Shafeeq Sinnamohideen

4221 Winterburn Ave. Apt B-301
Pittsburgh, PA 15207
412-443-0577
shafeeq@cs.cmu.edu
http://www.cs.cmu.edu/~shafeeq

Carnegie Mellon University
CIC 2221G
5000 Forbes Ave.
Pittsburgh, PA 15213
412-268-5911

Objective

To obtain a challenging research or development position in which I can advance the state of the art in the area of large-scale distributed storage systems.

Education

Ph.D., Computer Science, May 2010 (expected)
Carnegie Mellon University, Pittsburgh, PA
Advisor: Professor Gregory R. Ganger

B.S., Electrical & Computer Engineering, May 2000
Carnegie Mellon University, Pittsburgh, PA

Honors and Awards

FAST 2005 best paper award
- *Paper title*: Ursa Minor: versatile cluster-based storage

Professional Experience

Graduate Student — **August 2002 – Present**
Carnegie Mellon University

Intern — **May 2003 – August 2003**
Intel Research Pittsburgh

Intern — **April 2002 – August 2002**
Intel Research Pittsburgh

Researcher Programmer — **January 2000 – April 2002**
Carnegie Mellon University, Coda Project

Student Researcher Programmer — **May 1998 – December 1999**
Carnegie Mellon University, Coda Project

Teaching Experience

Teaching Assistant — **Fall 2004**
CS 15-441, Computer Networks

Teaching Assistant — **Fall 2003**
CS 15-462, Computer Graphics I

Special Skills

- Experience in FreeBSD and Linux kernel programming.
- Experience in developing complex distributed file systems.
- Proficient in C, C++, Java, HTML, ML, Perl, and Verilog languges.

Research Summary

Research interests: storage systems, distributed systems, operating systems, metadata scalability, multi-server operations

Sep. 2004
Present

Eliminating cross-server operations in scalable storage systems
Transparent scalability is a goal of many distributed storage systems. That is, it should be possible to increase both capacity and throughput by adding servers and spreading data and work among them, without client applications and users being aware of which servers are hosting which data. This allows the system to balance load by using an internal mechanism to migrate objects from one server to another as it sees fit.

Providing transparent scalability, however, is complicated by the fact that some operations involve more than one object (e.g.: rename) and the defined semantics require that all be modified atomically. If any directory may be on any server, it is possible that both directories are on different servers. In order to provide atomicity in this case, a distributed transaction protocol, such as a two-phase commit, would be required. While well understood, these protocols are complex to implement and complex to verify. Furthermore, from examining file system traces, it is apparent that these cross-server operations occur very rarely. Thus the work the system implementer has to do a lot of work to provide a feature that is necessary, but infrequently used. Many systems choose instead to provide a user-visible boundary, such as a volume in AFS or a mountpoint in NFS, across which atomic operations are not supported.

I propose a solution to make system implementer's task easier. If an operation involves objects on more than one server, the mechanism used to move objects for load balancing can be reused to move objects so that all the objects involved in the operation are on one server. That operation can now be executed using the single-server code path. Once it is complete, the objects can be migrated back to their original locations, or they can be left in place until they cause a load imbalance. Depending on the granularity and speed of object migration, it may be slower than a distributed protocol, but it is much simpler. If increased performance is necessary, the migration mechanism could potentially be optimized for this usage.

I have implemented this technique in the Metadata Service of the Parallel Data Lab's Ursa Minor storage system, where it works quite well — even with the overhead of migrating metadata at a "volume" granularity, scalability is linear for the mix of operations seen in well-known NFS traces and benchmarks. My thesis explores the applicability of this technique to a wider range of workloads and system architectures than those of Ursa Minor.

Jan 2007
Present

Zzyzx: Scalable Fault Tolerance through Byzantine Locking
Much of the complexity in Byzantine fault-tolerant replicated state-machine (RSM) protocols is due to the need to handle concurrent client accesses to the same data. In common file system workloads, however, concurrency is rare. Using a similar concept to my thesis work, Zzyzx is a new RSM protocol that adds a fast path on top of an existing, but slow, protocol. When there is no concurrency, Zzyzx can used its fast protocol, and when there is concurrency, it falls back to the underlying, slower, protocol. Doing so provides a significant performance and scalability improvement over prior RSM protocols, while providing the same fault-tolerance properties.

