# contributed articles

**With scalable high-performance storage entirely in DRAM, RAMCloud will enable a new breed of data-intensive applications.**

BY JOHN OUSTERHOUT, PARAG AGRAWAL, DAVID ERICKSON, CHRISTOS KOZYRAKIS, JACOB LEVERICH, DAVID MAZIÈRES, SUBHASISH MITRA, ARAVIND NARAYANAN, DIEGO ONGARO, GURU PARULKAR, MENDEL ROSENBLUM, STEPHEN M. RUMBLE, ERIC STRATMANN, AND RYAN STUTSMAN

# The Case for RAMCloud

FOR THE PAST four decades magnetic disks have been the primary storage location for online information in computer systems. Over that period, disk technology has undergone dramatic improvements while being harnessed by higher-level storage systems (such as file systems and relational databases). However, disk performance has not improved as quickly as disk capacity, and developers find it increasingly difficult to scale disk-based systems to meet the needs of large-scale Web applications. Many computer scientists have proposed new approaches to disk-based storage as a solution, and others have suggested replacing disks with flash memory devices. In contrast, we say the solution is to shift the primary locus of online data from disk to DRAM, with disk relegated to a backup/archival role.

A new class of storage called RAMCloud will provide the storage substrate for many future applications. RAMCloud stores all of its information in the main memories of commodity servers and uses hundreds or thousands of these servers to create a large-scale storage system. Because all data is in DRAM at all times, RAMCloud promises 100x–1,000x lower latency than disk-based systems and 100x–1,000x greater throughput. Though individual memories are volatile, RAMCloud can use replication and backup techniques to provide data durability and availability equivalent to disk-based systems.

The combination of latency and scale offered by RAMCloud will change the storage landscape in three ways: First, it will simplify development of large-scale Web applications by eliminating many of the scalability issues that sap developer productivity today; it will also enable a new class of applications that manipulate data 100x–1,000x more intensively than is possible today; and it will provide the scalable storage substrate needed for cloud computing and other data–center applications.[3] A RAMCloud cluster can support a single large application or numerous smaller applications, allowing small applications to grow into large ones without additional complexity for the developer.

This article describes the RAMCloud concept, including how it will enable new applications. Building a practical RAMCloud requires solutions to several interesting research problems (such as how to provide durability for data stored in volatile DRAM and managing clusters with thousands of servers).

## » key insights

- The Web has driven development of new large-scale applications that have effectively scaled compute power and storage capacity but have not scaled storage access rates to match.

- DRAM-based storage can be made as durable and available as disk-based storage without giving up its performance advantages.

- DRAM-based storage is not just faster than disk or flash; for I/O-intensive workloads it is also cheaper and more energy efficient.
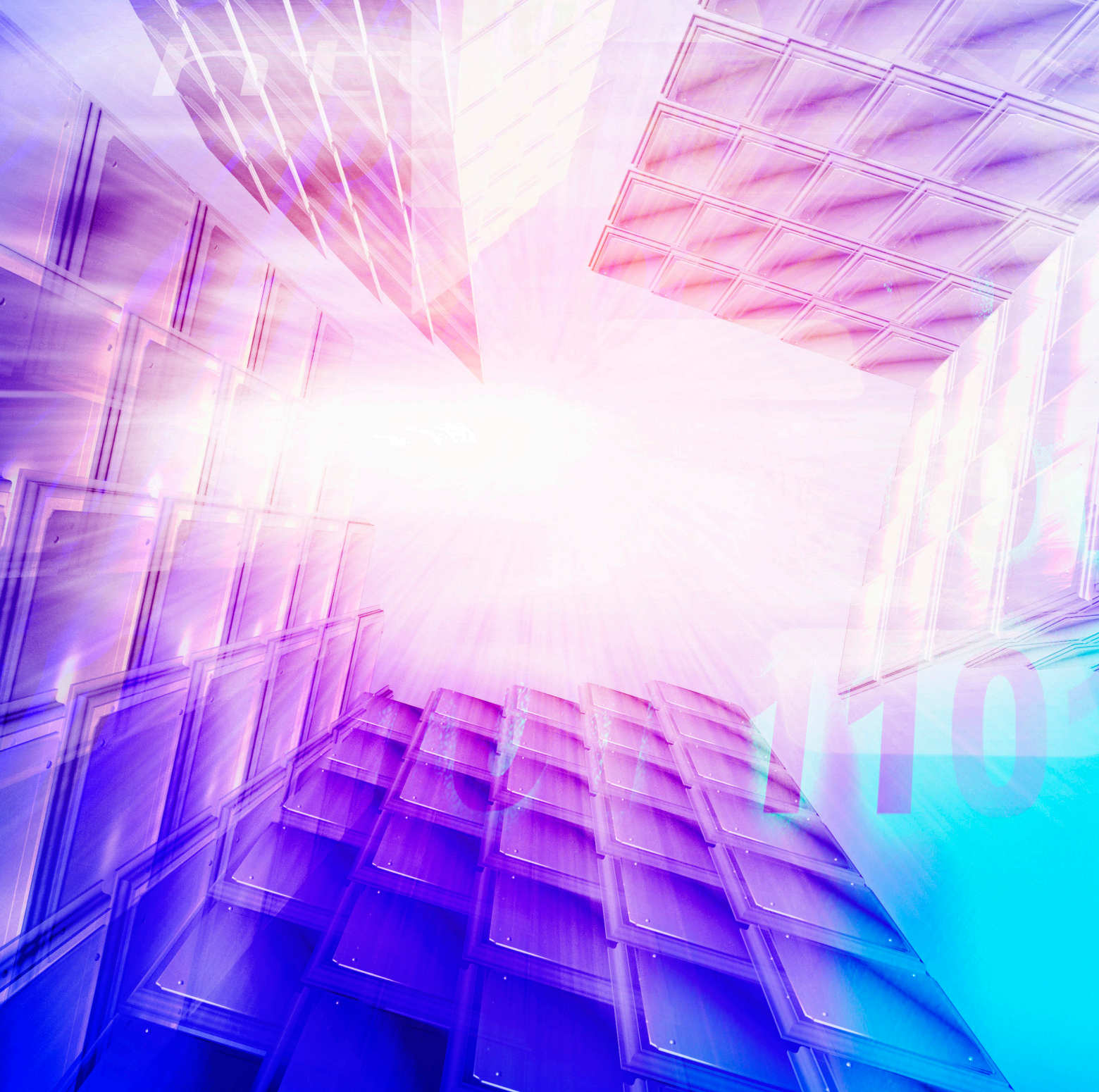
## RAMCloud Overview

