Towards a provenance-aware distributed filesystem

Chen Shou*, Dongfang Zhao*, Tanu Malik†, Ioan Raicu*‡
{*cshou, dzhao8*}*@hawk.iit.edu, tanum@ci.uchicago.edu, iraicu@cs.iit.edu*
Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA
†*Computation Institute, The University of Chicago, Chicago, IL, USA*
‡*Math and Computer Science Division, Argonne National Laboratory, Argonne, IL, USA*

Abstract

It has become increasingly important to capture and understand the origins and derivation of data (its provenance). A key issue in evaluating the feasibility of data provenance is its performance, overheads, and scalability. This paper presents a provenance-aware distributed filesystem, that offers excellent scalability while retaining the provenance overhead negligible under certain conditions. This work integrated two recent research projects, SPADE (Support for Provenance Auditing in Distributed Environments) and FusionFS (Fusion distributed File System) with simple and efficient communication protocols. The preliminary results on a 32-node cluster show that FusionFS+SPADE is a promising prototype with negligible provenance overhead and has promise to scale to larger scales as FusionFS has been shown to scale.

1 Introduction

Data provenance is the service describing how data was derived. The tracing of provenance is helpful in rich and descriptive metadata to accompany the data in order to better understand it. Provenance can be collected at different layers from application all the way down to the operating system. For example, the provenance at filesystem level helps analyst mining patterns hidden behind the data. It can also help in security audits, helping identify compromised data in security breaches.

Distributed file systems have so far proposed a central system for provenance collection [1]. This is a performance bottleneck, especially for file systems meant for extreme-scales. Various efforts have been made in distributing the metadata management of parallel and distributed file systems. For example, the FusionFS [2] implements a distributed metadata management based on distributed hash tables [3]. Ceph [4] also implements a distributed metadata management system.

One obvious solution to solve the bottleneck, is to develop a completely distributed provenance service from scratch. Yet a more efficient way is to integrate an add-on module to some existing filesystems with distributed metadata. SPADE [5] is an open source software infrastructure for data provenance collection and management. It applies graph database to store provenance and provides distributed query module. SPADE turns out to be a good choice to be integrated into FusionFS since both systems have similar manifestation of distributed concepts: (1) FusionFS provides a POSIX interface which makes a perfect corresponding for SPADE user-level file system provenance collection; (2) both systems work in a decentralized way which actively exploit the resource of each node.

This paper introduces a provenance-aware distributed file system, that aims to offer excellent scalability while retaining the provenance overhead negligible. We integrated SPADE and FusionFS, with simple and efficient communication protocols. The preliminary results on a 32-node cluster show that FusionFS+SPADE is a promising prototype with negligible provenance overhead and has promise to scale to petascale and beyond. FusionFS on its own, has shown to scale to 1K-nodes [2], and its design has been architected to scale to 1M-nodes, i.e. exascale. The ultimate goal of this work is to enable the negligible-overhead provenance in a distributed file system that is scalable up to exascale.

The remainder of this paper is organized as follows. We review some related work in Section 2. In Section 3 we describe the design and implementation of FusionFS+SPADE. Section 4 evaluates its performance, and Section 5 concludes this paper and discusses the future work.

2 Related Work

As distributed systems become more ubiquitous and complex, there is a growing emphasis on the need for tracking provenance. A good review is presented in [6]. Many Grid systems like Chimera [7] and the Provenance-Aware Service Oriented Architecture (PASOA) [8] provide provenance tracking mechanisms for various applications. However these systems are very domain specific and do not capture provenance at the file system level. The Distributed Provenance Aware Storage System (DPASS) tracks the provenance of files in a distributed file system by intercepting file system operations and sending this information via a netlink socket to user level daemon that collects provenance in a database server [9]. The provenance is however, collected in a centralized fashion, which is a poor design choice for distributed file systems meant for extreme scales. Similarly in efficient retrieval of files, provenance is collected centrally [1].

