, these techniques have not seen adoption, or even much in the way of evaluation.

Commonly used search tools, such as Spotlight [8], often rank based on the access time of the file, putting most recently accessed files first. While this is a convenient and generic approach, it does not take into account the frequency of usage. In this model, one access today is worth more than twenty yesterday. By contrast, we focus on the frequency of accesses, while still accounting for the idea of “working sets” and the importance of recent files.

More recently, the idea of tracking user activity as a way of improving search has gained some traction. Gaugaz *et al.* proposed a technique to temporally correlate file accesses [18], which is similar to provenance, but not as precise. Soules proposed the idea of using activity tracking to suggest additional search results by correlating files [46]. Dumais *et al.* suggested using visual user activity cues such as the file’s time and author [15] to help the user quickly identify relevant search results. Lifestreams [16] suggests presenting a single time organized stream of data in order to provide visual and temporal context to the user. And Shah describes using provenance to aid in file system search [45]. All of these utilize user activity to improve the *recall* of queries, by drawing in additional search results. By contrast, our ranked search algorithms assume that the search recall is sufficient, and focus on improving the *precision* of the presented results. Popularity-ranked search would be novel, even in a desktop environment.

Previous work has been done on provenance ranking [36], ProvRank and SubRank. These two algorithms are designed to determine importance within a provenance graph, and use the stationary distribution of a Markov chain, similar to TaPIR. However, these algorithms both focus on determining ranking in the context of provenance queries. Both are designed to prune provenance in the face of underspecified provenance queries. ProvRank and SubRank determine the rank of a provenance subgraph, in order to prune ubiquitous ancestor trees from the graph (such as the kernel compilation sub-tree, on which every file must depend). This approach is complementary to our approach, and in fact, can be used in conjunction with it. Pruning the provenance tree of extremely old nodes can reduce the computational cost of TaPIR, in conjunction with rank smoothing to guarantee that every node has a value.

3.2 Ranking for the web

Ranking for the web has moved far beyond the days of Okapi BM25 [29, 30] and other unadorned term frequency/inverse document frequency (*tf/idf*) similarity metrics. PageRank [41] is commonly considered the gold standard for general purpose web ranking. PageRank, much like our proposed ranking scheme, relies on the link structure of the web. However, it makes assumptions about the structure of the link graph which are not appropriate in a provenance context. In addition, because the search engine does not have direct insight into the usage of individual websites, it has to rely on links as a surrogate measure of importance. Since we have direct knowledge of what the user accesses, we can take advantage of the more detailed information.

Recent ranking algorithms have looked at online learning for ranking [28], relying on metrics such as click through data to determine, over time, whether the results presented are helpful and well ranked. In a web context, click through data is a surrogate statistic for usage data, and is similar in spirit to AccessRank. These techniques can improve on an existing ranking scheme and might also be useful in a file system context, where click through data can offer feedback on ranking over time, but do not replace the need for an initial ranking of search results.

3.3 Predictive caching

There are similarities of function between ranking and file-level predictive caching, and there may be some overlap in algorithms. Both predict what people are likely to access, but predictive caching focuses on current file system activity as a context for prediction, whereas ranked search uses the query as context, and maintains an overall list of predictions. Algorithms for ranking and caching may be able to complement one another, allowing the system to arrive at a better picture of the user's current activity and needs in order to pre-fetch files and choose relevant search results.

Algorithms for predictive caching often rely on a successor model [21, 33], in which the most recent files are used to pre-fetch files which were previously opened after the currently active file. Amer's work on adjustable accuracy predictors using a successor model [5] is similar in spirit to our time decayed access counter ranking. However, Amer's work uses a fixed size list, whereas we maintain a rank for every file, and allow files which were extremely popular in the past to maintain a small positive rank indefinitely.

3.4 File system search at scale

Large scale file system search is a known problem. However, existing solutions have focused on the problem of indexing data, not ranking it. Most solutions assume a SQL-like query interface exists, and then focus on indexing, early discard, and so on.

