Examining Extended and Scientific Metadata for Scalable Index Designs

Aleatha Parker-Wood
Storage Systems Research Center, University of California, Santa Cruz
aleatha@cs.ucsc.edu

Brian A. Madden
Storage Systems Research Center, University of California, Santa Cruz
madden@cs.ucsc.edu

Michael McThrow
Storage Systems Research Center, University of California, Santa Cruz
mmcthrow@cs.ucsc.edu

Darrell D. E. Long
Storage Systems Research Center, University of California, Santa Cruz
darrell@cs.ucsc.edu

Ian F. Adams
Storage Systems Research Center, University of California, Santa Cruz
iadams@cs.ucsc.edu

Avani Wildani
Storage Systems Research Center, University of California, Santa Cruz
avani@cs.ucsc.edu

ABSTRACT

While file system metadata is well characterized by a variety of workload studies, scientific metadata is much less well understood. We characterize scientific metadata, in order to better understand the implications for index design. Based on our findings, existing solutions for either file system or scientific search will not suffice for indexing a large scientific file system.We describe the problems with existing solutions, and suggest column stores as an alternative approach.

Categories and Subject Descriptors

E.5 [**Files**]: [Sorting/searching]; H.3.2 [**Information Storage and Retrieval**]: Information Storage—*File organization*

General Terms

Design, Measurement,Performance

Keywords

File systems, Index design, Metadata, Search, Scientific data

1. INTRODUCTION

In this paper, we examine the problem of searching scientific data on HPC systems. While there are many workload studies for file system metadata [14, 9, 8, 20, 13, 28], they have focused on POSIX metadata. Search systems based on them [19, 17, 27, 21] attempt to extrapolate performance for other use cases. By contrast, we examine scientific metadata directly, in order to better understand the design space of scientific metadata and content indexing systems. Having examined scientific data, we consider existing indexing solutions for both file system search and scientific search, and conclude that they are not well suited for HPC search.

Consider scientists working together on a shared HPC computing and storage system. Scientists may want to search files by size or age, but are more interested in searching metadata about content [25]. An oceanographer might want files containing observations at a certain salinity, and a biologist might need files about a species or watershed area. Each of these searches is a metadata search, but rather than relying on system generated metadata, it relies on metadata that is domain specific, and may be embedded in content or stored in an extended attribute.

We find that in contrast to POSIX metadata, scientific metadata is heterogenous, with different metadata for different disciplines. It is sparse, with many missing values even within files from a single discipline. It is high-dimensional, and those dimensions are a mix of numeric, textual, and categorical data. It can also be non-atomic, with many values for a single field. We also find that it has a number of features which an indexing strategy should take advantage of, such as compressibility.

Search for scientific metadata poses a number of unique problems. Scientific metadata is large, and can easily outstrip the size of the data it describes. In many cases file content and metadata are indistinguishable. For instance, in astronomy a dataset may contain raw data such as pixel brightness, and metadata such as a catalog identifier. What matters is whether the data will be queried. In this work, we focus on search for data in *fields*, a single dimension of metadata such as temperature or author, as opposed to free text with a document, since free-text search at scale is a well explored problem.

File system search, especially at scale, poses additional problems. Results to a file system query are in the form of a list of matching files. This is different from many database indexing problems, where results are records, and require eventually performing a join to retrieve every field. Finally, HPC file systems are often operating near full capacity, and cannot use a large fraction of storage to store indexes, or CPU for brute force search.

A search system for scientific metadata must balance each of these constraints. It must make it possible to query a large

SYSTOR '13, June 30 - July 02 2013, Haifa, Israel

ACM 978-1-4503-2116-7/13/06 ...$15.00.

	Discipline	Native format	Record count	Subsampled?	Sample count	Total size	Fields
Dryad	Biology	XML	31K	No	31K	400MB	44
WISE	Astronomy	CSV	564M	Yes	10K	1TB	285
ARGO[1]	Oceanography	NetCDF	2B	Yes	634,880	330GB	108
ORNL	Climatology	CSV	1478	No	1478	154KB	14

Table 1: Data sets. [1] Full size is extrapolated based on recent file sizes and duration of data collection. Actual size has not been published.

body of sparse, high-dimensional data without consuming a large fraction of CPU, I/O bandwidth, or storage. It should be able to add new files to indexes efficiently. Finally, it must efficiently handle a mix of data types. We examine a variety of indexes, and conclude that column stores are the best match for these requirements.