RAMCloud is most likely to be used in data centers where large numbers of servers are divided into two main categories: application servers, implementing application logic (such as generating Web pages and enforcing business rules), and storage servers, providing longer-term shared storage for the application servers. Data centers typically support numerous applications, ranging from small ones using only a fraction of an application server to large ones with thousands of dedicated application and storage servers.

RAMCloud represents a new way of organizing storage servers in such a system. Two key attributes differentiate RAMCloud from other storage systems: First, all information is kept in DRAM at all times. RAMCloud is not a cache, as with memcached,[18] and data is not stored on an I/O device, as with flash memory. DRAM is the permanent home for data, with disk used only for backup. Second, RAMCloud must scale automatically to thousands of storage servers; applications see a single storage system independent of the actual number of storage servers.

Information stored in RAMCloud must be as durable as if it was stored on disk, and failure of a single storage server must not be allowed to result in data loss or more than a few seconds of unavailability. Techniques for achieving this level of durability and availability are discussed later in the article.

Keeping all data in DRAM will allow RAMCloud to achieve performance 100x to 1,000x better than today's highest-performing disk-based storage systems:

▸ It should be possible to achieve access latencies of 5µs–10µs measured end-to-end for a process running in an application server to read a few hundred bytes of data over the network from a storage server in the same data center. In comparison, today's systems typically require 0.5ms–10ms, depending on whether data is cached in the server's memory or must be fetched from disk; and

▸ A single multi-core storage server should be able to service at least one million small read requests per second.[a] In comparison, a disk-based system running on a comparable machine with a few disks and a main-memory cache can service 1,000–10,000 requests per second, depending on configuration and cache hit rates.[b]

These goals represent what is possible, but achieving them is by no means guaranteed; later, the section on research issues discusses some of the obstacles that must be overcome.

Table 1 outlines a RAMCloud configuration feasible today, assuming 24GB of DRAM on each server, the most cost-effective configuration we could find; memory prices rise dramatically for larger memory sizes. With 2,000 servers, the configuration offers 48TB of storage at $65/GB. With additional servers, it should be possible to build RAMClouds with capacities up to a few hundred terabytes today. By 2020, assuming continued improvement in DRAM technology, it will be possible to build RAMClouds with capacities of 1PB–10PB for $6/GB.

RAMCloud is already practical for

a variety of applications; for example, Table 2 estimates that customer data for a large-scale online retailer or airline could be stored in RAMCloud for a few hundred thousand dollars. Such applications are unlikely to be the first to use RAMCloud but illustrate that many applications that have historically been considered "large" are easily accommodated by RAMCloud. As of August 2009, all non-image data for Facebook occupied approximately 260TB,[15] which is probably near the upper limit of practicality for RAMCloud today.

It is not yet practical to use RAMCloud for large-scale storage of media like videos, photos, and songs; these objects make better use of disks because of their large size. However, RAMCloud is practical for almost all other online data today, and future improvements in DRAM technology may make RAMCloud attractive for media within a few years.

## Motivation

The RAMCloud concept can be motivated from several different angles, with the most interesting that RAMCloud may enable a new class of data-intensive applications. To understand this potential, consider the two styles of applications in Figure 1. The traditional approach (Figure 1a) is for an application to be delivered by loading its code, along with all of its data, into the main memory of a single machine. This allows the application to access its data at memory speeds, thereby enabling a variety of complex data manipulations but limits the scale of the application to the capacity of one machine.

Over the past 10 years, a new approach to application delivery has emerged, driven by large-scale Web applications serving millions of users. In the Web approach (Figure 1b) the code and the data for an application are kept on separate machines in a data center. The application servers use a stateless

---

a For evidence that a server can handle one million requests per second see Dobrescu et al.,[12] where a single multi-core server was able to receive and retransmit up to 20 million packets per second with a small amount of processing per packet.

b This calculation assumes that disks can service approximately 100 random requests per second and that individual servers hold 10 disks, resulting in total server throughput of 1,000 requests per second if every request involves disk I/O. If disk storage is supplemented with DRAM-based caches on the servers, then a 90% cache hit rate will produce total throughput of 10,000 requests per second.

**Table 1. Example RAMCloud configurations using commodity server technology available today and technology expected by 2020.**

Total server cost is based on list prices, does not include networking infrastructure or racks, and assumes a single copy of data in DRAM, as described in the section on durability and availability. The 2020 estimates assume the cost per server remains approximately the same, but memory capacity per server increases 10x (about two generations of DRAM technology).

| | 2010 | 2020 |
|---|---|---|
| # servers | 2,000 | 4,000 |
| Capacity/server | 24GB | 256GB |
| Total capacity | 48TB | 1PB |
| Total server cost | $3.1M | $6M |
| Cost/GB | $65 | $6 |
| Total ops/sec. | $2 \times 10^9$ | $4 \times 10^9$ |

**Table 2. Estimates of total storage capacity needed for one year's worth of customer data of a hypothetical online retailer and a hypothetical airline.**

Total requirements for each application are no more than a few terabytes, which would fit in a modest-size RAMCloud. The last line of each table (RAMCloud cost) estimates the purchase cost for RAMCloud servers using data from Table 1.

| Online Retailer | |
|---|---|
| Revenue/year: | $16B |
| Average order size: | $40 |
| Orders/year: | 400M |
| Data/order: | 1,000B–10,000B |
| Order data/year: | 400GB–4.0TB |
| RAMCloud cost: | $26K–$260K |

| Airline Reservations | |
|---|---|
| Flights/day | 4,000 |
| Passengers/flight: | 150 |
| Passenger flights/year: | 220M |
| Data/passenger flight: | 1,000B–10,000B |
| Passenger data/year: | 220GB–2.2TB |
| RAMCloud cost: | $14.3K–$143K |

approach, storing only the data for the current request and maintaining little or no state between browser interactions; data must be fetched from storage servers for each browser request. This approach allows applications to scale to thousands of application servers and thousands of storage servers.