Spyglass [35] focused on how to create indexes with fast update and query time, by using a log and merge technique for index building, multiple indexes with Bloom filters to facilitate early discard of irrelevant areas of the file system, and K-D trees to take advantage of the skewed distributions of metadata. SmartStore [24], rather than using K-D trees, used R-trees and clustering, in an attempt to make index rebalancing more efficient in the face of frequent updates.

Diamond [25] is a system for large scale search, focused on efficient search where not all searchable attributes can be kept in indexes, and some data must be extracted from files in real time. They describe a system for early discard of files based on filters, using distributed computation to process files and evaluate whether they match a criteria. The techniques described in Diamond are still useful even in a well indexed system such as we describe, but do not fill the same niche as a ranking algorithm, since they do not improve search precision.

LiFS [7] focuses on the problem of creating links between files as well. However, in LiFS, unlike TaPIR, links are created either manually, by the user, or are driven by file content, requiring computationally expensive transducers to be run over every file. LiFS considers the possibility of ranking using links, but does not provide a mechanism for doing so, and focuses on the applicability of application specific link types, much like Chirita [14] and Bhagwat [10] do in their desktop search research.

3.5 Search security

Most research on search security has focused on full text search and corporate intranets, where the search system must deal with a wide variety of heterogeneous security models and systems, and late binding security checks. Research such as Bailey, *et al.* [9] suggests that this heterogeneity comes at a high cost, and that if exact result counts are needed, query processing time will scale linearly with the size of the entire result set, not just the portion the user can see.

Büttcher [13] proposed a similar late binding model for full text file system search, and also demonstrated that doing ranked full text search could be used for statistical attacks which revealed information about the contents of files outside the user's security scope.

In our own work [42], we performed empirical studies of file system security permissions, and demonstrated that partitioning the file system indexes along security lines, such that users could either see every file or no file within a partition, not only prevented this statistical leakage, but allowed the system to perform late-binding security checks at a partition level, and could generate search partitions with good properties for searching conventional POSIX metadata. However, our focus was primarily on boolean search, and did not cover novel ranking models for the file system such as the one we are proposing. In our current work, we extend the scope to focus on less conventional metadata, such as provenance and scientific metadata. Our intent is to demonstrate that a partitioned model for provenance graphs will also create search partitions which are efficient, and allow ranking to be done at a local level if necessary.

Braun *et al.* describe models for provenance security [12], and perform user studies to describe the types of provenance security required in different environments, from scientific computing to medicine and administrative contexts. The solution we describe to usage security for provenance requires *nil* knowledge of links under the categories they describe.

3.6 Naming and disambiguating

File naming can be thought of as analogous to the problem of disambiguating search results on the web. Web search, like file system search, has a huge number of files, many with the same name, and when returning results, the search engine must help the user choose between them. On the web, the historical assumption is that the user is retrieving textual information, and the common approach is to reveal a snippet from the page containing the search terms in context. Newer search types, such as video, rely on a title and key frame. However, in the file system arena, particularly scientific files, approaches aimed at text and video are flawed. Many files are in opaque data formats, and no snippet is available. Search may be selecting files based on metadata rather than content, in which case the user already knows the search term's context. The search term may help guide the choice of name, but excerpts from the data will not be sufficient.

The problems of naming directories has been a subject of interest for some time. Research in semantic file systems [19, 20, 40] has required that the system present directory names based on the metadata of files within it. For instance, the original Semantic File System (SFS) [19] treats all directory names as queries. If the user enters a query containing an unbound field name (such as `user:`), SFS will return `/jones, /root, /smith` and so on as subdirectories. However, no consideration is given to ordering or pruning of these subdirectories, so a query simply returns all possible values in alphabetical order. The Logic File System (LISFS) [40] establishes a *taxonomy* of attributes, such that some attributes subsume others. If the results to a query contain one or more attributes which are subsumed, only the higher level attribute will be displayed as a directory name, and only attributes which distinguish between the query results are shown.