2. EXPERIMENTAL DESIGN

In this section, we define some terminology, and describe the index types we examine, as well as the data sets which we analyzed.

2.1 Terminology

Since different index structures organize data differently, it can be hard to find a common terminology to discuss them. In addition, metadata can refer to different things when used in a scientific context, rather than a file system context. We will discuss *fields*, a single dimension of metadata such as temperature or author. *Metadata* refers to data in fields, both extended metadata and metadata embedded in file contents. We refer to a single data object, typically a file and its metadata, as an *item*. And we use the term *data element* to refer to data in a single field of a single item. A field of an item may have multiple data elements, as we will discuss later.

2.2 Index Types

In this paper we consider a number of index types for file system search and scientific indexing in the light of our findings about scientific metadata.

Relational databases store items in row-order on disk, and employ additional indexes for fast search on specific fields. Answering a query requires retrieving the entirety of every row that matches. They support complex transactions and query models.

Column stores are a more recent database design where data is laid out on disk by column. They support higher insert rates than row stores, and can efficiently answer queries with a few terms, because only the columns present in the query need to be loaded. However, they do not offer good support for transactions and joins, and perform poorly when an entire row needs to be recomposed.

Spatial trees are designed for indexing multi-dimensional numeric data, such as geographical information or points in a numeric space. They are good for answering range and proximity queries on spatial data, and can be used for clustering.

Inverted indexes are designed for textual data, but can be used for any sort of point query. In an inverted index, each keyword or value has an index, containing a list of pointers to matching documents. In order to implement inverted indexes for structured data, each field must have a list of keywords and associated indexes. Inverted indexes are excellent for sparse data. Since the structure is simple, inverted indexes can also be implemented on top of other database structures. For instance, a relational database might have a table containing keywords as a primary key, and a column containing a list of documents.

FastBit [29] is designed explicitly for scientific databases. FastBit is a bitmap index optimized for high dimensional scientific data, which performs extremely well on numeric data, and is very compact. It performs well on point and range queries.

2.3 Data Sets

In order to get a representative sample of scientific metadata, we chose files from Dryad [1], the Wide-field Infrared Survey Explorer (WISE) All-Sky Release [5], Argo [6], and carbon-14 observations from Oakridge National Laboratories [16]. Dryad was further divided into data using The Open Archives Initiative (*OAI*) Protocol for Metadata Harvesting [4], and the Metadata Encoding & Transmission Standard (*METS*) [3]. We chose these data sets because they are readily available, and cover a wide range of science one might find in a large computing installation. Our goal is to characterize what could be expected in a system where one or more of these data types is resident. While some scientific computing installations may handle only a single kind of data, the largest installations support a wide range of scientific research, and will have indexes which must support multiple types of data.

While our choices were made on the basis of breadth, we also find that they each have very different metadata characteristics, offering a broad perspective on index requirements. We summarize their characteristics in Table 1. Where we subsampled, we assume that the data is uniform.

To analyze the data, we decomposed each data set into a columnar format by generating sorted lists from each field. Where data sets were natively in a non-tabular format, such as XML or NetCDF, we generated a list of all available fields, and then grouped elements into sorted lists based on tag or array name, respectively. For row analyses, we used the top level item and the array position, respectively, to create a row ID.

3. ANALYSIS AND RESULTS

To characterize the data, we examine the *sparsity* of fields, how often values are missing from a given field, since some indexes cannot handle sparse data, or must store it as a null value. We analyze the *atomicity* of fields, *i.e.* whether a field can be present multiple times in a single item, to consider the need for indexes which handle multiple field values. We then examine the *entropy* of fields,which affects query selectivity and index compression. Finally, we look at the overall distribution of data types for fields, both in terms of storage and semantic data types, to determine necessary

index types. Each of these tests guides choices when building a scientific data index or indexing system. We summarize our findings in Table 3.

3.1 Sparsity

We determine the total number of fields possible for any item, and then calculate what number of those fields were actually present in each of the individual items. Many indexing systems assume a fixed schema, where all fields are present for all items of the same type. In contrast, we find that even within a single discipline elements tend to be sparse, as shown in Figure 1. Sparse data can't be represented in some indexes, and can impact index size.