We believe that the integration of FusionFS and SPADE offers an ideal combination for an initial prototype. The major drawback we see at this point is the reliance on JAVA for SPADE, which makes it a poor choice for portability to some of the large high-end computing systems. We hope to address this in future work.

3 Design and Implementation

3.1 Architecture

The high-level architecture of the proposed work is showed in Figure 1. Each node has two services installed: FusionFS service and SPADE service. One service type can only communicate to the other type on the local node. That is, a SPADE service only communicates with its local FusionFS service, and vice versa. For services of the same type (e.g. FusionFS ⇔ FusionFS, SPADE ⇔ SPADE), they are free to talk to others remotely.

Figure 2 shows the logical structure of FusionFS service and SPADE service on a single node. FusionFS provides POSIX interface to the applications. When the application makes a request to FusionFS, this request is interpreted and transmitted to SPADE by some protocol which will be discussed in the next subsection. SPADE then takes care of the provenance management and persistence to local database.

Communication between SPADE services only occurs during provenance querying. When executing a query, a SPADE service would spread the query (based on network provenance stored locally) to all remote SPADE services that also contain related provenance, and wait for response. In the end, SPADE would merge all the responded query results and output to the result file.

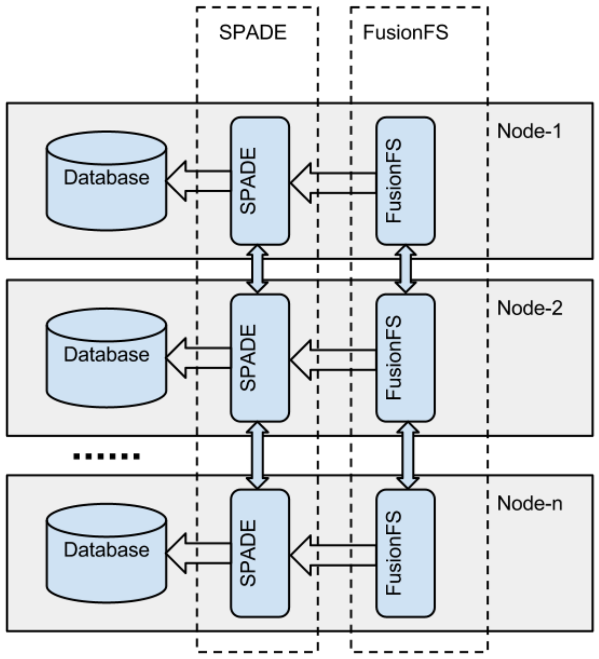

Figure 1: FusionFS+SPADE architecture overview

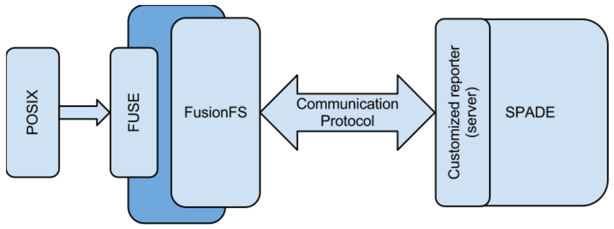

Figure 2: Relationship between FusionFS and SPADE

3.2 Integration of SPADE and FusionFS

The key challenge of the proposed work is how to efficiently integrate SPADE and FusionFS. All communication between these two services is implemented with TCP. Asynchronous communication is not preferred because of the short life cycle of some processes. SPADE collects parts of the process information based on system files under directory /proc/pid. If a process starts and terminates too fast for SPADE to catch, there will be provenance loss. Therefore it is critical to keep synchronous communication between SPADE and FusionFS, at least while the two systems are completely decoupled. We hope to address this in future work with a tighter integration between FusionFS and SPADE.