The semantic file systems just mentioned have been applying techniques from faceted search, a navigation model often found on the web for navigating databases and other well-structured data sets. ViewFS [32] explicitly focuses on file system search as a faceted search problem. ViewFS is also a semantic file system, offering virtual directories as query results. They propose using user feedback to refine the order of facet presentation over time. Research on faceted search such as Zhang and Zhang [50] and van Zwol *et al.* [49] has focused on learning facet ordering, based on relevance feedback via click-through rates. While these techniques work well on the web, where large number of users are expected to visit a site, they require sufficient user feedback for learning, and perform poorly when little or no data is yet available. Van

Zwol does not offer a way to bootstrap initial facet presentation. Zhang and Zhang suggest using overall facet/value term frequency in the top n ranked documents, normalized by inverse document frequency (in other words, tf/idf), as a method of cold starting the facet selection. By contrast, we focus on finding attributes with a Zipfian distribution, which appears to closely match to human-like data patterns. We will also compare our approach to a tf/idf approach.

Our ranking and naming algorithms are agnostic to the directory view used, and could be applied in any semantic file system. The problem of appropriately naming files is similar to the problem of presenting useful facets to the user, and the metadata analysis we perform could be used as a way of selecting facets. Our research offers an *a priori* model for choosing attributes to present, which requires no user intervention aside from creating metadata. Relevance feedback techniques such as proposed above could then be used to improve ordering over time, if desired.

Having described the work others have done in this area, in the next section, we focus on our own previous research and discuss our preparation for our proposed research.

Chapter 4

Preliminary Work

A thick tree grows from a tiny seed. A tall building arises from a mound of earth. A journey of a thousand miles starts with one step.

Lao-tzu

While much remains to be done, we have already done some exploratory research in areas related to search, ranking, security, and naming. In this section, we will describe our previous work in the design of semantic, searchable file systems, in the exploration of metadata and user data organization, and on search security. And finally, we briefly cover our current implementation status.

4.1 Ranking

An initial version of the TaPIR algorithm has been written and run on sample provenance graphs. Implementation of a version which uses an existing library for reading and parsing provenance files is under way. Our earlier work on partitioning file system trees is being re-written and leveraged to work on provenance graphs, also using the existing provenance parser, in order to do the necessary security analysis.

We previously assisted with the design of Copernicus [34], a parallel non-hierarchical file system which was designed from the ground up to be a scalable semantic file system, where all directories were virtual, and all paths were seamlessly translated into metadata queries. Copernicus used a graph clustering algorithm to lay out metadata and attribute clusters for easier searching, which was designed to replace the traditional directory based layout.

We recently described the design of a search system designed specifically for SciHEC

[31], which collects provenance, provides a unified ranked search space over disparate file systems and archival storage using *transient provenance*—provenance which records when and where a file is migrated out of the file system—and provides mechanisms for users to annotate and query their files using metadata, content, and provenance. Our research will use a simulated version of this file system in order to provide users with a query interface for user studies.

In our earlier work on Security Aware Partitioning [42], we characterized properties of partitioned indexes that will lead to efficient searches. In Security Aware Partitioning, the file system indexes are are *partitioned* into sub-indexes. This approach has been used in the past for speeding up search [35], by using in-memory Bloom filters to determine relevant partitions, without incurring the cost of seeking to disk to retrieve a full index. However, in Security Aware Partitioning, the partitions are chosen in order to meet security criteria. A partition is created such that a user can either see every file in a partition, or no file, and each partition has a known set of users and groups which has access to it. This means that security checks can be done at the partition level, before ever retrieving results. Each partition can maintain a rank sorted index without revealing information about other partitions and their files. Thus, Security Aware Partitioning prevents content-based statistical attacks on ranked search, while still allowing ranking to be calculated globally.

In a system with partitioned indexes and early discard, indexes with high information gain and low entropy offer the most bang for the buck, because relevant results will be concentrated in a few indexes, allowing fewer index retrievals from disk, and permitting more effective caching for hot queries and indexes. We have previously shown that using security as a method for index partitioning can create indexes with low entropy and high information gain, leading to efficient searches [42]. We will use these criteria to evaluate the effectiveness of partitioning the provenance graph along security lines as well.