Sparsity is challenging for spatial tree based indexing schemes, which cannot place data with missing values in the tree, and must use estimation to fill in missing values, creating fake data that must then be stored [26]. A single table row-based index will also have space implications. Even if a table is built for each data type, it will still need to store nulls for missing values [12]. Although bitmap indexes are capable of storing data using masks, such that an element is only stored when data is present for a field, FastBit does not currently have mask support [7]. Inverted indexes and column stores only store data when data is present for that field, and will have space savings for sparse data.

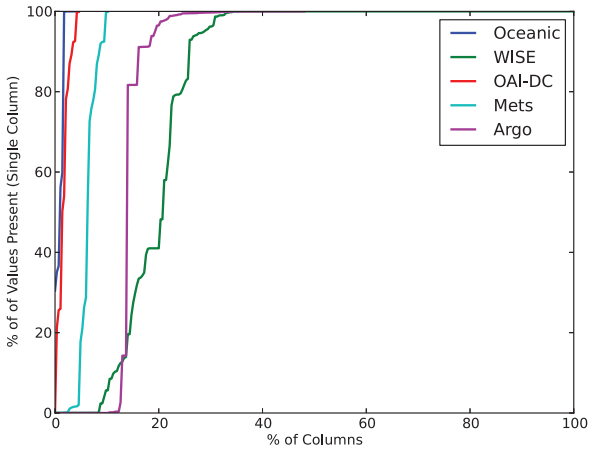

Figure 1: Sparsity for all data sets. For a randomly chosen element from X% of columns, there is a Y% chance it will be null. We find that even within one discipline, fields are sparse. Only a few fields are present in all items, and these fields tend to be unique identifiers for the system's use. A poor choice of index will waste space storing nulls, or be unable to represent the data at all.

3.2 Atomicity

Atomicity describes how often a field can be present for an data item (*i.e.,* a file). An *atomic* value is one where only zero or one elements can be present in a field. For instance, tabular data is strictly atomic, since it can only support one value per column in a row. Non-atomic data may represent an list of many elements in a field, such as a list of authors, or a single range element, such as a geographic area. Non-atomic data requires an index design that can represent multiple values or ranges, and match them to queries. We determine the distribution of non-atomic values in our data sets.

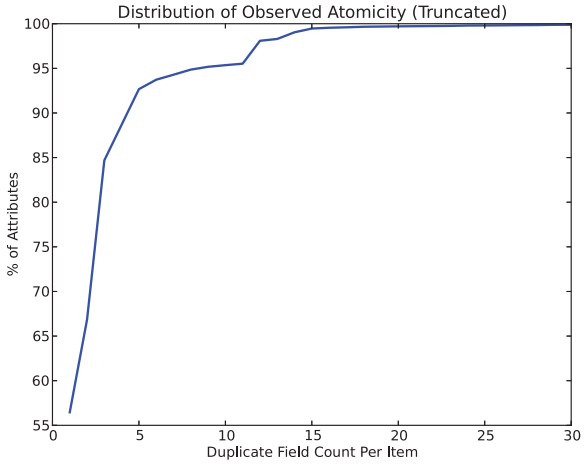

Figure 2: Field value counts for both OAI and METS data, truncated on both axes to show detail, since one item in our data has over eight hundred species associated with it. Only 55% of Dryad fields are atomic, *i.e.,* have a single entry. Less than 1% of fields have more than fifteen entries, but can go much higher. Indexes which cannot handle multiple values for a field for a single entry will be unable to store this data, and a poor choice of index design can result in large amounts of duplicated data.

While most of the data sets we examined are strictly atomic, the biology data have many fields with multiple elements per field for a single item, as shown in Figure 2, and one of the numerical fields in the ORNL data mixes ranges and point values. Any system which supports a variety of scientific metadata must handle non-atomic data. Both list entries and range entries will cause problems for spatial trees, which expect point values. At best, a spatial tree will require an additional data point for each value, leading to duplicated data and consistency problems. Inverted indexes cannot represent range values at all. With careful schema design an RDBMS can support list and range values. Column stores and bitmap indexes will also need schema design to handle range values, but can support high cardinality list values in a single column given a simple schema, and are a better choice.

3.3 Entropy

Entropy serves two purposes. First, it allows us to consider the selectiveness of queries. Fields with low entropy do a poor job of reducing the results size, whereas high entropy fields can be very selective. Secondly, it allows us to characterize compressibility. Low entropy data is easy to compress, and is less expensive to store. Entropy is agnostic to data type, which allows us to do direct comparisons of string and numeric data.

