

Speed and Persistence for Real-Time Transactions

by TimesTen and Solid Data Systems

July 2002

TimesTen

 SolidData®

Table of Contents

Abstract	1
Who Needs Speed and Persistence	2
The Reference Architecture	3
Benchmark Results	4
Shared vs. Dedicated RAID Cache	6
Alternative Configurations	8
Performance vs. Cost and Risk	9
Application Examples	11
Summary	12

Abstract

Telecommunications carriers, financial service providers and other enterprises are developing new revenue-generating and cost-saving applications that drive rapidly increasing volumes of electronic messages and transactions. These real-time applications demand very high throughput, and many of them also require persistent capture of transaction data. This paper outlines a high-performance alternative for application architects and system integrators who are designing high-volume transaction and messaging services. By configuring the application with a real-time event processing system and a persistent file cache using solid-state disk, a solution architect or integrator can deliver unparalleled performance and reliability with an efficient, scalable and cost-effective infrastructure.

Who needs Speed and Persistence?

The increasing popularity of Internet computing and the proliferation of new mobile services are driving profound changes in systems infrastructures. In these environments, transaction processing is less about humans sitting behind forms, and more about applications exchanging messages with other applications, triggering sequences of lookups, updates, and follow-on messages. Thus, a significant portion of the world is now initiating transactions indirectly and both the volume and the immediacy of transaction processing are increasing. The financial services industry has been processing extreme volumes of real-time message-driven transactions for years, and is also in the midst of a transformation in search of additional value-added services and even more rapid processing.

This paper shows how a combination of software and hardware can deliver transaction rates that meet and exceed the requirements of today's most demanding applications, such as high-volume online trading, mobile text messaging and real-time billing. These are generally message- or event-processing applications, characterized by thousands or tens of thousands of small, system-driven transactions per second. They are outstripping traditional OLTP software and hardware approaches, particularly when applications must process thousands of transactions per second.

Many of these high-speed applications also require persistent capture of critical messages and transaction data - i.e., *durable commits* to persistent storage. The resulting data-capture requirement exceeds the capability of traditional hardware architectures that use mechanical disk drives for persistent storage. Data persistence is vital for applications that process revenue-generating transactions, or that promise reliable message handling as the basis for new services and competitive differentiation.

Enterprises need a new combination of speed and persistence, when transaction volumes are very high and *when every transaction counts*.

The Reference Architecture

The ultimate transaction-server or message-server architecture includes a TimesTen real-time event processing system, with the transaction logs written to Solid Data file cache (Figure 1).

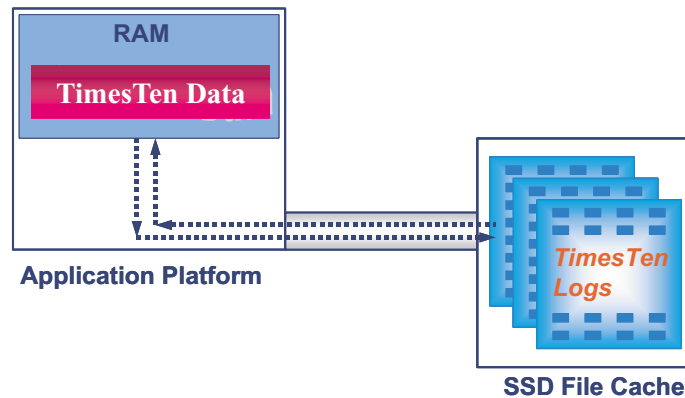


Figure 1. Reference Architecture for Transaction Servers

The TimesTen software achieves very high transaction rates by employing memory-optimized data structures and algorithms, and using non-volatile storage solely for persistence and recovery. When the application requires durable commits, the TimesTen transaction logs must be written to a log file located on persistent media. The reference architecture uses a high-speed, solid-state file cache, because traditional disk-based solutions cannot sustain the required transaction volumes and response times.