Our previous work in digital forensics [17] has also looked at correlating security permissions and metadata on multi-user desktop systems. This work investigated building machine learning predictors to determine file ownership, using the file system and embedded metadata as features for the machine learning algorithm. This work demonstrated that at least on desktop systems, time was an excellent predictor of file ownership. If this holds for larger systems, this suggests that temporally correlated parts of the provenance graph are likely to fall into the same security partition, which would lead to more efficient provenance partitions.

4.2 Naming

We have previously done unpublished preliminary analysis of file metadata on the Real Data Corpus [1]. We examined the distribution of file metadata and content within the file system hierarchy, examining the degree of locality within the directory hierarchy each displayed, to determine how users organized their files. This analysis focused on desktop file usage, rather than scientific files, and did not analyze the statistical distribution of individual fields. However, it did confirm that a number of common file types present on real world file systems have well structured metadata fields which are populated, suitable for more in depth statistical analysis and use in naming experiments. It also showed that users are not good at either organizing their files, or populating rich metadata manually, confirming that manual data management leads to poor searchability in the file system.

We have run an early version of the algorithm on an MP3 collection of the author's, which uses the entire metadata field to calculate vocabulary length and distribution, and does not check for independence of fields. When run on an MP3 collection, this yields names such as the following:

'Londonbeat-BravoHits92-2008-06-18T04:41:30Z-'
'B.B. King-The Ultimate Collection-2008-06-18T04:41:11Z-'
'Siouxsie & The Banshees-Superstition-2008-06-18T06:23:56Z-'

In this case, the algorithm has determined that 'Artist', 'Album', and 'Date Added' are the most Zipfian of the fields. However, since the date added is likely to correlate strongly with the album, reducing correlation, as we plan to do in future versions of the algorithm, would likely remove this field from the top three, replacing it with one of the next most Zipfian fields, such as 'Composer' (which will likely be correlated with Artist) or 'Sort Name'. Most humans would likely find a file name of artist-album-name to be a very good unique identifier of an MP3.

4.3 Summary

Our previous work in searchable file systems and file system analysis has demonstrated a need for high quality data management. Manual data management has proven to be error prone, and users often do not manage their data at all if manual intervention is required. Our previous analysis has shown that using security access rights to partition indexes can lead to high quality indexes, and suggests that the security rights of files is temporally correlated, which gives us hope that provenance partitions based on security will be effectively sized. We have an existing file system design for SciHEC, and intend to prove the general applicability of our algorithms within the context of this design. In the next section, we describe our expected timeline for performing the remaining research.

Chapter 5

Plan and Timeline

The best-laid schemes o' mice an' men, gang aft agley

Robert Burns

This section contains a list of tasks needed to complete the proposed work, along with a suggested timeline and a discussion of risks and dependencies. In addition, we discuss the expected scope of research, and areas we do not intend to cover in the pre-dissertation time frame.

5.1 Worklist

This is a list of high level tasks necessary to complete this research, organized by topic area. Each of these is also included in the timeline in Section 5. Each task is labeled with the expected quarter (including summer quarters), starting from advancement, and a reference to the proposed work section where it is detailed.

5.1.1 Ranking

- A number of provenance and file usage traces collected on real world SciHEC systems using a user-space file system (Q4-Q5, Section 2.1.4)
- Two proposed ranking algorithms, using provenance and decayed access counters (completed, Section 2.1.1, 2.1.2)
- Proof of concept implementations of both ranking algorithms (Q3, Section 2.1.1, 2.1.2)
- A user study in which we compare the two ranking algorithms and two baseline algorithms, PageRank and a "newest first" algorithm (Q6, , Section 2.1.5)

Security for ranking

- An analysis of file ownership as it relates to the collected provenance graphs (Q5, Section 2.1.3.2)
- An analysis of the security properties of provenance based ranking (Q5, Section 2.1.3.2)
- A proposal for partitioning provenance graph indexes according to security permissions (completed, Section 2.1.3)
- An analysis of the tradeoffs of partitioned provenance graphs for ranking (Q5, Section 2.1.3.2)