We chose to focus the entropy of whole values, to study query selectiveness. Whole value entropy does not entirely

	Column stores	Row stores	Spatial trees	Inverted Indexes	HDF5	FastBit
High dimensional data	Yes	Yes	No	Yes	Yes	Yes
Sparse data	Yes	Stores nulls	No	Yes	Yes	Stores nulls
Multiple values	Yes	More schema design	No	List, not range	Yes	Yes
Non-numeric data	Yes	Yes	No	Yes	Yes	No
Range queries	Yes	Yes	Yes	No	Yes	Yes
Specialized indexes	Yes	Yes	No	No	No	No
High compression	Yes	No	No	Yes	No	Yes

Table 2: Comparing index types

capture compressibility, but serves as an upper bound, more compression is often possible for fields which can be segmented or bucketed, such as free text and numerical data. We examine block-based column and row entropy, such as a database might use during compression, to compare their effectiveness. We discounted null values, since these are not relevant to selectivity.

We examined the entropy within each data set in two ways, using the bit-wise Shannon entropy

$$H(x) = -\sum_{i=1}^{n} p(x_i) log_2\, p(x_i)$$

where $p(x_i)$ is the probability of value x_i occurring. For columns, we sorted the values for each field (corresponding to a column), divided it into 4K blocks and calculated each block's entropy. For row entropy, we took all fields present in an item (excluding nulls), and took enough rows to fill a 4KB block, then calculated entropy.

This allows us to consider the effects of column organization versus row organization for compression. Field entropy also characterizes the selectiveness of a query on that field. Fields with very low entropy are very non-specific. In the worst case, an entropy of zero, where every value is the same, that field does not allow us to eliminate any possible results. By contrast, very high entropy fields, such as a unique identifier, quickly narrow the search space. In between are fields which are moderately selective.

As Figure 3 shows, row entropy is very low variance. This is intuitive, since taking entropy over all fields will average the entropy of those fields, leading to lower inter-block variance. However, mixing different types increases total entropy, meaning row data will be harder to compress. Column stores, inverted indexes, and bitmap indexes can take advantage of the structure of low entropy data in order to compress it. Some spatial trees can take advantage of data entropy to balance trees efficiently, but do not compress the data itself, and row stores cannot take full advantage of low entropy columns, since they must average entropy over all columns.

3.4 Data Types

We examine the data types of the fields, based on the data set documentation. Knowing the distribution of field types is useful for index choices. Different index designs are better for text versus numeric data, and there are efficient specialized indexes for data which is geospatial or time based. To examine type distributions, we categorize fields as numeric or string data. We then further examine them to determine if they are geospatial, a set of flags encoded as a string or number, *etc.* In Table 3.4, we show the distributions of raw data types and semantic types.

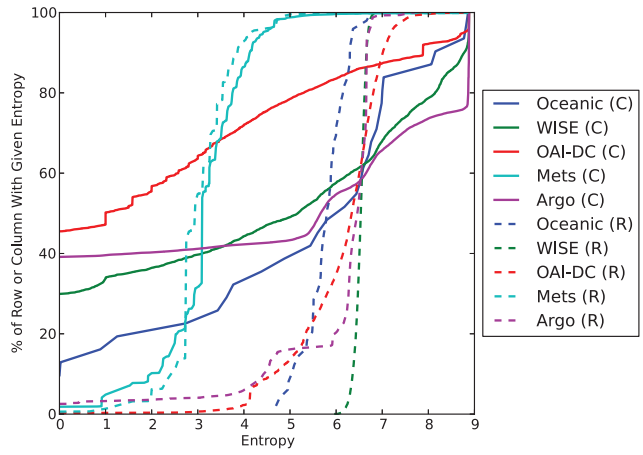

Figure 3: CDF of entropy for all fields in each data set. Dashed lines are row entropy, and solid lines are field entropy. Fields consistently have lower entropy than rows, indicating they will compress better, allowing more indexes to be stored in the same amount of memory or disk. However, field entropy varies significantly, suggesting that some fields are much more useful to query (and therefore index) than others.

Indexes which cannot handle string data, such as bitmap indexes and spatial trees, will not suffice for indexing scientific data. On the other hand, spatial indexes can be very useful as part of an indexing system, and bitmap indexes can be very efficient for flag sets and numeric data. The ideal system will take advantage of the semantics and structure of each field to search efficiently over heterogeneous data.

