Transaction Processing With General-Purpose Servers

How Solid Data File Caching Technology Delivers Real-Time Transaction and Messaging Performance

May 2002

by Solid Data Systems, Inc.

SolidData®

Table of Contents

Executive Summary

We live in an age of rapidly expanding global commerce and communications. For example, roughly half the world's 900 million total users of wireless devices currently use text messaging to generate 55 billion text messages per month, and adoption is growing explosively. Merrill Lynch projects that by 2005, total message traffic worldwide will exceed 400 billion messages per month as customers use their mobile devices for everything from checking train schedules to purchasing their tickets. Likewise, the Internet continues to drive e-commerce in industries such as banking and securities. The Securities Industry Association (SIA) reports that average daily stock trades grew from 150 million transactions in 1995 to 350 million in 1999 – a trend that continues unabated, in transactions and total volume.

Driven by ever-greater pressure to outdo competitors in functionality and features, service providers are simultaneously forced by ever-lower consumer prices to keep expenses low. As the workload from new services and expanding customer bases grows, so does the need for providers of messaging and transaction services to build an infrastructure capable of supporting it – in real time and cost-effectively.

Messaging and transaction processing inherently require data persistence, high availability, scalability and performance. Not only that, systems must constantly handle large amounts of random data with heavy peak loads, a formidable performance challenge in itself. Historically, efforts to address these issues have been less than successful because the available tools have lacked one or more of these key capabilities.

File cache, using Solid Data's solid-state disk (SSD) technology, offers a new, on-target approach to these issues. Already delivering significant benefits to current users, it opens up new possibilities for still better performance in the future. The goal of this white paper is to show application architects and IT staff how to take advantage of these capabilities.

Messaging and Transactions — The Need For A New Architecture

High-speed transactions were once the sole domain of mainframes. Now, however, as electronic commerce is handled at "transaction aggregation points" on global networks, transaction workloads are moving increasingly to midrange Unix and high-end Windows 2000 systems using client-server, distributed computing architectures.

Abundant developer populations, scalable distributed architecture, and lower cost of ownership make these systems the dominant choice for communications, financial services, and many database applications. Unfortunately, while they offer many advantages over centralized mainframe computing, they are not well suited to the highly variable, high-volume, random-access nature of messaging and transactions. For example, the volume of messages at a wireless Short Message Service (SMS) gateway can exceed 40 million per day, most handled during a few peak periods.

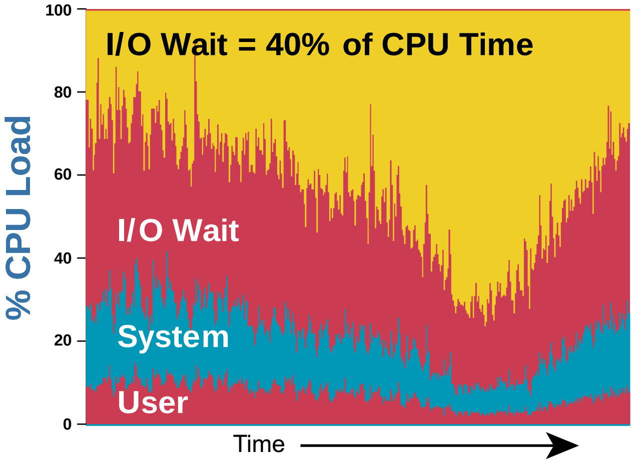


Figure 1. CPU Utilization

Source: Solid Data measurement of large email system workload over 24-hour period

Under such conditions, the limitations of the general-purpose server's I/O and memory subsystems become the system's Achilles Heel. Even under the best of conditions, the slow mechanical access times of disk systems "waste" 70-90 percent of available processing power. The result is highly inefficient processing that requires a larger number of faster, more expensive servers to handle the load.

While it is common to add server memory to solve this problem, memory's susceptibility to data loss during power outages and server crashes makes this approach unsuitable for such business-critical transactions as trading, electronic payment, messaging, or mobile-commerce applications.

Thus, in general-purpose Unix and Windows 2000 server architectures, dual requirements for transaction speed and persistence are fundamentally at odds. For these applications, high-speed random access is essential, but it is equally critical to ensure protection of such data as messages and business transactions. This mandates persistent memory.

Also crucial to these applications is high availability – 24x7 operations and worldwide access demand reliability, redundant design, and serviceability. Simple, easy-to-manage infrastructure offers significant management and maintenance-cost advantages over more complex alternatives. Moreover, as the volume of messaging traffic and transactions grows at an explosive rate, it is vital that a system can be scaled upwards to handle larger workloads without disrupting operations.