5.1.2 Naming

- An analysis of the statistical distributions of rich metadata on a file type by file type basis, focusing on scientific data file types (Q1, Section 2.2.1)
- An proposed algorithm for producing unique, meaningful names for files (completed, Section 2.2)
- A user study in which we evaluate the naming algorithm on a range of corpuses (Q2, Section 2.2.2)

5.1.3 Publications

- Publish metadata analysis (Q2)—submit to MASCOTS
- Publish file naming user study (Q4)—submit to USENIX ATC
- Publish study on security of partition ranked search (Q6)—venue TBD
- Publish study on ranked search (Q7)—venue TBD
- Complete dissertation (Q7)

5.2 Expected Timeline

This section includes all tasks listed in the worklist. In addition, it lays out an expected timeline for various administrative tasks necessary to complete the worklist. Quarters are numbered relative to the time of advancement, and an absolute date is also included. The expected date of completion is Fall 2013.

5.2.1 Q1 - Spring 2012

Arrange for data collection sites. Determine storage architecture at data collection sites, and fill out appropriate paperwork. Select sources of scientific data for preliminary naming studies. Begin metadata analysis of scientific data. Get IRB approval for naming studies. Solicit users for preliminary naming studies and for data collection.

5.2.2 Q2 - Summer 2012

Begin implementation of provenance and usage data collection. Implement proof of concept implementations of both ranking algorithms, based on expected data format from collected data. Begin writing paper based on results of metadata analysis. Begin user study on naming.

5.2.3 Q3 - Fall 2012

Conclude user study on naming. Begin writing paper based on results of user study. Implement analysis for partitioned provenance graphs.

5.2.4 Q4 - Winter 2013

Begin soliciting additional users for ranking studies. Complete implementation of data collection if not already done. Begin collecting provenance data at sites. Implement database mockup of searchable file system. Begin writing dissertation.

5.2.5 Q5 - Spring 2013

Conclude data collection. Analyze security properties of provenance graph. Partition and analyze properties of partitioned graph.

5.2.6 Q6 - Summer 2013

Begin user study on ranked search. Write paper on security of ranked search.

5.2.7 Q7 - Fall 2013

Conclude user study on ranked search. Write paper on results of ranked search study. Complete dissertation.

5.3 Risks and Dependencies

The greatest risks to on-time completion are in the area of data collection. Since the sites have an as-yet unknown environment, both in terms of computing and politics, the difficulty of getting the necessary data will vary wildly. Implementation may be simple or complex, and there may be bureaucratic obstacles to data collection. Delays in data collection will result in all work related to ranking, including security, slipping schedule.

The second risk is in the area of user studies. Finding users who are both capable and qualified to evaluate the ranking and naming will be challenging. Naming is lower risk, since we will attempt to use files which are freely accessible and have a wide user base wherever possible, as well as including files of a non-technical nature, which will allow us to solicit users "off the street" if necessary. Ranking requires access to the file system where the data was collected, and therefore the potential user pool is small. A delay in soliciting users will push out user studies, but not the analysis papers. If sufficient users are simply not available for statistically significant results, then the resulting papers will be weak.

5.4 Exclusions

While there is always room for more research, a dissertation is a finite body of work. To avoid confusion about the scope of work, we include a list of topics which are related to the area of research, and might plausibly be part of this dissertation, but we do not intend to study. We feel that these are already either well understood topics, or that the state of the art is sufficient to support the core goals of this research.

This dissertation proposal does not anticipate the author doing novel research in the following areas:

- Techniques for building indexes over provenance, metadata, or content within a file system. This work assumes that a searchable file system is available, although the evaluation may require simulating or building an index. Other members of our group are doing complementary research in this area, which we intend to leverage.

- Techniques for extracting metadata from files. This area has already been well researched, and is sufficient for the current work.

- Comparison strategies for ranking algorithms. This is a well explored area, and is outside of the core focus of the research.

- File system tracing techniques. This is necessary for our research, but we believe there is sufficient research to support the data collection we will do.