Jan 2004
Present

Self-* Storage: automating storage management
Reducing the human effort required to administer large storage systems is a major concern for future IT infrastructures. To this end, we are exploring ways to create self-configuring, self-organizing, self-tuning, self-healing, and self-managing systems made up of storage bricks. My area of interest is the metadata service that maintains the mapping of file name or object identifier to storage bricks and locations. The metadata service in the Parallel Data Lab's Self-* prototype, which I am responsible for, is a complex distributed system in its own right, and is the platform for my current research.

Jun. 2005
Sep. 2006

Improving small file performance in object-based storage
Current object-based storage systems, while offering excellent scalability and performance for large-file workloads, such as HPC applications, do not perform well for small-file workloads. This is largely due to the extra overhead of first contacting a metadata server to determine which OSD to contact, and then contacting the storage brick for a small access. As most workstation workloads, such as home directories, compilation, and email, are small file workloads, this is an impediment to using a single shared storage system for both tasks.

In this work, we explored two techniques that greatly reduced the need to interact with the metadata server. The first is server-driven metadata prefetching, in which the server returns the metadata for several related objects in addition to the metadata requested. For instance, if the server knows an object is a directory, it could return the metadata for all files within the directory, on the grounds that a client is likely to open one of them. Client-driven prefetching would be difficult in this case, because the client doesn't know which files are in the directory until it has read the directory itself. My co-authors explored a complementary technique to make determining related objects easier. When the system assigns an object ID to a file, the object ID can be chosen so that files in the same directory receive object IDs numerically close to each other. Similarly, files in nearby directories receive OIDs that slightly further away. This encoding of the filesystem hierarchy into the object ID allows the OSD to store related objects near each other, preserving spatial locality. Similarly, the metadata server can find related objects by simply looking at numerical distance.

May 2003
Dec. 2003

Data Storage for Perishable Clients
The risk of lost writeback in distributed file systems using write-back caching has traditionally been accepted. For mobile or public clients with slow long-distance networks, for example, users in a wireless equipped cafe, however, the new data buffered at the client may represent several hours or days of user effort, instead of the usual few seconds or minutes. Furthermore, the client machines themselves are at risk of theft, damage, or operator intervention in the case of public machines. This research focused on methods of quantifying the risk of data loss and identifying policies and mechanisms for efficiently reducing this risk while not wasting resources. Methods included spreading data onto nearby surrogate machines and trickling data from them to the distant servers, as well as tuning the operation trickle-back policy to minimize risk instead of minimizing data transferred.

Jan. 2003
May 2003

Remote Lookaside Caching for Internet Suspend-Resume
This extension of Data Staging bridged three projects at Intel Research Pittsburgh : Internet Suspend-Resume, a virtual-machine migration system that uses the Coda distributed file system for data storage, CASLIB, a library allowing applications to access data or content-addressable storage devices, and lookaside caching, an extension to Coda that allows the client cache manager to fetch the data for a file from alternate sources. Basically, the file server provides a cryptographic hash of the file data, and if the client can locate a file with a matching hash from a faster source than the server, the client can use that source instead. This would be of great benefit for a mobile client connected to it's file server through a slow network path. I extended the lookaside caching system to use CASLIB to locate alternate sources for file data, providing an initial application for CASLIB. Because CASLIB provides a uniform interface for a variety of storage devices : object stores, local file systems, and DHTs, this composition increases the probability of finding alternate data sources.

Sep. 2001
Dec. 2002

Data Staging on Untrusted Surrogates
Despite network bandwidth improvements, mobile clients connected to distant file servers still experience slow file transfers and application unresponsiveness due to the latency of long-distance networks. Prefetching data into the client's cache is a known solution to this problem, but is not sufficient in the case of, for example, a handheld client with a small amount of local storage, or where aggressive prefetching would reduce battery life.

We explored a solution that speculatively stages data from the file server to a local, untrusted, surrogate server. The client can then service cache misses from the surrogate at local network speeds, providing significant performance improvements. Security and integrity are preserved by encrypting all data stored on the surrogate, and exchanging decryption keys and secure hashes over a low-bandwidth secure channel between the server and client.