The ideal solution would include "persistent memory"– storing data in a solid-state device that:

- Provides random access with memory-like speed
- Protects data in the event of power loss
- May be shared among multiple servers
- May be added easily onto conventional servers to deliver order-of-magnitude performance improvement.

As a rule, any transaction-intensive applications that handle "business-critical" data (i.e., data that has large cost associated with its loss) with high peak loads are good candidates for file cache.

File Cache: Using SDRAM technology and an architectural approach called ***file cache***, Solid Data has developed a new, on-target approach to these challenges. Specifically, this approach transforms a general-purpose system to one optimized for transactions and messaging. File cache accelerates transaction performance by providing the following benefits to server operations:

- It provides low-latency, high-speed access similar to main memory
- Data stored in file cache is persistent, i.e., non-volatile
- Access to file cache may be shared among multiple servers
- The interface to file cache, either SCSI or Fibre Channel, follows standard industry protocol, eliminating the need for upgrade as the server operating system is upgraded and simplifying both installation and management.

At a logical level, Solid Data's file cache serves as a ***persistent extension of main memory*** in which high-demand files are stored and protected - hence the name "file cache."

Case Study: File Cache Triples SMS Performance

The rapid growth of the wireless market has required that companies quickly and easily deploy systems that will not only handle the peak performance loads but also scale effectively. Here, a small incremental investment in Solid Data file cache delivers disproportionately large benefits in performance, scalability, and lower cost.

A major system integrator determined that its wireless Short Message Service Center (SMSC) application was initially limited by moderate performance due to its design architecture's dependence on disk subsystems. The integrator more than tripled its online throughput and the speed of its nightly maintenance processes by storing its entire wireless messaging database on mirrored pairs of Solid Data file-cache subsystems. The current SMSC offering uses mirrored Solid Data file cache as a standard component of the architecture to easily meet performance, scaling, availability and reliability SLAs, contributing to the integrator's competitive advantage as one of the world's leading SMS providers.

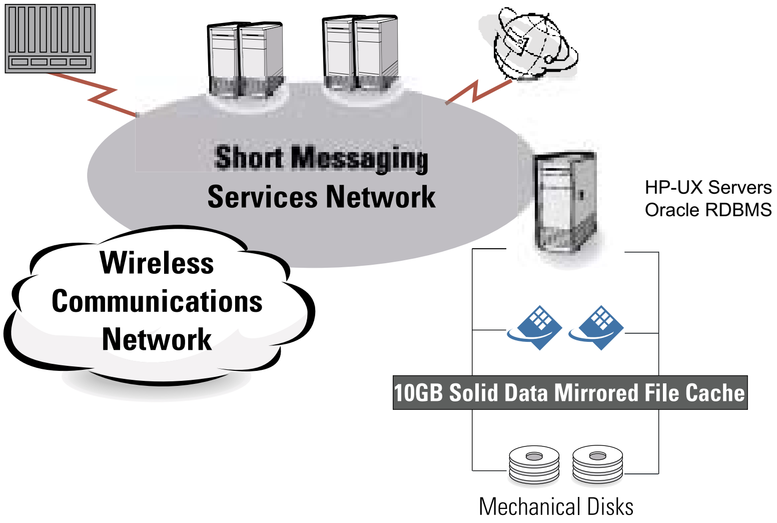


Figure 2. Short Messaging Acceleration

SMS Architecture: File Cache produced dramatic results, improving message capacity per day by over 300% and reducing cost per message by 65%. This improvement was achieved with an incremental investment equal to 15% of the base cost.

Solid Data File Cache — Designed for Transactions and Messaging

Solid Data file cache technology is designed to meet the specialized performance and availability needs of transaction processing and messaging applications, as well as metadata access in file systems and storage networks.

- **Speed:** Solid Data's file-caching technology delivers 2x-10x performance improvement compared to physical disks and RAID arrays. As file cache has virtually no latency compared to disk, small-block reads and writes have little impact on system performance in random-access applications. Furthermore, since the server no longer wastes processing power while waiting (and compensating) for mechanical disk I/O, its full power is made available to accomplish useful work.

- **Persistence:** Unlike volatile RAM, Solid Data's technology delivers high transaction speed while also preserving data in the event of a system crash or power failure. Solid Data file caches use a combination of on-board UPS and physical data retention disk storage to protect the data until power is restored.

- **Shareability:** Solid Data file cache is equipped with dual ports, facilitating high-availability server failover and clustering configurations. With data residing in external, shareable and non-volatile memory, surviving servers (after a failure occurs) are able to access file-cache data and continue to deliver critical applications and services.