Unfortunately, the large-scale architecture of Figure 1b increases the latency of data access by four to five orders of magnitude relative to the single-machine architecture, increasing the application's complexity and limiting its functionality. For example, when Facebook receives an HTTP request for a Web page, the application server makes an average of 130 internal requests for data (inside the Facebook site) as part of generating the HTML for the page,[15] and the requests must typically be issued sequentially. The cumulative latency of these requests is one of the limiting factors in overall response time to users, so considerable developer effort is expended to minimize the number and size of requests. Features requiring more requests than this are not feasible. Amazon has reported similar results, with 100–200 internal requests to generate HTML for each page.[9] These limitations rule out entire classes of algorithms (such as those traversing large graphs).

If the two architectures in Figure 1 are compared based on how many distinct (randomly accessed) small objects each architecture can access per second, the thousands of servers in Figure 1b can support only about the same aggregate access rate as the single machine in Figure 1a. The "scalable" architecture of Figure 1b has scaled total compute power and total storage capacity but has not significantly scaled total data access rate.

The difficulty of scaling data access rates helps explain the recent rise in popularity of the MapReduce paradigm for applications. MapReduce[8] organizes large-scale applications as a series of parallel stages where data is accessed sequentially in large blocks. Sequential access eliminates the latency issue and provides much higher data access rates; as a result, MapReduce has made it possible to solve many large-scale problems efficiently. However, its insistence on sequential data access makes MapReduce difficult to use for applications requiring random access to data.

RAMCloud aims to combine the best of both worlds represented in Figure 1—retaining the scalability of Web applications while reducing data access latency close to that of traditional applications. If it can, it will enable a

new breed of data-centric applications that access information 100x–1,000x more intensively than has ever been possible. It is difficult to identify specific applications that will take full advantage of RAMCloud, since they are not feasible today, but a possible example is an application that would enable massive collaboration at the level of crowds. Another is an online application for statistical machine translation that must iteratively process large graphical language models[4]; in general, RAMCloud will enable graph algorithms at a scale never before possible. Once RAMCloud becomes available, many application developers will discover exciting ways to take advantage of the combination of scale and latency RAMCloud can offer.

**Scalable storage for existing applications.** In addition to enabling new applications, RAMCloud will make it easier to build and scale applications similar to those that already exist on the Web. Creating large-scale Web applications is difficult today due to the lack of a scalable storage system. Virtually all Web applications start out using relational databases for storage, but, as they grow, quickly discover that a single relational database cannot meet their throughput requirements. Each large site undergoes a series of massive revisions that introduce ad hoc techniques to scale its storage system (such as partitioning data among multiple databases). They work for a while, but scalability issues return when the site reaches a new level of scale or a new feature is introduced, requiring yet more special-purpose techniques. Common lore is that every order-of-magnitude increase in a site's traffic requires a major redesign of the site's storage system. For example, as of August 2009, the storage system for Facebook included 4,000 MySQL servers. Distribution of data across the instances and consistency between the instances must be handled explicitly by Facebook application code.[15] Even so, the database servers were incapable of meeting Facebook's throughput requirements by themselves, so Facebook also employed 2,000 memcached servers that cached recent query results in key-value stores kept in main memory. Unfortunately, consistency between the memcached and MySQL

## Figure 1. Traditional applications and scalable Web applications compared.

In traditional applications (a) the application's data structures reside in memory on the same machine that contains application logic and user-interface code, allowing the application to access its data at main-memory speeds. In scalable Web applications (b) the data is stored on servers separate from the user interface and application logic, resulting in much higher latencies for an application to access its data, ranging from 0.5ms (for data cached in the storage server's memory) to 10ms or more (for data that must be read from disk).
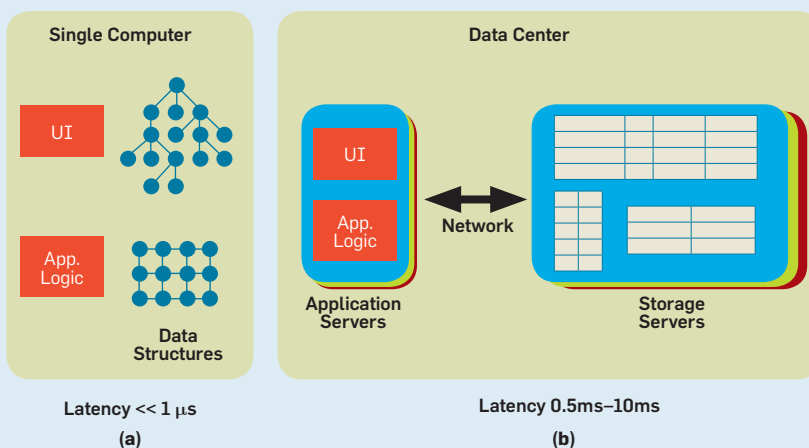
servers must be managed by application software (such as flushing cached values explicitly when the database is updated), adding to application complexity.

Looking to address the scalability limitations of relational databases, numerous new storage systems have appeared in recent years. They are collectively referred to as "NoSQL" because they omit one or more features of a full-fledged relational database in order to achieve higher performance or scalability in certain domains. For example, some NoSQL systems use simpler data models (such as key-value stores), and others offer weaker forms of consistency during updates; examples of NoSQL systems include Bigtable,[5] Dynamo,[9] and PNUTS.[7] Unfortunately, NoSQL systems tend to be less general-purpose than relational databases, and their performance is still limited by disk speed.