Most communication between SPADE and FusionFS consists of simple operation bindings. For example, FusionFS write operation invokes SPADE to collect write

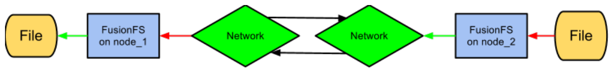

Figure 3: Network Transmission

provenance for this operation. However, as a distributed file system, FusionFS sometimes needs to migrate files between nodes. The original network provenance collection in SPADE is not optimized for FusionFS, thus it is too expensive. We make some customization to the network provenance collection in PAFS, to fully hide unnecessary provenance data outside FusionFS.

In order to make the collected provenance consist with Open Provenance Model (OPM), when there is a network transmission, SPADE creates a "dummy" FusionFS process vertex to connect two artifacts: a file vertex and a network vertex. We call it a "dummy" process because clients do not need to be concerned with this process when querying provenance; it is just a symbol to indicate the network transmission is triggered by FusionFS in OPM. Figure 3 shows how a network transmission is represented.

3.3 File-Level vs. Block-Level

One common practice in file manipulations is to split (large) files into blocks to improve the space efficiency and responsive time. However, for the purpose of provenance, it is less interesting to keep track of file traces at the block level: in most cases, a file-level provenance would suffice.

This work implements both the file-level and the block-level provenance tracings, namely, the fine-grained provenance and the coarse-grained provenance: (1) **fine-grained** – the same as SPADE, which collects provenance on each I/O run; and (2) **coarse-grained** – each operation unit (e.g. read, write, etc) invokes one provenance collection.

4 Evaluation

We carried out three experiments to evaluate this work. The testbed is a 32-node cluster, where each node has two Quad-Core AMD Opteron 2.3GHz processors with 8GB memory. All nodes are interconnected by 1Gbps Ethernet. All experiments are repeated at least 3 times to obtain stable results (i.e. within 5% difference).

4.1 Single-Node Throughput

We first measured performance of provenance collection within FusionFS on a single node. A client reads/writes a 100MB file from/to FusionFS. We compare the performance between fine-grained and coarse-grained provenance collection with different block sizes. The benchmark we used is IOZone [10], which is carefully tuned to avoid operating system cache.

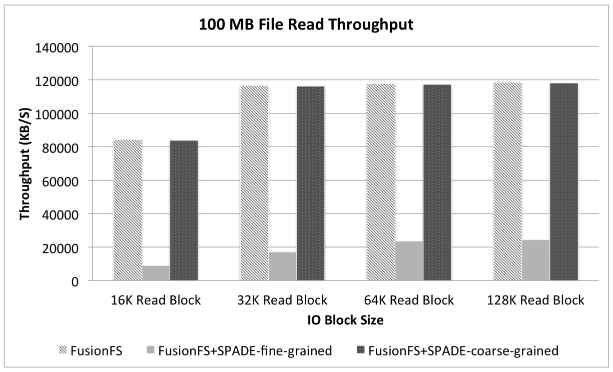

Figure 4: Read Throughput

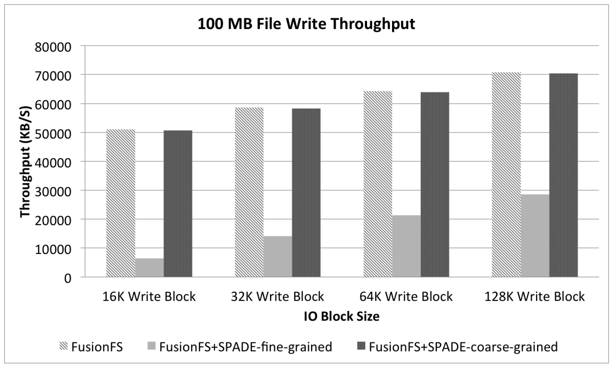

Figure 5: Write Throughput

Figure 4 and Figure 5 show that a fine-grained provenance collection introduces a high overhead. Even though a larger block size could reduce the overhead to some degree, the number is still significantly high (i.e. around 75%), compared to coarse-grained provenance (i.e. less than 5%). This is expected since a bigger I/O block size results in fewer I/O runs, which further involves less time to collect provenance (SPADE spends on average 2.5 ms for each provenance recording, which corresponds to a single I/O run in fine-grained provenance collection).