- Efficient mechanisms for time decayed counters. This is a very well explored area, and is outside of the core focus of the research.
- Manipulation of ranking in order to steer users to a particular file. We feel this threat model is unlikely to be relevant in the file system context.

In addition, in the interests of time, we expect to save the following areas for future work:

- An evaluation of personalized provenance ranking
- Social network analysis via provenance

Chapter 6

Conclusions

The future is already here—it's just not very evenly distributed.

William Gibson

High end scientific computing is on the cutting edge of science and computing technology, and yet it is needlessly trapped in the past when it comes to file management. Three-ring binders, manual file organization and naming, and single threaded search tools are inefficient for both the system and the user, and do not scale to modern data volumes. Without robust file management, we are all adrift on a sea of data. Ranked search will allow file systems to scale better, by reducing the number of results retrieved, and allowing the user to quickly refine searches. Automatic file naming allows the user to spend less time managing their files, and reduces file portability problems due to long file names.

Our proposed work will be contributing the following research to the field of file management. File naming is an uncharted area, and ours is the first research to focus on file naming as such, although there has been previous work in related fields such as faceted search and web snippets. In addition, ours will be the first application of provenance data in the area of file system ranking, and the first to focus on ranking for SciHEC file systems. We will be demonstrating the feasibility of provenance based ranking and ranking using decayed usage counters, as well as providing a basis for comparison against other file system ranking algorithms. The security properties of ranking are a known area of concern, and an evaluation of the effectiveness of secured search, using our provenance partitioning algorithm, will allow comparison of the tradeoffs between globally calculated ranking and secure partitioned ranking.

Our proposed work is designed to make sense of the file system, managing data automatically and transparently, and freeing the file system from the vagaries of manually managed and searched files. Our algorithms will be to be secure, scalable, and easy to use and understand. File systems should work to the benefit of both the user and the computing system, and our research will advance that goal.

Bibliography

[1] The Real Data Corpus. http://digitalcorpora.org/corpora/disk-images/rdc-faq.

[2] E. Adar, J. Teevan, and S. T. Dumais. Large scale analysis of web revisitation patterns. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08, pages 1197–1206, New York, NY, USA, 2008. ACM.

[3] N. Agrawal, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. Generating realistic *impressions* for file-system benchmarking. In *Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST)*, pages 125–138, Feb. 2009.

[4] A. Amer, D. D. E. Long, and R. C. Burns. Group-based management of distributed file caches. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS '02)*, Vienna, Austria, July 2002. IEEE.

[5] A. Amer, D. D. E. Long, J.-F. Pâris, and R. C. Burns. File access prediction with adjustable accuracy. In *Proceedings of the 21st IEEE International Performance Conference on Computers and Cmmunication (IPCCC '02)*, Phoenix, Apr. 2002. IEEE.

[6] A. Ames, N. Bobb, S. A. Brandt, A. Hiatt, C. Maltzahn, E. L. Miller, A. Neeman, and D. Tuteja. Richer file system metadata using links and attributes. In *Proceedings of the 22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies*, Monterey, CA, Apr. 2005.

[7] S. Ames, N. Bobb, K. M. Greenan, O. S. Hofmann, M. W. Storer, C. Maltzahn, E. L. Miller, and S. A. Brandt. LiFS: An attribute-rich file system for storage class memories. In *Proceedings of the 23rd IEEE / 14th NASA Goddard Conference on Mass Storage Systems and Technologies*, College Park, MD, May 2006. IEEE.

[8] Apple Developer Connection. Working with Spotlight. http://developer.apple.com/macosx/tiger/spotlight.html, 2004.

[9] P. Bailey, D. Hawking, and B. Matson. Secure search in enterprise webs: Tradeoffs in efficient implementation for document level security. In *Proceedings of the 2006 International Conference on Information and Knowledge Management Systems (CIKM '06)*, Nov. 2006.

[10] D. Bhagwat and N. Polyzotis. Searching a file system using inferred semantic links. In *Proceedings of the Sixteenth ACM Conference on Hypertext and Hypermedia*, HYPERTEXT '05, pages 85–87, New York, NY, USA, 2005. ACM.