- **Scalability:** By eliminating I/O latency, file cache also minimizes CPU "wait" conditions, and greatly improves overall system performance. More performance per server means better and simpler scalability through modular enhancement of system processing power.

- **Reliability:** File-cache data store is based on memory as opposed to mechanical disk, offering significant reliability advantages inherent in solid-state devices.

Advantages Over Traditional Approaches to Transaction Performance Improvement

The performance problems of messaging and transaction-processing applications have proved resistant to traditional solutions. While such approaches as tuning, adding servers or memory, adding disks, striping data, or implementing cache LUNs have delivered significant benefits in many applications, they fail to meet the particular needs of those that require data persistence combined with random access and small-block I/O - needs to which Solid Data file cache is ideally suited.

Tuning: One of the techniques most often used for achieving greater application performance is application tuning. This may take the form of an in-depth code review for custom applications, or, in the case of databases, a Database Administrator will analyze the schema and normalize or de-normalize in an attempt to effect the desired goals. Most applications can benefit marginally from tuning; nominally a 10-20 percent increase in performance is typical. Moreover, since people already on staff do the work, performance gains are often perceived as "free."

However, there are downsides to this method. Tuning requires highly experienced technical specialists, and the process can be very time-consuming, often taking three to six months. Tuning can also introduce new bugs. Finally, it entails a real opportunity cost: how much revenue or productivity will the enterprise lose during the tuning effort? What other problems might the in-house experts solve instead?

Adding Main Memory: Another common technique used to boost application performance is adding main memory in order to give the application program more space. This can reduce the swapping of virtual memory to disk as well as the application's need for disk writes. Adding memory is relatively straightforward and easy to accomplish. If the application is memory-starved, this technique can improve performance by as much as 30-50 percent.

However, this too has disadvantages. One, volatility of server memory, is unacceptable in high-value applications such as messaging and transactions - if the server crashes or loses power, all data in memory will be lost. Another problem is that applications will often require modification to take advantage of more main memory - and in some cases, when source code is not available, these changes are impossible.

Finally, in some cases adding memory does nothing for performance. If the application is very I/O intensive, adding memory may have little or no impact on performance. Before committing to memory addition, it is helpful to analyze the problem with tools such as Solid Data's I/O Dynamics (http://www.soliddata.com/products/iodynamics/).

Adding More CPUs/Servers: Most of today's midrange servers support more than one CPU, and many IT experts resort to adding processors to boost performance. If a multiprocessor server has reached its maximum limit, the next step is adding an additional server. This approach can often double an application's performance, and, if the application is CPU-intensive - e.g. complex solid modeling and mathematical rotation - this may be the only viable solution. Moreover, where multiple CPUs are involved, only multi-threaded applications can scale across the CPUs; hence, many legacy applications do not lend themselves to this method of performance scaling.

Here again, Solid Data's I/O Dynamics can be used to assess I/O behavior characteristics before adding CPUs or servers.

Besides the obvious expense, this approach often entails hidden costs. Adding processors increases maintenance and support fees for the server. Some software licensing is based on MIPS or the number of processors, so adding new processors can lead to higher licensing fees. A new server brings new requirements for floor space, maintenance, power and cooling - often the annual maintenance costs can approach the cost of the server itself.

Adding More Disks/Striping Data: Adding disks or striping data is an approach ideally suited for bandwidth-intensive uses such as streaming video and processing of large images, as commonly found in medical and scientific applications. Database applications with very large data requests may also benefit from this method. Since disks are relatively inexpensive, this solution can be very cost-effective for storage-intensive applications.

However, maintenance costs for disk systems can approach eight times the cost of acquisition. Also, adding more disks reduces MTBF.

Finally, adding disks may reduce average access time (latency), but not enough to eliminate I/O-wait by the CPU, a central issue in transaction-intensive applications. Arrays increase the number of megabytes per second the application can handle (via higher overall transfer rates) but do not change the access time sufficiently to make the system suitable for high transaction rates. The problem: response times of RAID (Redundant Array of Independent Disks) or JBOD (Just a Bunch of Disks) storage are limited by the slow mechanical access times of disks compared to the access speed of solid-state devices.

Thus, for small-block, random-access, transaction-intensive applications, adding more spindles will likely have nominal performance impact. This approach is analogous to adding more lanes on a toll way - additional lanes add to the total capacity of the road, but the actual volume of traffic is limited by the ability of the tollbooths to take payments from the toll way traffic. Once the tollbooth payment limit is reached, more lanes do not speed up traffic flow.