We investigated the sensitivity of file size attributed to the I/O throughput. Figure 6 shows that a fine-grained provenance collection might have a fair overhead when the file size is small enough, while a coarse-grained one keeps a low overhead (less than 5%) no matter what the file size is.

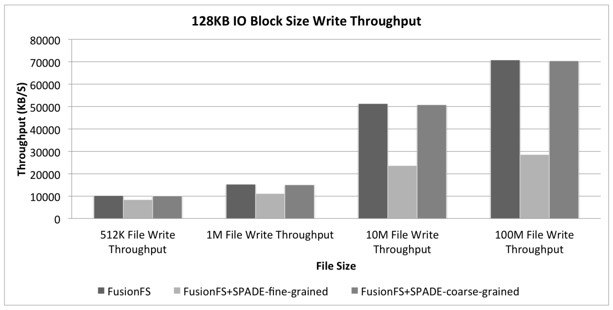

Figure 6: Overhead of 128KB block size

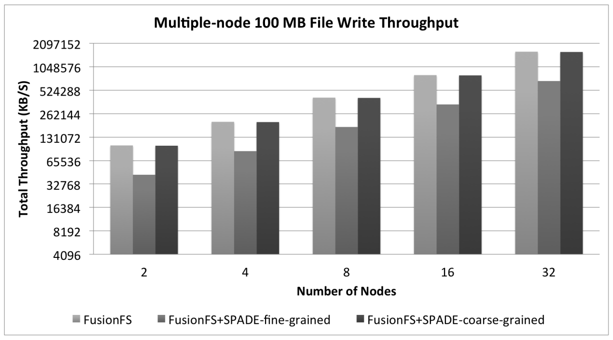

Figure 7: Multiple-Node 100MB Write Throughtput

4.2 Multiple-Node Throughput

In the 32-node cluster, multiple clients read/write distinct files from/to FusionFS. The file size is set to 100MB and the I/O block size is set to 128KB.

In Figure 7, a coarse-grained provenance collection shows a much better performance than the fine-grained counterpart (consistent with the single-node benchmark results). Both fine-grained and coarse-grained provenance show excellent scalability with linear increase in performance. This can be explained by two facts: (1) SPADE only collects provenance of the local node, and (2) FusionFS scales linearly with respect to the number of nodes by getting high data locality in the data access pattern evaluated. We have evaluated FusionFS (without SPADE) at scales of up to 1K nodes on a IBM BlueGene/P supercomputer with similar excellent results. We will conduct larger scale experiments of FusionFS+SPADE in future work.

4.3 Queries in Distributed Systems

We are interested in the query time of the provenance of a particular file that has been read by multiple remote nodes. This write-once-read-many is a very frequent pattern in the context of a distributed system. The query is

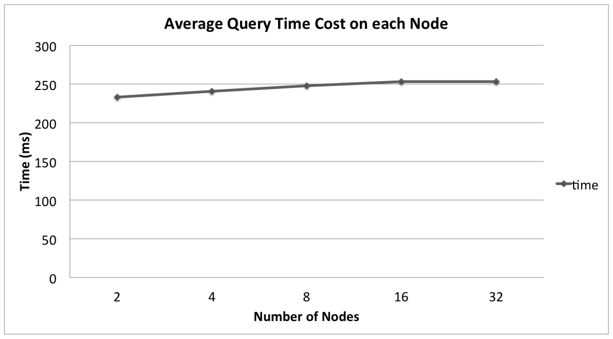

Figure 8: Query Time Cost

shown in the following format:

query lineage descendants $vertex-id$ 100 null filename:$test.file.name$

Since SPADE (with version) does not support executing sub-query in parallel, the total query time increases as it scales up. However, according to Figure 8, with different scales from 2 to 32 nodes, the average per-node query time is about constant, indicating that adding more nodes will not put more burden to the provenance system. This is expected, since the underlying FusionFS has an excellent scalability and SPADE on each node adds negligible overheads locally.