One motivation for RAMCloud is to provide a general-purpose storage system that scales far beyond existing systems, so application developers need not resort to specialized approaches (such as NoSQL systems). Ideally, RAMCloud should offer a simple model that is easy to use for new applications and also provide scalable performance that allows applications to grow without constant restructuring.

**Technology trends.** The final motivation for RAMCloud comes from disk technology evolution. Disk capacity has increased more than 10,000-fold since the mid-1980s and seems likely to continue increasing in the future (see Table 3). Unfortunately, the access rate to information on disk has improved much more slowly; the transfer rate for large blocks has improved "only" 50-fold, and seek time and rotational latency have improved by only a factor of two.

As a result of this uneven evolution, the role of disks must inevitably become more archival; it simply isn't possible to frequently access information on disk, so frequently accessed information must be kept in memory. Table 3 illustrates this in two ways. First, it computes the capacity/bandwidth ratio; if the disk is filled with blocks of a particular size, how often can each block be accessed, assuming random accesses? In the mid-1980s, 1KB records could be accessed on average approximately every 10 minutes; with today's disks, each record can be accessed only about six times per year on average, a rate that will drop with each future improvement in disk capacity. Larger blocks allow more frequent access, but, even in the best case, data on disk can be accessed only 1/300th as frequently as it was in the mid-1980s.

One possible solution is to reduce disk utilization; if only half the disk is used, the blocks in use can be accessed twice as frequently. As the required access rate increases, disk utilization must drop; for example, as of late 2009, Facebook could utilize only 10% of its disk capacity. However, reducing disk utilization increases the cost per usable bit; eventually, a crossover point is reached where the cost/bit of disk is no better than DRAM. The access rate corresponding to this crossover point is called "Jim Gray's Rule"[14]; it has increased by a factor of 360x since the mid-1980s (see Table 3), meaning it

**Table 3. Disk technology available in 2009 versus 25 years previously, based on typical personal computer disks.**

Capacity/bandwidth measures how long it takes to read the entire disk, assuming accesses in random order to blocks of a particular size. Capacity/bandwidth also indicates how frequently each block can be accessed on average, assuming the disk is filled with blocks of the same size. For large blocks (>10MB) capacity/bandwidth is limited by the disk transfer rate and for small blocks by latency. The last line (Jim Gray's Rule) assumes disk utilization is reduced to allow more frequent accesses to a smaller number of records, using the approach of Gray and Putzolu[14] to calculate the access rate at which memory becomes cheaper than disk; for example, if a 1KB record is accessed at least once every 30 hours, it is not only faster to store it in memory than on disk but cheaper as well; this access rate allows only 2% of the disk space to be utilized.

|  | Mid-1980s | 2009 | Improvement |
|---|---|---|---|
| Disk capacity | 30MB | 500GB | 16,667x |
| Maximum transfer rate | 2MB/sec. | 100MB/sec. | 50x |
| Latency (seek + rotate) | 20ms | 10ms | 2x |
| Capacity/bandwidth (large blocks) | 15s | 5,000s | 333x *worse* |
| Capacity/bandwidth (1KB blocks) | 600s | 58 days | 8,333x *worse* |
| Jim Gray's Rule [12] (1KB blocks) | 5 min. | 30 hours | 360x *worse* |

**Figure 2. Which storage technology has the lowest total cost of ownership over three years, given an application's requirements for data-set size and query rate?**



Reproduced from Andersen et al.[1]; cost includes servers and energy use.
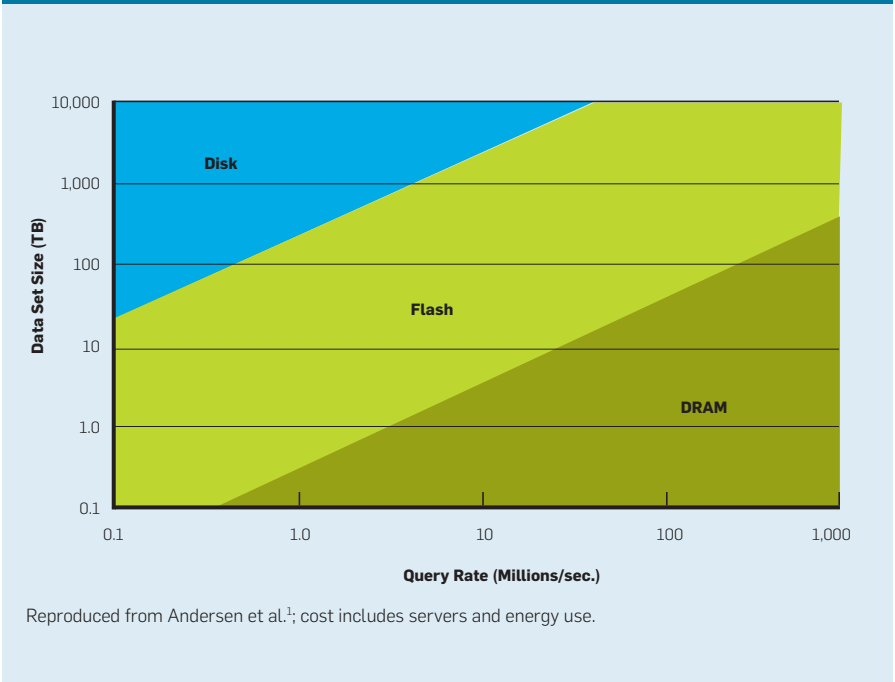
makes economic sense to keep more and more data in memory.

**Caching.** Historically, software engineers have viewed caching as the answer to the problems of disk latency; if most accesses are made to a small subset of the disk blocks, high performance can be achieved by keeping the most frequently accessed blocks in DRAM. In the ideal case, a system with caching offers DRAM-like performance with disk-like cost.