Here, tools like Solid Data's I/O Dynamics can be used to diagnose I/O patterns, determine what files are impacting performance, and suggest which ones might best be moved to file cache.

Cache LUNs (RAID Cache): For many years, cache LUNs have been available under various trade names in larger storage-array systems from EMC, Hitachi, HP and Sun.

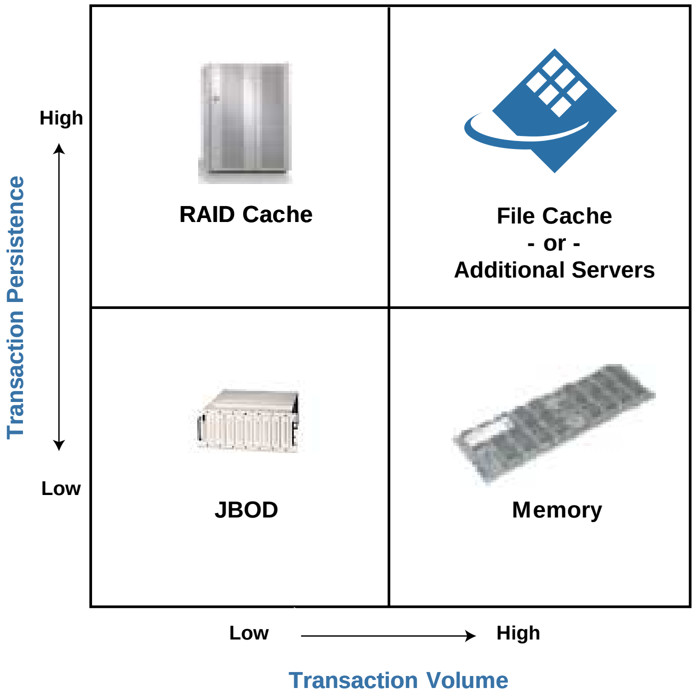
Faster than mechanical disk, a cache LUN is not as fast as a file cache, because the memory it uses is designed to be an intermediate memory location, with rotating disk as the final destination. Thus, along with the data it must include all the information necessary to move the data to its final, persistent location. When transaction rates are high, managing this additional information degrades storage-array performance to such an extent that a separate, external file cache (which simply stores data in memory blocks directly addressable by the host file system) is significantly faster.

A cache LUN consists of two parts: the actual cache memory, and specialized software to reserve part of the cache and control its interface. A high-end storage array may appear to have enough cache memory to support a cache LUN. However, because the amount of intermediate (cache) memory storage is already orders of magnitude smaller than the amount of disk storage, reserving any of this memory for cache LUN normally impacts the performance of the entire storage system. Conversely, removing very high transaction-rate data from RAID arrays allows substantially improved performance from the RAID array on the remaining files.

Costs of cache LUN, however, can be high. RAID cache memory for high-end arrays is very expensive and usually must be purchased in mirrored configurations. In addition, recurring license fees are typically required for the software that controls the cache LUN, thus adding operating expense to capital investment.

Figure 3. Approaches for Improving Transaction Performance

While many techniques are used to improve performance, file cache is the most cost-effective method for achieving high transaction rates without exposure to data loss.



Speed and Persistence: Figure 3 summarizes these considerations in terms of two dimensions: transaction volume (required throughput, or infrastructure speed) and transaction persistence. File cache delivers high speed and high persistence for messaging and transaction processing - without multiplying the number of inefficiently-utilized servers.

Case Study: File Cache Speeds Up Trading

A European software company develops one of the world's leading real-time electronic trading and risk-management software applications, for international electronic markets in financial commodities and energy. By displaying and analyzing real-time information feeds from multiple trading exchanges, the software provides a rich information environment and powerful decision-support tools. It offers advanced functionality for analysis and pricing, trading and risk management, providing its users with the ability to trade simultaneously on multiple electronic marketplaces and to manage massive order flows for automated straight-through processing. In addition, the system contains advanced calculation models for pricing various derivative instruments.

While this system provides good performance on general-purpose servers, its response time to high-volume trading and fast market changes was limited by the disk subsystem's inherent high latency and inability to provide quick responses to trade requests. This resulted in system delays to traders, reducing trading profits during peak trading periods. By utilizing a Solid Data file cache to provide the lowest latency for randomly accessed key database files, the company found that the system not only benefited from improved response times, but also captured additional CPU capacity that had previously been lost to I/O wait conditions.