4. RELATED WORK

Here we describe related work in the areas of file system indexing, indexing for sparse and semi-structured data, and metadata studies.

4.1 File System Indexing

Spyglass [19] and Smartstore [17], were the first to suggest using spatial indexes for metadata. While these performed well on their test data, they focused strictly on POSIX metadata. Loris [27] and Pantheon [21] were both indexing systems tested for system metadata only. Pantheon used B-trees, which are row-based, and will face challenges with sparse data. Loris used log-structured merge-trees [22]. BeFS [15] was designed to handle both system metadata and

		Dryad (46 fields)	WISE (285 fields)	Argo (108 fields)	ORNL (14 fields)	Totals 453 fields
Storage type	String	100%	4%	62%	29%	28%
	Numeric	0%	96%	38%	71%	72%
Semantic type	Date	2%	4%	7%	7%	5%
	Spatial	2%	9%	2%	21%	7%
	Flagsets	0%	19%	14%	0%	15%
	Native types	96%	68%	77%	72%	73%

Table 3: Data types in scientific data for all data sets. We examine storage types, and semantic types that can have specialized indexes. String support is a must, and having native index support for time and space can significantly speed up queries.

extended metadata. In BeFS, all metadata was stored in a B+-tree, using row-major order. This technique was effective at desktop scales, but it suffers from problems with sparse, heterogeneous data.

4.2 Scientific Indexing

Systems such as FastBit [29] and FastQuery [11] are designed specifically for a certain type of scientific data. They have evaluated against scientific data, but only focused on numeric data. As previously mentioned, scientific data draws from a wide mix of types, not only numeric data. In addition, FastBit and FastQuery used a mix of synthetic and real data sets, but do not provide a close analysis of the contents of scientific files that would allow others to use similar data for testing, a gap which we propose to fill.

4.3 Metadata studies

There have been a number of previous metadata studies. However, they have focused exclusively on file system metadata and file types, rather than rich scientific metadata. For instance, Douceur's large-scale study of file-system contents [14], and Agrawal's five-year study of file-system metadata [9], also did detailed statistical analysis of distributions. Both focused on desktops. On a larger scale, we note Leung's large scale network file system study [20], which tracked behavior and file system metadata for corporate file servers. Perhaps the closest to our research are Dayal's study of HPC at rest [13] and Wang's study of HPC workloads [28]. However, they focused on file system metadata, not rich metadata.

4.4 Other indexing

Indexing shares many challenges with databases as well as file systems. Column stores such as C-store [24], HBase [2], or Cassandra [18], are one popular approach to dealing with sparse data. These have some advantages for scientific data, since they are well organized for tasks such as computing maximums, minimums, and averages. WideTable [12] was specifically designed to meet the challenges of extremely sparse high-dimensional indexes.

Patil et al. [23] also explored the question of appropriate architectures for searchable metadata in file systems. They suggested using BigTable [10] as the underlying storage for a file system. BigTable has good support for sparse indexes, and is highly scalable.

5. CONCLUSIONS

In this paper, we have examined scientific metadata, and demonstrated that it is large, sparse, heterogenous, high entropy, and high dimensional. Based on our findings, existing approaches to file system indexing, such as spatial trees and row major databases, will perform poorly for indexing scientific metadata. We conclude that column stores are an excellent fit for scientific data, and can answer file system search queries in a very efficient fashion.

Acknowledgements

This work has been supported by grants from NASA, the DoE, and the NSF. We also thank our industrial sponsors.

This publication makes use of data products from Oakridge National Laboratories, Dryad, and the Wide-field Infrared Survey Explorer, which is a joint project of the University of California, Los Angeles, and the Jet Propulsion Laboratory/California Institute of Technology, funded by the National Aeronautics and Space Administration, as well as data collected and made freely available by the International Argo Program and the national programs that contribute to it.

6. REFERENCES

[1] Dryad. `http://www.datadryad.org/`, September 2012.

[2] Hbase. `http://hbase.apache.org/`, September 2012.

[3] Metadata Encoding & Transmission Standard. `http://www.loc.gov/standards/mets/`, November 2012.

[4] The Open Archives Initiative Protocol for Metadata Harvesting. `http://www.openarchives.org/OAI/openarchivesprotocol.html`, November 2012.