However, the trends outlined in Table 3 dilute the benefits of caching by requiring a larger and larger fraction of data to be kept in DRAM. Furthermore, some new Web applications (such as Facebook) appear to have little or no locality due to complex linkages between data (such as friendships in Facebook). As of August 2009, approximately 25% of all online data for Facebook was kept in main memory on memcached servers at any given point in time, providing a hit rate of 96.5%. Counting additional caches on the database servers, the total amount of memory used by the storage system is approximately 75% of the total size of the data (excluding images). Thus RAMCloud would increase memory use for Facebook by only about one-third.

In addition, the 1,000x gap in access time between DRAM and disk means a cache must have exceptionally high hit rates to avoid significant performance penalties; even a 1% miss ratio for a DRAM cache costs a factor of 10x in performance. A caching approach makes the deceptive suggestion that "a few cache misses are OK," luring programmers into configurations where system performance may be poor.

For these reasons, future caches will have to be so large they will provide little cost benefit while still introducing significant performance risk. RAMCloud may cost slightly more than caching systems but will provide guaranteed performance independent of access patterns or locality.

**Flash memory instead of DRAM?** Flash memory is an alternative storage technology with lower latency than disk; it is most commonly used today in cameras and media players but receives increasing attention for general-purpose online storage.[1] RAMCloud could be constructed with flash memory as the primary storage tech-

nology instead of DRAM (FlashCloud) and would be cheaper and consume less energy than a DRAM-based approach. Nonetheless, a DRAM-based implementation is attractive because it offers higher performance.

Latency is the primary advantage of DRAM over flash memory. Flash devices have read latencies as low as 20µs–50µs but are typically packaged as I/O devices, adding latency for device drivers and interrupt handlers. Write latencies for flash devices are 200µs or more. Overall, RAMCloud is likely to have latency 5x–10x lower than FlashCloud.

RAMCloud also provides higher throughput than FlashCloud, making it attractive even in situations where latency isn't important. Figure 2 (reproduced from Andersen et al.[1]) generalizes Jim Gray's Rule and indicates whether disk, flash, or DRAM is cheapest for a given system, given its requirements in terms of data set size and operations/sec. For high query rates and smaller data set sizes, DRAM is cheapest; for low query rates and large data sets, disk is cheapest; and in the middle ground, flash is cheapest.

Interestingly, the dividing lines in Figure 2 are all shifting upward with time, meaning RAMCloud coverage will increase in the future. To understand this effect, consider the dividing line between flash and DRAM. At this boundary the cost of flash is limited by cost/query/sec, while the cost of DRAM is limited by cost/bit. Thus the boundary moves upward as the cost/bit of DRAM improves and moves to the right as the cost/query/sec of flash improves. For all three storage technologies, cost/bit is improving much more quickly than cost/query/sec, so all boundaries are moving upward.

The latency of flash memory may improve to match DRAM in the future; in addition, several other emerging memory technologies (such as phase-change memory) may ultimately prove better than DRAM for storage. However, considerable uncertainty characterizes all these technologies; for example, not clear is whether flash memory can scale more than another generation or two. In any case, all these technologies are similar in that they provide fast access to small chunks of data. Most of the implementation techniques developed for RAMCloud (such as replication mechanisms, cluster management, and a systemic approach to latency) will be relevant regardless which technology dominates.

## Research Challenges

Many challenges must be addressed before a practical RAMCloud system can be constructed. This section describes a few of them, along with some possible solutions; a common theme is the impact of latency and scale on a system's abstractions and architecture; for a more detailed discussion, see Ousterhout et al.[19]

**Low-latency remote procedure calls.** Though RPC latencies less than 10µs have been achieved in specialized networks (such as Infiniband and Myrinet), most data centers use networking infrastructure based on Ethernet/IP/TCP, with typical round-trip times of 300µs–500µs. Table 4 lists some of the factors contributing to these high times; improvements in both networking hardware and system software are required to achieve 5µs–10µs latency.

On the hardware side, newer 10Gbit/sec switches offer lower latencies; for example, the Arista 7100S[2] claims latencies of less than 1µs (total delay of

**Table 4. Some factors contributing to high latency for RPCs in today's data centers.**

For each component, the "Delay" column indicates the cost of a single traversal of that component, and "Round Trip" indicates the total effect for an RPC; for example, in a three-tier data center network, packets must traverse five switches or routers in each direction.

| Component | Delay | Round Trip |
|---|---|---|
| Network switch | 10µs–30µs | 100µs–300µs |
| Network interface controller | 2.5µs–32µs | 10µs–128µs |
| Protocol stack | 15µs | 60µs |

10µs round-trip in a large data center), and additional improvements seem possible in the future. Similar improvements are needed in network interface controllers (NICs) as well.

On the software side, it will be necessary to simplify protocol processing and eliminate context switches and layer crossings that contribute to latency. One possible approach is to map NICs directly into the address space of applications, so there is no need to cross into and out of kernel code and dedicate a processor core to polling the NIC to eliminate the overhead of taking interrupts. Using these techniques, we have been able to reduce total round-trip software overheads to about 1µs in simplified "best-case" experiments.

Much of today's networking infrastructure was designed to optimize throughput at the expense of latency. We hope designers of future networks will focus more on latency to enable systems like RAMCloud.

**Durability and availability.** For RAMCloud to be widely adopted it must offer a level of durability and availability at least as good as today's disk-based systems. At minimum, this means a crash of a single server cannot cause data to be lost or affect system availability for more than a few seconds, and a systemic loss of power to a data center cannot result in permanent loss of information.

One possible approach is to replicate each object in the memories of several server machines. It offers the highest performance but also high cost, since DRAM is the greatest overall contributor to system cost, and at least three copies are likely required. It is also vulnerable to power failures, since DRAM contents will be lost.

Another approach is to keep a single copy of information in DRAM, with multiple backup copies on disk. This would reduce cost and protect against power failures. However, if the disk copies must be updated synchronously during write operations, write latencies will increase by three orders of magnitude, losing much of the benefit of RAMCloud.