5 Conclusion and Future Work

This paper presents a provenance-aware distributed filesystem with excellent scalability for extreme-scale computing. We integrate two latest research projects, SPADE and FusionFS, as two building blocks for the provenance management and the underlying distributed filesystem. This work aimed at exploring the feasibility of providing provenance tracking at the storage layer at modest scales. Our preliminary results show that, at least at the coarse-grained level, FusionFS+SPADE is very promising to deliver data provenance tracking in the storage layer.

We plan to develop more efficient, reliable, and portable protocols between SPADE and FusionFS. We hope to offer a hybrid granularity of provenance, rather than two extremes i.e. fine-grained and coarse-grained, to meet different needs in scientific computing in the near future. We will also work with the Swift parallel programming system [11] to deploy real scientific applications on FusionFS+SPADE at much larger scales, at petascales and beyond with tens of thousands of nodes.

Acknowledgement

This work was supported by the National Science Foundation (NSF) under grant OCI-1054974.

References

[1] Kiran-Kumar Muniswamy-Reddy, David A. Holland, Uri Braun, and Margo Seltzer. Provenance-aware storage systems. In *Proceedings of the annual conference on USENIX '06 Annual Technical Conference*, ATEC '06, pages 4–4, Berkeley, CA, USA, 2006. USENIX Association.

[2] Dongfang Zhao and Ioan Raicu. Distributed File Systems for Exascale Computing (poster). ACM/IEEE Supercomputing, Salt Lake City, UT, 2012.

[3] Tonglin Li, Xiaobing Zhou, Kevin Brandstatter, Dongfang Zhao, Ke Wang, Anupam Rajendran, Zhao Zhang, and Ioan Raicu. ZHT: A Light-weight Reliable Persistent Dynamic Scalable Zero-hop Distributed Hash Table. IEEE IPDPS, Boston, MA, 2013, to appear.

[4] Sage A. Weil, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long, and Carlos Maltzahn. Ceph: a scalable, high-performance distributed file system. In *Proceedings of the 7th symposium on Operating systems design and implementation*, OSDI '06, pages 307–320, Berkeley, CA, USA, 2006. USENIX Association.

[5] Ashish Gehani and Dawood Tariq. SPADE: Support for Provenance Auditing in Distributed Environments. ACM/USENIX Middleware, pages 101–120, 2012.

[6] Kiran-Kumar Muniswamy-Reddy. Foundations for provenance-aware systems. Doctoral dissertation, Harvard University, 2010.

[7] Ian T. Foster, Jens-S. Vckler, Michael Wilde, and Yong Zhao. The virtual data grid: A new model and architecture for data-intensive collaboration. In *CIDR'03*, pages –1–1, 2003.

[8] Provenance aware service oriented architecture. http://twiki.pasoa.ecs.soton.ac.uk/bin/view/ pasoa/webhome.

[9] Rich Metadata, Aleatha Parker-wood, Rich Metadata, Darrell D. E. Long, and Aleatha Parker-wood. Proposal for advancement to candidacyuniversity of california santa cruz making sense of file systems through provenance, 2012.

[10] D. Capps. IOzone Filesystem Benchmark. *www.iozone.org*, 2008.

[11] Yong Zhao, Mihael Hategan, Ben Clifford, Ian T. Foster, Gregor von Laszewski, Veronika Nefedova, Ioan Raicu, Tiberiu Stef-Praun, and Michael Wilde. Swift: Fast, reliable, loosely coupled parallel computation. In *IEEE SCW*, pages 199–206, 2007.