File cache, also known as solid-state disk or SSD, is a rack-mount hardware module based on SDRAM memory, combining the speed of main memory with the persistence of disk. Because of its built-in data retention system - which includes an on-board UPS battery back-up system and an embedded disk drive for data recovery - it provides the non-volatile storage required for durable commits in real time, without the mechanical access delays that characterize mechanical disk storage. And since the connection to file cache is standard SCSI or Fibre Channel, file cache is sharable between servers, and easy to implement.

Since the file-cache hardware stores transaction log data on memory chips, rather than rotating disk media, it supports synchronous writes at high data rates -- without the multi-millisecond latencies imposed by mechanical disk storage. And without the inherent volatility of a pure-memory solution.

This reference architecture combines simple software and hardware components to achieve very high data rates - even on write-intensive applications that require immediate, durable commits for all transactions. This is an optimized architecture: fast, robust and cost-effective. It provides a standard, proven, predictable solution that maximizes performance and minimizes risk.

Benchmark Results

Figure 2 shows the performance impact of placing the TimesTen logs on file cache, compared to traditional approaches that place the logs on cached disk-array storage. These tests were conducted using the TimesTen **tptbm** benchmark running on a 2-processor Sun E450 server.

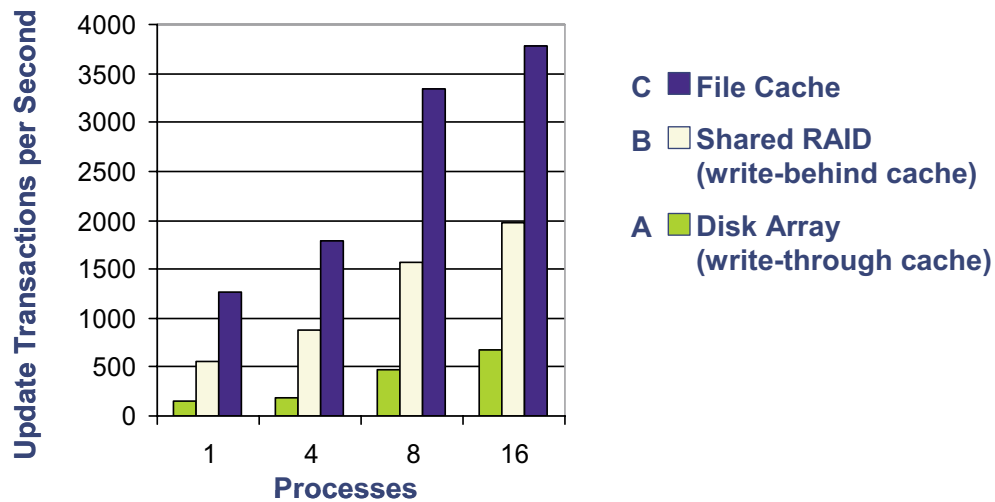


Figure 2. TimesTen Benchmark Throughput (100% Updates)

To ensure durable commits, each update transaction was written to a log file stored on a Solid Data file cache or a Sun T-3 disk array. For these tests, the T-3 array was configured with 256 Mbytes of RAID controller cache, and set up for RAID-1 (mirrored) operation.

Configuration A - based on write-through RAID cache - achieved 148 durable update transactions per second, using the TimesTen **tptbm** benchmark running 1 process, and 476 per second when running 8 processes. In the write-through caching mode, each transaction is written to disk media before the RAID controller reports completion to the server.

Configuration B - using shared write-behind RAID cache - processed 522 durable update transactions per second on one process, and 1570 per second with 8 processes. In the write-behind caching mode, each transaction is acknowledged when received in the RAID cache -before the data is written to persistent disk media. In this test, the T-3 array was shared between TimesTen and another benchmark workload (Solid Data's *I/O Test*), and the test results reflect the impact of sharing the array. A dedicated RAID array performs somewhat better but involves other tradeoffs, as discussed later in this paper.