We are exploring an alternative we call "buffered logging" that uses both disk and memory for backup (see Figure 3). A single copy of each object is stored in DRAM of a master server, and copies are kept on the disks of two or more backup servers, with each server acting as both master and backup. However, the disk copies are not updated synchronously during write operations. Instead, the master server updates its DRAM and forwards log entries to the backup servers where they are stored temporarily in DRAM. The write operation returns as soon as the log entries are written to DRAM in the backups. Each backup server collects log entries into batches that can be written efficiently to a log on disk. Log entries written to disk can be removed from the backups' memories. This approach allows writes to proceed at DRAM speeds and utilizes the full bandwidth of disks for maximum write throughput. The log can be managed using techniques similar to those developed for log-structured file systems.[22]

One potential problem with buffered logging is that recently written data could be lost if the master and all backups lose power simultaneously. The ideal solution is to provision each server with small batteries that keep the server alive after a power failure long enough for it to flush buffered log entries to disk. Google uses such a configuration in its data centers; we hope server manufacturers will make this mechanism widely available.

With only a single copy of data in main memory, RAMCloud must recover quickly after server failures to guarantee continuous service. However, the simplest approach to recovery—transferring the memory image of the failed machine from a disk on a backup over the network to a replacement server—would take approximately 10 minutes,[c] which would be unacceptable for many applications. Fortunately, recovery time can be reduced by taking advantage of the scale of a RAMCloud cluster; for example, backup data can be scattered across hundreds of servers in the cluster so it can be read in parallel during recovery. Moreover, the memory image of the failed server can be reconstructed in pieces using hundreds of replacement servers, thereby avoiding a bottleneck at the network interface of a single replacement server. Using a combination of such techniques, recovery time can be reduced to 1–2 seconds. A cluster with 5,000–10,000

**Figure 3. Buffered logging approach to durability.**
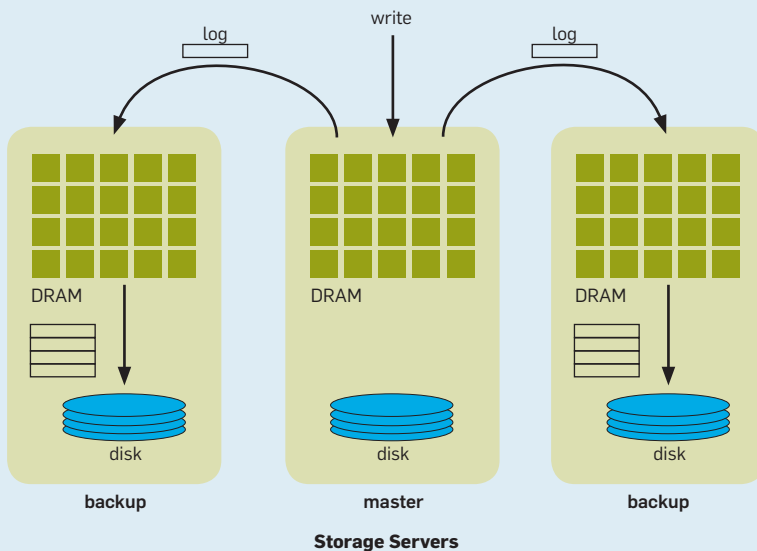


In the buffered-logging approach to durability, a single copy of each object is stored in DRAM. When an object is modified, the changes are logged to two or more other servers. The log entries are initially stored in the DRAM of the backup servers and transferred to disk asynchronously in batches in order to maximize disk bandwidth.

c   This assumes the failed server has 64GB of memory and the backup disk has a transfer rate of 100MB/sec. Even if the disk was infinitely fast, it would take approximately 60 seconds to transfer 64GB data over a 10Gbit/sec network.

servers has enough resources to support multiple simultaneous recoveries. It may even be possible to recover from the loss of an entire rack within a few seconds.

**Cluster management.** RAMCloud must function as a single unified storage system, even though it is actually implemented with thousands of server machines, meaning data placement must be managed automatically by the RAMCloud software. For small tables it is most efficient to store the entire table plus any related indexes on a single server, since it would allow multiple objects to be retrieved from the table with a single request to a single server. However, some tables will eventually be too large or too hot for a single server, in which case the RAMCloud software must automatically partition them among multiple servers. This reconfiguration and others (such as adding and removing servers) must be carried out transparently, without affecting running applications.

Due to the high throughput of its servers, RAMCloud is unlikely to need data replication for performance reasons; replication will be needed only for data durability and availability. If a server becomes overloaded, then some of its tables can be moved to other servers; if a single table overloads the server, then the table can be partitioned to spread the load. Replication for performance will be needed only if the access rate to a single object exceeds the one-million ops/sec throughput of a single server; however, this situation may be so rare it can be handled with special techniques in the affected applications.

**Multi-tenancy.** A single large RAMCloud system must support shared access by hundreds of applications of varying sizes. For example, it should be possible to provide a few gigabytes of storage to a small application for only a few hundred dollars per year, since this storage represents just a small fraction of one server. If an application is suddenly popular, it should be able to scale quickly within its RAMCloud to achieve high performance levels on short notice. Due to its scale, RAMCloud efficiently amortizes the cost of spare capacity over many applications.

In order to support multiple tenants, RAMCloud must provide access

> If RAMCloud succeeds, it will probably displace magnetic disk as the primary storage technology in data centers.

control and security mechanisms that isolate antagonistic applications from one another. RAMCloud may also need to provide mechanisms for performance isolation, so high-workload applications cannot degrade the performance of other applications.

**Data model.** The abstractions provided by RAMCloud will significantly affect its usability, performance, and scalability. Though the relational data model is time-tested and provides a rich and convenient programming interface,[20] its built-in overheads may be incompatible with the low-latency goals of RAMCloud, and no one has yet constructed a system with ACID (atomicity, consistency, isolation, and durability) properties at the scale we envision for RAMCloud. Thus the best data model for RAMCloud is likely to be something simpler (such as a key-value store).[4,9]

**Concurrency, transactions, consistency.** Another question is how much support RAMCloud should provide for transactions; for example, should it support atomic updates to multiple objects? One potential benefit of RAMCloud's extremely low latency is that it might enable a higher level of consistency/atomicity at larger system scale; if transactions execute quickly, then conflicting transactions are less likely to overlap, which should reduce the costs of preventing or resolving conflicts.