[5] Wide-field Infrared Survey Explorer (WISE) All-Sky Release. `http://irsadist.ipac.caltech.edu/wise-allsky/`, September 2012.

[6] Argo. `http://www.argodatamgt.org/`, March 2013.

[7] [fastbit-users] sparse data. `https://hpcrdm.lbl.gov/pipermail/fastbit-users/2012-October/001510.html`, Mar 2013.

[8] 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.

[9] N. Agrawal, W. J. Bolosky, J. R. Douceur, and J. R. Lorch. A five-year study of file-system metadata. In

Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST), pages 31–45, Feb. 2007.

[10] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI)*, Seattle, WA, Nov. 2006.

[11] J. Chou, K. Wu, and P. Prabhat. Fastquery: A parallel indexing system for scientific data. In *Cluster Computing (CLUSTER), 2011 IEEE International Conference on*, pages 455–464. IEEE, 2011.

[12] E. Chu, J. Beckmann, and J. Naughton. The case for a wide-table approach to manage sparse relational data sets. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, SIGMOD '07, pages 821–832, New York, NY, USA, 2007. ACM.

[13] S. Dayal. Characterizing HEC storage systems at rest. Technical report, Carnegie-Mellon University, 2008.

[14] J. R. Douceur and W. J. Bolosky. A large-scale study of file-system contents. In *Proceedings of the 1999 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, SIGMETRICS '99, 1999.

[15] D. Giampaolo. *Practical File System Design with the Be File Sstem*. Morgan Kaufmann, 1st edition, 1999.

[16] H. Graven, A. Kozyr, and R. M. Key. Historical observations of oceanic radiocarbon conducted prior to GEOSECS. `http://cdiac.ornl.gov/ftp/oceans/Historical\_C14\_obs/`, 2012.

[17] 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.

[18] A. Lakshman and P. Malik. Cassandra: a decentralized structured storage system. *SIGOPS Oper. Syst. Rev.*, 44(2):35–40, Apr. 2010.

[19] 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.

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

[21] J. Naps, M. Mokbel, and D. Du. Pantheon: Exascale file system search for scientific computing. In *Scientific and Statistical Database Management*, 2011.

[22] P. O'Neil, E. Cheng, D. Gawlick, and E. O'Neil. The log-structured merge-tree (LSM-tree). *Acta Inf.*, 33(4):351–385, June 1996.

[23] S. Patil, G. A. Gibson, G. R. Ganger, J. Lopez, M. Polte, W. Tantisiroj, and L. Xiao. In search of an API for scalable file systems: under the table or above it? In *Proceedings of the 2009 conference on Hot topics in cloud computing*, HotCloud'09, Berkeley, CA, USA, 2009. USENIX Association.

[24] M. Stonebraker, D. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. O'Neil, P. O'Neil, A. Rasin, N. Tran, and S. Zdonik. C-Store: A column oriented DBMS. In *Proceedings of the 31st Conference on Very Large Databases (VLDB)*, pages 553–564, Trondheim, Norway, 2005.

[25] 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.

[26] D. A. Talbert and D. Fisher. An empirical analysis of techniques for constructing and searching k-dimensional trees. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pages 26–33, New York, NY, USA, 2000. ACM.

[27] R. van Heuven van Staereling, R. Appuswamy, D. van Moolenbroek, and A. Tanenbaum. Efficient, Modular Metadata Management with Loris. In *Networking, Architecture and Storage (NAS), 2011 6th IEEE International Conference on*, pages 278 –287, July 2011.

[28] F. Wang, Q. Xin, B. Hong, S. A. Brandt, E. L. Miller, D. D. E. Long, and T. T. McLarty. File system workload analysis for large scale scientific computing applications. In *Proceedings of the 21st IEEE / 12th NASA Goddard Conference on Mass Storage Systems and Technologies*, pages 139–152, College Park, MD, Apr. 2004.

[29] K. Wu, S. Ahern, E. W. Bethel, J. Chen, H. Childs, E. Cormier-Michel, C. Geddes, J. Gu, H. Hagen, B. Hamann, W. Koegler, J. Lauret, J. Meredith, P. Messmer, E. Otoo, V. Perevoztchikov, A. Poskanzer, Prabhat, O. Rübel, A. Shoshani, A. Sim, K. Stockinger, G. Weber, and W.-M. Zhang. Fastbit: interactively searching massive data. *Journal of Physics: Conference Series*, 180(1), 2009.