Configuration C - based on Solid Data file cache - delivered 1,268 durable update transactions per second on one process and 3,341 per second for 8 processes. The file cache combines the speed of DRAM with the persistence of disk - as described for the reference architecture - so completion of each write operation is acknowledged as soon as the data is placed in the file cache.

Comparisons: Over the range from 1 to 8 processes, file-cache-based configuration C was more than 7 times faster than *write-through* cached RAID (configuration A) for durable commits.

The *shared write-behind* RAID cache (configuration B) was more than 3 times faster than the *write-through* RAID (configuration A) when running 1 to 8 processes, but it delivered less than 50% of the throughput achieved with the Solid Data file cache. (Somewhat better results were achieved in these benchmark tests by dedicating the array cache to TimesTen logfiles, rather than sharing it with other workloads. This option - more a theoretical possibility than a practical recommendation for most enterprises - is explored in the following section, followed by a cost/benefit comparison of the alternative solutions, and a couple of illustrative applications.)

The *write-behind cache* configuration relies on the RAID controller's battery to maintain power to the cache for any extended period of line-power outage, or to sustain power to the entire array until the data can be written to disk. These data-protection features will vary depending on the RAID vendor's implementation, and some RAID controller designs will increase the risk of data loss in the event of power outage. Therefore write-behind caching requires the customer to carefully evaluate and test the specific RAID model, to ensure the required degree of data persistence, before it is deployed for durable commit logging.

A major benefit of the reference architecture, using a dedicated Solid Data file cache for durable commit logging, is that it defines a proven and cost-effective solution that does not require evaluation of a wide range of different cached RAID controllers by individual TimesTen customers.

Shared vs. Dedicated RAID Cache

Somewhat better results were achieved for configuration B by dedicating the write-behind array cache to TimesTen logfiles, rather than sharing it with other workloads. This variation of configuration B, using *dedicated* write-behind RAID cache, processed 898 durable update transactions per second on one process, and 2,696 per second with 8 processes. Thus the dedicated write-behind configuration outperformed the write-through RAID (configuration A) by a factor of 5 or 6; but it still delivered only 70% to 80% of the throughput achieved with the Solid Data file cache. In other words, the file-cache solution proved to be 25% to 40% faster than dedicated write-behind RAID cache.

Moreover, most enterprises will question dedicating an entire RAID system so its cache can service the TimesTen log files, especially given the dedicated RAID's slower performance and the necessity of evaluating the risk of data loss for each specific RAID model. When considering RAID, and accepting its costs and risks, enterprises will be inclined to use the asset for additional files and workloads, and that is the reason we have used the test results for a shared-cache deployment in Figure 2 (configuration B).

Figure 3 shows the detailed test results for *write-behind* cached RAID (configuration B). These tests used the TimesTen **tptbm** benchmark running first with the T3 Array *dedicated* to the TimesTen log files. The results are shown as the "no load" data points in Figure 3 for 1, 4, 8 and 16 processes. The **tptbm** benchmark was then repeated while the server was also running Solid Data's **I/O Test** benchmark, i.e., reading and writing separate data files of various sizes to the T3 array. When the external load process file size reached 1GB, the performance of the write-behind RAID configuration had declined by 60%.