[11] S. Brandt, C. Maltzahn, N. Polyzotis, and W.-C. Tan. Fusing data management services with file systems. In *Proceedings of the 4th Petascale Data Storage Workshop (PDSW '09)*, Nov. 2009.

[12] U. Braun and A. Shannar. A security model for provenance. Technical report, Harvard University, 2006.

[13] S. Büttcher and C. L. A. Clarke. A security model for full-text file system search in multi-user environments. In *Proceedings of the 4th USENIX Conference on File and Storage Technologies (FAST)*, pages 169–182, San Francisco, CA, Dec. 2005.

[14] P. A. Chirita, R. Gavriloaie, S. Ghita, W. Nejdl, and R. Paiu. Activity based metadata for semantic desktop search. *Lecture Notes in Computer Science : The Semantic Web: Research and Applications*, pages 439–454, 2005.

[15] S. Dumais, E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. Stuff I've seen: a system for personal information retrieval and re-use. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 72–79. ACM Press, 2003.

[16] S. Fertig, E. Freeman, and D. Gelernter. Lifestreams: an alternative to the desktop metaphor. In *CHI '96: Conference Companion on Human Hactors in Computing Systems*, pages 410–411, 1996.

[17] S. L. Garfinkel, A. Parker-Wood, D. Huynh, and J. J. Migletz. An Automated Solution to the Multiuser Carved Data Ascription Problem. *IEEE Transactions on Information Forensics and Security*, pages 868–882, 2010.

[18] J. Gaugaz, S. Costache, P.-A. Chirita, C. Firan, and W. Nejdl. Activity based links as a ranking factor in semantic desktop search. In *Proceedings of the 2008 Latin American Web Conference*, pages 49–57, Washington, DC, USA, 2008. IEEE Computer Society.

[19] D. K. Gifford, P. Jouvelot, M. A. Sheldon, and J. W. O'Toole, Jr. Semantic file systems. In *Proceedings of the 13th ACM Symposium on Operating Systems Principles (SOSP '91)*, pages 16–25. ACM, Oct. 1991.

[20] B. Gopal and U. Manber. Integrating content-based access mechanisms with hierarchical file systems. In *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation (OSDI)*, pages 265–278, Feb. 1999.

[21] J. Griffioen and R. Appleton. Reducing file system latency using a predictive approach. In *Proceedings of the Summer 1994 USENIX Technical Conference*, pages 197–207, 1994.

[22] P. Guo. Burrito: Rethinking the electronic lab notebook. http://www.stanford.edu/ pgbovine/burrito.html.

[23] H. Heaps. *Information retrieval, computational and theoretical aspects*. Library and Information Science. Academic Press, 1978.

[24] Y. Hua, H. Jiang, Y. Zhu, D. Feng, and L. Tian. SmartStore: A new metadata organization paradigm with semantic-awareness for next-generation file systems. In *Proceedings of SC09*, Nov. 2009.

[25] L. Huston, R. Sukthankar, R. Wickremesinghe, M. Satyanarayanan, G. R. Ganger, E. Riedel, and A. Ailamaki. Diamond: A storage architecture for early discard in interactive search. In *Proceedings of the 3rd USENIX Conference on File and Storage Technologies (FAST '04)*, pages 73–86, San Francisco, CA, Apr. 2004. USENIX.

[26] B. J. Jansen, D. L. Booth, and A. Spink. Determining the informational, navigational, and transactional intent of web queries. *Inf. Process. Manage.*, 44:1251–1266, May 2008.

[27] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20:422–446, October 2002.

[28] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 133–142, New York, NY, USA, 2002. ACM.

[29] K. S. Jones, S. Walker, and S. Robertson. A probabilistic model of information retrieval: development and comparative experiments: Part 1. *Information Processing & Management*, 36(6):779 – 808, 2000.

[30] K. S. Jones, S. Walker, and S. Robertson. A probabilistic model of information retrieval: development and comparative experiments: Part 2. *Information Processing & Management*, 36(6):809 – 840, 2000.