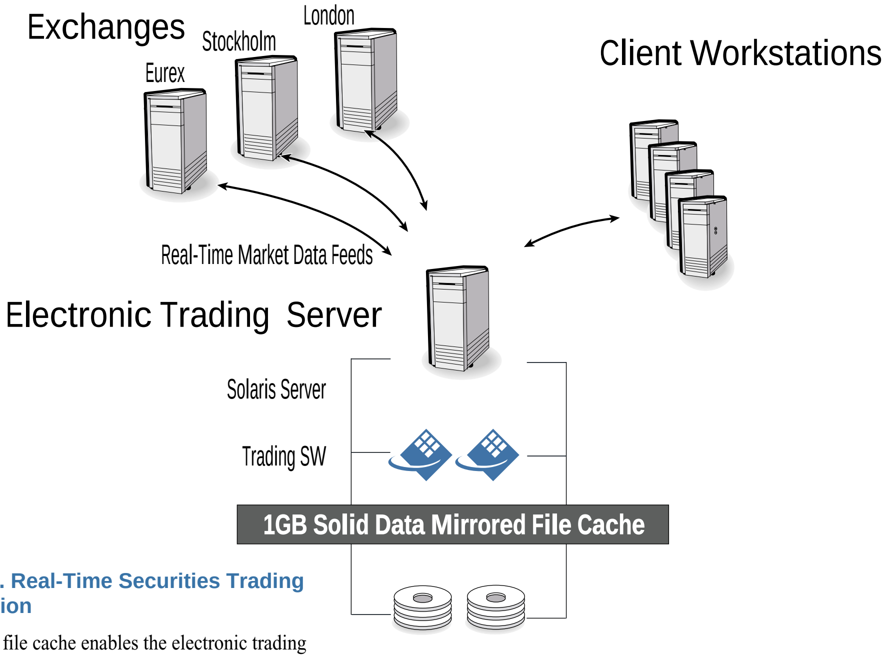

Exchanges
London
Stockholm
Eurex
Client Workstations
Real-Time Market Data Feeds
Electronic Trading Server
Solaris Server
Trading SW
1GB Solid Data Mirrored File Cache

Figure 4. Real-Time Securities Trading Application

Solid Data file cache enables the electronic trading server to deliver real-time market information with powerful functionality.

Optimizing Results With File Cache — Design Guidelines

To achieve best results with file cache, it is often necessary to forego techniques that have been beneficial in more traditional contexts. For example, many legacy programs - applications, file systems, operating systems and device drivers - are built on the assumption that all data resides on mechanical disk drives. To optimize these programs for disk-based data access, software designers have incorporated features that are unnecessary and wasteful when the data is located on the near-zero latency file cache.

For example, read-ahead caching is a useful technique for minimizing the impact of mechanical-disk latencies, but it is counterproductive when applied to data placed in external, persistent file cache. In this situation, read-ahead caching just fills up the I/O channel with unnecessary data traffic, which can greatly reduce the effective performance of a zero-latency, random-access device.

Based on experience with a wide range of applications and environments, Solid Data recommends the following design guidelines:

- Turn off read-caching functions at higher levels in the software stack when accessing data from a file cache.
- Specify Direct I/O (where supported) when mounting file systems that are stored on file cache.
- Place log files and swap files on file cache.
- For transaction-intensive applications that occupy less than 20GB, consider placing the entire application on a file cache.
- Place message queues on file cache, but place large message stores or mailboxes on mechanical disk.
- Move database indexes to file cache, especially when sorts and queries apply multiple indexes or the application generates frequent inserts and deletes.
- Partition a large, complex data file to segregate the most active data into a separate file. Then place the resulting hot file on file cache.
- Consider file cache for file-system metadata and shared name spaces - and for address tables in storage-virtualization architectures.

Solid Data also offers architectural evaluation services and related software tools. These can help solution architects optimize the use of memory and storage devices by making well-informed choices about system configurations and data-access methods.

Summary

Characterized by the need to process large amounts of random data in real time, messaging and transaction-processing systems historically have suffered I/O performance problems. Moreover, because these applications require speed, data persistence, scalability and high availability, as well as cost-effectiveness, traditional solutions have not proven satisfactory.

File cache, using Solid Data's solid-state disk technology, offers a new approach that is ideally suited to the needs of these applications. File cache enables enterprises to get the most benefit from their existing equipment while achieving the performance, scalability, and cost advantages necessary for successful deployments. Already delivering significant benefits to providers of messaging and transaction-processing services, file cache has even greater potential benefits for a wide range of growing applicaitons.

SolidData®