Refereed publications

A transparently scalable metadata service for the Ursa Minor storage system. Shafeeq Sinnamohideen, Raja R. Sambasivan, Likun Liu, James Hendricks, Gregory R. Ganger. To appear in the proceedings of the 2010 USENIX Annual Technical Conference. June 2010. Boston, MA, USA.

Scalable Fault Tolerance through Byzantine Locking. James Hendricks, Shafeeq Sinnamohideen, Gregory R. Ganger, Michael K. Reiter. To appear in the proceedings of the 2010 International Conference on Dependable Systems and Networks (DSN 2010). June 2010. Chicago, IL, USA.

Ursa Minor: versatile cluster-based storage. Michael Abd-El-Malek, William V. Courtright II, Chuck Cranor, Gregory R. Ganger, James Hendricks, Andrew J. Klosterman, Michael Mesnier, Manish Prasad, Brandon Salmon, Raja R. Sambasivan, Shafeeq Sinnamohideen, John D. Strunk, Eno Thereska, Matthew Wachs, Jay J. Wylie. In the proceedings of the 4^{th} USENIX conference on File and Storage Technologies (FAST'05). December 2005. San Francisco, CA, USA.

Data Staging on Untrusted Surrogates. Jason Flinn, Shafeeq Sinnamohideen, Niraj Tolia, M. Satyanarayanan. In the proceedings of the 2^{nd} USENIX conference on File and Storage Technologies (FAST'03). March 2003, San Francisco, CA, USA.

The Case for Cyber Foraging. Rajesh Balan, Jason Flinn, M. Satyanarayanan, Shafeeq Sinnamohideen, Hen-I Yang. In the proceedings of the 10^{th} ACM SIGOPS European Workshop. September 2002. Saint-Emilion, France.

Write-Back Caching for Coda. Shafeeq Sinnamohideen, Lawrence Greenfield. In the proceedings of the 2^{nd} CMU Student Symposium on Computer Systems. October 1999, Pittsburgh.

Invited papers

Early experiences on the journey towards self-* storage. Michael Abd-El-Malek, William V. Courtright II, Chuck Cranor, Gregory R. Ganger, James Hendricks, Andrew J. Klosterman, Michael Mesnier, Manish Prasad, Brandon Salmon, Raja R. Sambasivan, Shafeeq Sinnamohideen, John D. Strunk, Eno Thereska, Matthew Wachs, Jay J. Wylie. In the Bulletin of the IEEE Computer Society Technical Committee on Data Engineering. September 2006.

Seamless Mobile Computing on Fixed Infrastructure Michael Kozuch, Mahadev Satyanarayanan, Thomas Bressoud, Casey Helfrich, Shafeeq Sinnamohideen. In IEEE Computer. July 2004.

Technical reports

A transparently scalable metadata service for the Ursa Minor storage system. Shafeeq Sinnamohideen, Raja R. Sambasivan, Likun Liu, James Hendricks, Gregory R. Ganger. Carnegie Mellon University Parallel Data Lab Technical Report CMU-PDL-10-102. March 2010.

Eliminating cross-server operations in scalable file systems. James Hendricks, Shafeeq Sinnamohideen, Raja R. Sambasivan, Gregory R. Ganger. Carnegie Mellon University Parallel Data Lab Technical Report CMU-PDL-06-105. May 2006.

Improving small file performance in object-based storage. James Hendricks, Raja R. Sambasivan, Shafeeq Sinnamohideen, Gregory R. Ganger. Carnegie Mellon University Parallel Data Lab Technical Report CMU-PDL-06-104. May 2006.

References

Dr. Gregory R. Ganger	**Dr. Charles Cranor**
Professor of ECE & CS	Research Faculty
Director, Parallel Data Lab	Systems Scientist, Parallel Data Lab
Carnegie Mellon University	Carnegie Mellon University
5000 Forbes Avenue	5000 Forbes Avenue
Pittsburgh, PA 15213	Pittsburgh, PA 15213
`ganger@ece.cmu.edu`	`chuck@ece.cmu.edu`