[31] S. Jones, C. Strong, A. Parker-Wood, A. Holloway, and D. D. E. Long. Easing the Burdens of HPC File Management. In *Proceedings of the 6th Petascale Data Storage Workshop (PDSW '11)*, Nov. 2011.

[32] J. Koren, Y. Zhang, S. Ames, A. Leung, C. Maltzahn, and E. L. Miller. Searching and navigating petabyte scale file systems based on facets. In *Proceedings of the 2007 ACM Petascale Data Storage Workshop (PDSW '07)*, Reno, NV, November 2007.

[33] T. M. Kroeger and D. D. E. Long. Design and implementation of a predictive file prefetching algorithm. In *Proceedings of the 2001 USENIX Annual Technical Conference*, pages 105–118, Boston, Jan. 2001.

[34] A. Leung, A. Parker-Wood, and E. L. Miller. Copernicus: A scalable, high-performance semantic file system. Technical Report UCSC-SSRC-09-06, University of California, Santa Cruz, Oct. 2009.

[35] A. Leung, M. Shao, T. Bisson, S. Pasupathy, and E. L. Miller. Spyglass: Fast, scalable metadata search for large-scale storage systems. In *Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST)*, pages 153–166, Feb. 2009.

[36] D. Margo, P. Macko, and M. Seltzer. Addressing underspecified lineage queries on provenance. Technical report, Harvard University, 2012.

[37] N. Megiddo and D. S. Modha. ARC: A self-tuning, low overhead replacement cache. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST '03)*, pages 115–130, San Francisco, CA, Mar. 2003. USENIX Association.

[38] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, B. Plale, Y. Simmhan, E. Stephan, and J. a. Van den Bussche. The open provenance model core specification (v1.1), 07 2010.

[39] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. Seltzer. Provenance-aware storage systems. In *Proceedings of the 2006 USENIX Annual Technical Conference*, Boston, MA, 2006.

[40] Y. Padioleau and O. Ridoux. A logic file system. In *Proceedings of the 2003 USENIX Annual Technical Conference*, pages 99–112, San Antonio, TX, June 2003.

[41] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford, Nov. 1998.

[42] A. Parker-Wood, C. Strong, E. L. Miller, and D. D. E. Long. Security aware partitioning for efficient file system search. In *Proceedings of the 26th IEEE Conference on Mass Storage Systems and Technologies*, May 2010.

[43] S. V. Patil, G. A. Gibson, S. Lang, and M. Polte. GIGA+: scalable directories for shared file systems. In *Proceedings of PDSW '07*, 2007.

[44] C. Sar and P. Cao. Lineage file system. http://theory.stanford.edu/ cao/lineage-fs.ps, Jan. 2005.

[45] S. Shah, C. A. N. Soules, G. R. Ganger, and B. D. Noble. Using provenance to aid in personal file search. In *Proceedings of the 2007 USENIX Annual Technical Conference*, pages 171–184, June 2007.

[46] C. A. N. Soules and G. R. Ganger. Connections: using context to enhance file search. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles (SOSP '05)*, pages 119–132, New York, NY, USA, 2005. ACM Press.

[47] C. Strong, S. Jones, A. Parker-Wood, A. Holloway, and D. D. E. Long. Los Alamos National Laboratory Interviews. Technical Report UCSC-SSRC-11-06, University of California, Santa Cruz, Sept. 2011.

[48] S. K. Tyler and J. Teevan. Large scale query log analysis of re-finding. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 191–200, New York, NY, USA, 2010. ACM.

[49] R. van Zwol, L. Garcia Pueyo, M. Muralidharan, and B. Sigurbjörnsson. Machine learned ranking of entity facets. In *Proceedings of the 33rd international ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 879–880, New York, NY, USA, 2010. ACM.

[50] L. Zhang and Y. Zhang. Interactive retrieval based on faceted feedback. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 363–370, New York, NY, USA, 2010. ACM.

[51] G. K. Zipf. *Human Behaviour and the Principle of Least Effort.* Addison-Wesley, 1949.