### RAMCloud Disadvantages

The most obvious drawbacks of RAMCloud are high cost per bit and high energy use per bit. For both metrics, RAMCloud storage will be 50x–100x worse than a pure disk-based system and 5x–10x worse than a storage system based on flash memory; see Andersen et al.[1] for sample configurations and metrics. A RAMCloud system also requires more floor space in a data center than a system based on disk or flash memory. Thus, if an application must store a large amount of data inexpensively and has a relatively low access rate, RAMCloud is not the best solution.

However, RAMCloud is much more attractive for applications with high throughput requirements. Measured in terms of cost per operation or energy per operation, RAMCloud is 50x–100x more efficient than disk-based sys-

tems and 5x–10x more efficient than systems based on flash memory.[1] For systems with high throughput requirements, RAMCloud can provide not just high performance but also energy efficiency.

Another RAMCloud disadvantage is it provides high performance only within a single data center. For applications requiring data replication across data centers, the latency of updates will be dominated by speed-of-light delays between data centers, so RAMCloud will have little or no latency advantage for writes. However, RAMCloud can still offer exceptionally low latency for reads, even with cross-data-center replication.

### Related Work
The role of DRAM in storage systems has steadily increased over recent decades, and many RAMCloud ideas have been explored in other systems; for example, several research experiments in the mid-1980s involved databases stored entirely in main memory.[10,13] Main-memory databases were not widely adopted at the time, perhaps due to their limited capacities, but there has been a resurgence of interest lately, evidenced by such systems as H-store.[16] The latency benefits of optimizing a storage system around DRAM have also been demonstrated in Rio Vista and other projects,[17] and the limitations of disk storage have been noted by many researchers, including Jim Gray, who predicted that data would migrate from disk to RAM.[21]

Many Web-related applications make aggressive use of DRAM; for example, both Google and Yahoo! store their search indices entirely in DRAM. Memcached[18] provides a general-purpose key-value store entirely in DRAM and is widely used to offload back-end database systems; however, memcached makes no durability guarantees, so it must be used as a cache. Google's Bigtable storage system allows entire column families to be loaded into memory where they can be read without disk accesses.[5] The Bigtable project has also explored many of the issues in federating large numbers of storage servers.

Low-latency network communication has been explored extensively, including in research projects (such

**RAMCloud may cost slightly more than caching systems but will provide guaranteed performance independent of access patterns or locality.**

as Chun et al.[6] and Dittia[11]) and industry standards (such as InfiniBand and iWARP). These projects and standards demonstrate the RAMCloud goal of 5μs–10μs RPC is achievable. However, most of the fastest results were achieved in specialized environments. For example, InfiniBand uses a different networking infrastructure from the Ethernet switches common in data centers. Both InfiniBand and iWARP support the remote direct memory access (RDMA) protocol, allowing a client application to issue read and write requests to selected memory regions of a server; the requests are processed on the server entirely in the network interface controller, with no software involvement. Though RDMA provides low latency, its operations are too low-level, forcing many questions (such as how to organize the server's memory) back to the clients, where the solutions require complex, expensive synchronization among clients. For RAMCloud, the best approach is for all RPCs to pass through application-specific software on the server, possibly making individual RPCs slower but allowing higher-level operations that improve overall system performance.

### Potential Impact
If the RAMCloud concept is shown to be practical and is widely used, it could have broad impact across the field of computing. The most important will be to enable a new class of applications that access large data sets at a very high rate. A RAMCloud containing 1,000–10,000 storage servers will support data sets of $10^{14}$–$10^{15}$B (100TB–1PB) being accessed by tens of thousands of application servers at a total rate of $10^9$–$10^{10}$ requests/sec. The storage model for RAMCloud is "flat," meaning all objects can be accessed quickly; unlike disk-based systems, performance is not affected by data placement. This model should make RAMCloud particularly attractive for graph algorithms operating at very large scale, where accesses are random and unpredictable.

RAMCloud will accelerate adoption of cloud computing. Although cloud computing offers many advantages, it is limited by the absence of scalable storage systems.[3] As a result, developers find it difficult to create applica-

tions that reap the full benefit of cloud computing. RAMCloud will make it easier to develop such applications and eliminate many of the problems associated with scaling them.

RAMCloud could also have a significant effect on network infrastructure. In the past, the design of networks focused primarily on bandwidth, and it has been common practice to sacrifice latency to improve bandwidth. RAMCloud will require extremely low network latency; we hope therefore to see a change of mind-set, with latency becoming a key area where vendors compete for superiority. Latency-driven network design will have a far-reaching effect; for example, networking vendors today are under pressure to incorporate deep packet buffers to avoid dropped packets, causing problems for higher-level protocols (such as TCP). However, these deep buffers can result in huge delays (tens of milliseconds). A low-latency approach would require elimination of almost all buffering, possibly requiring new strategies for handling packet loss in higher-level protocols.

RAMCloud is likely to influence the design and management of data centers, in part through the cloud computing and networking changes discussed earlier; RAMCloud will require data-center managers to think much more about latency than they do today. In addition, if RAMCloud succeeds, it will probably displace magnetic disk as the primary storage technology in data centers. RAMCloud may also lay the groundwork that enables adoption of alternative storage technologies (such as flash memory and phase-change memory).

RAMCloud will also encourage new approaches to server architecture. One change we hope to see is new power-management techniques (such as battery backup and supercapacitors), allowing servers to continue operation for a fraction of a second after power failure in order to flush volatile information to secondary storage. RAMCloud may also enable a spectrum of server designs representing trade-offs among speed, memory capacity, and power consumption. A RAMCloud cluster could also contain several different kinds of servers and automatically adjust system configuration to match access patterns with server characteristics; for example, tables with relatively low access rates could be migrated to servers with processors that are slower but more energy efficient.

## Conclusion

In the future, technology trends and application requirements will dictate a larger and larger fraction of online data be kept in DRAM. We have argued here that the best long-term solution for many applications may be a radical approach where all data is kept in DRAM all the time. The two most important aspects of RAMCloud are its extremely low latency and its ability to aggregate the resources of large numbers of commodity servers. This combination of latency and scale will enable creation of many new data-intensive applications and simplify construction of existing large-scale applications. In addition, RAMCloud will provide an attractive substrate for cloud-computing environments.

Many challenging issues must still be addressed before a practical RAMCloud can be constructed. Toward this end, we have begun a research project at Stanford University to build a RAMCloud system. Over the next few years, we hope to answer some of the research questions about how to build an efficient and reliable RAMCloud, as well as observe the effect of RAMCloud on application development.

## Acknowledgments

Ⅽ

### References
1. Andersen, D., Franklin, J., Kaminsky, M. et al. FAWN: A fast array of wimpy nodes. In *Proceedings of the 22nd Symposium on Operating Systems Principles* (Big Sky, MT, Oct. 11–14). ACM Press, New York, 2009, 1–14.
2. Arista Networks. *Arista Networks 7100 Series Switches*; http://www.aristanetworks.com/en/7100Series
3. Armbrust, M., Fox, A., Griffith, R. et al. A view of cloud computing. *Commun. ACM 53*, 4 (Apr. 2010), 50–58.
4. Brants, T., Popat, A.C., Xu, P., Och, F.J., and Dean, J. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (Prague, June 28–30). Association for Computational Linguistics, Stroudsburg, PA, 2007, 858–867.
5. Chang, F., Dean, J, Ghemawat, S. et al. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems 26*, 2 (2008), 4:1–4:26.
6. Chun, B., Mainwaring, A., and Culler, D. Virtual network transport protocols for Myrinet. *IEEE Micro 18*, 1 (Jan. 1998), 53–63.
7. Cooper, B., Ramakrishnan, R., Srivastava, U. et al. PNUTS: Yahoo!'s hosted data serving platform. In *Proceedings of the 34th International Conference on Very Large Data Bases* (Auckland, New Zealand, Aug. 23–28, 2008), 1277–1288.
8. Dean, J. and Ghemawat, S. MapReduce: Simplified data processing on large clusters. In *Proceedings of the Sixth USENIX Symposium on Operating Systems Design and Implementation* (San Francisco, Dec. 6–8). USENIX Association, Berkeley, CA, 2004, 137–150.
9. DeCandia, G., Hastorun, D., Jampani, M. et al. Dynamo: Amazon's highly available key-value store. In *Proceedings of the 21st ACM Symposium on Operating Systems Principles* (Stevenson, WA, Oct. 14–17). ACM Press, New York, 205–220.
10. DeWitt, D., Katz, R., Olken, F. et al. Implementation techniques for main memory database systems. In *Proceedings of the ACM SIGMOD Conference* (Boston, June 18–21). ACM Press, New York, 1984, 1–8.
11. Dittia, Z. *Integrated Hardware/Software Design of a High-Performance Network Interface*, Ph.D. dissertation. Washington University in St. Louis, 2001; http://www.arl.wustl.edu/Publications/2000-04/zDittia-2001.pdf
12. Dobrescu, J., Egi, N., Argyraki, K. et al. RouteBricks: Exploiting parallelism to scale software routers. In *Proceedings of the 22nd Symposium on Operating Systems Principles* (Big Sky, MT, Oct. 11–14). ACM Press, New York, 2009, 15–28.
13. Garcia-Molina, H. and Salem, K. Main memory database systems: An overview. *IEEE Transactions on Knowledge and Data Engineering 4*, 6 (Dec. 1992), 509–516.
14. Gray, J. and Putzolu, G.F. The five-minute rule for trading memory for disc accesses and the 10 byte rule for trading memory for CPU time. In *Proceedings of the SIGMOD Conference* (San Francisco, May 27–29). ACM Press, New York, 1987, 395–398.
15. Johnson, R. and Rothschild, J. Personal communications (Mar. 24, 2009 and Aug. 20, 2009).
16. Kallman, R., Kimura, H., Natkins, J. et al. H-store: A high-performance distributed main memory transaction processing system. In *Proceedings of the 34th International Conference on Very Large Data Bases* (Auckland, New Zealand, Aug. 23–28, 2008), 1496–1499.
17. Lowell, D. and Chen, P. Free transactions with Rio Vista. In *Proceedings of the 16th ACM Symposium on Operating Systems Principles* (Saint-Malo, France, Oct. 5–8). ACM Press, New York, 1997, 92–101.
18. Memcached. A distributed memory object caching system; http://www.danga.com/memcached/
19. Ousterhout, J., Agrawal, P., Erickson, D., Kozyrakis, C., Leverich, J., Mazières, D., Mitra, S., Narayanan, A., Parulkar, G., Rosenblum, M., Rumble, S., Stratmann, E., and Stutsman, R. The case for RAMClouds: Scalable high-performance storage entirely in DRAM. *SIGOPS Operating Systems Review 43*, 4 (Dec. 2009), 92–105.
20. Ramakrishnan, R. and Gehrke, J. *Database Management Systems, Third Edition.* McGraw-Hill, New York, 2003.
21. Robbins, S. RAM is the new disk... *InfoQ* (June 19, 2008); http://www.infoq.com/news/2008/06/ram-is-disk
22. Rosenblum, M. and Ousterhout, J. The design and implementation of a log-structured file system. *ACM Transactions on Computer Systems 10*, 1 (Feb. 1992), 26–52.

**John Ousterhout, Parag Agrawal, David Erickson, Christos Kozyrakis, Jacob Leverich, David Mazières, Subhasish Mitra, Aravind Narayanan, Diego Ongaro, Guru Parulkar, Mendel Rosenblum, Stephen M. Rumble, Eric Stratmann,** and **Ryan Stutsman** are all affiliated with the Department of Computer Science of Stanford University, Stanford, CA.