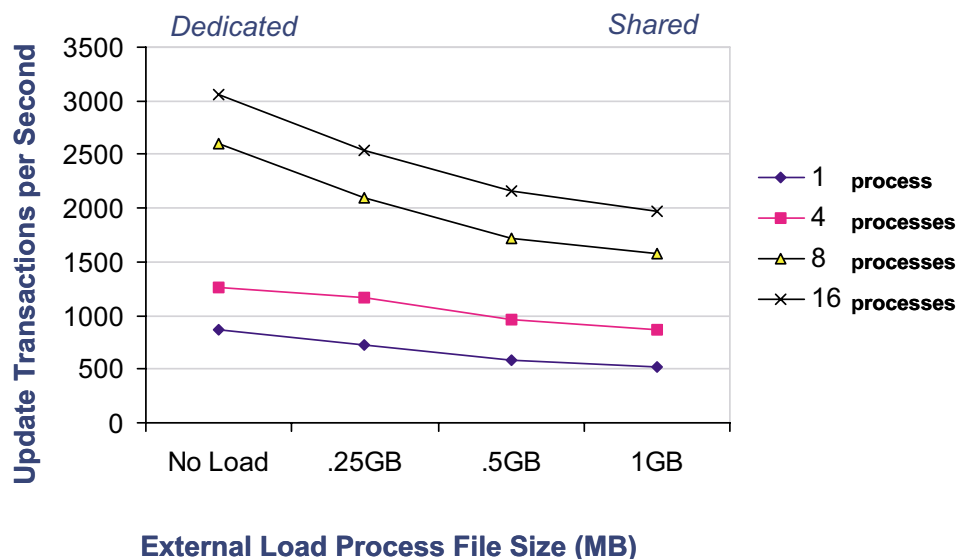


Figure 3. Impact of Sharing a Write-Behind RAID Cache

Interpretation: As the size of the data load exceeded the size of the RAID controller block cache, the controller was forced to write **tptdm** log-file data blocks to disk, to make room for incoming data from the **I/O Test** workload. In effect, this shifted the RAID-cache operating mode from write-behind to write-through, and greatly decreased the TimesTen benchmark throughput. The degree of performance reduction depends on the amount of non-log-file data being written, as well as the size of the RAID controller's block cache.

To avoid this *shared-cache* performance reduction when using an entry-level or midrange RAID device, an enterprise would have to *dedicate* an entire cached RAID system to the TimesTen log files. While this is theoretically a valid approach, it does involve management costs and risks, including the need to evaluate each specific RAID model to ensure performance and persistence. In contrast, the reference architecture - employing Solid Data file cache - is a standard, proven solution. The reference architecture delivers higher performance and is also more cost-effective than many RAID-based configurations, as illustrated in the following section.

Alternative Configurations

Figure 4 compares four alternative configurations, in terms of data placement strategy and key characteristics. Sustained *performance* is shown as a percentage of the demonstrated throughput of the reference architecture (configuration C), in which the TimesTen log files are placed on a dedicated file cache.

Configuration A shows the performance obtained when the TimesTen log files are placed on *write-through* RAID cache - less than 15% of the benchmark throughput of the reference architecture.

Configuration B summarizes the performance impact of placing the TimesTen log files in a *write-behind* RAID cache - either shared with other applications or dedicated to the log files, as discussed above. In the most likely implementation - where the cache is shared with other workloads - write-behind RAID cache delivers less than 50% of the reference architecture's throughput, over the range from 1 to 8 processes in the **tptbm** benchmark.

Configuration	Cache MB	Disk GB	A	B		C
			Disk Array	RAID cache		File Cache
				shared	dedicated	dedicated
Data Placement:						
Log files			RAID, write-through cache	RAID write-behind (block cache)	RAID write-behind	File Cache
Other data files				Disk	Disk	Disk
Characteristics:						
Sustained Performance			< 15%*	< 50%*	< 80%*	100%
Persistence			High	Varies	Varies	High
Cost:						
Cached RAID (Sun A-1000 or T-3 *)	24 to 1,000	72 to 327	\$9K to 39K	9K to 39K	9K to 39K	—
Small JBOD (Sun S-1)	—	32	—	—	\$2K	\$2K
File cache (Solid Data e-100L)	1,000	—	—	—	—	\$8K
Total price:			\$9K to 39K	\$9K - 39K	\$11K - 41K	\$10K

* RAID performance was measured on Sun T-3 array with 256MB of cache, as described in benchmark results. Minimum list price for a new T-3 array at Sun online is \$38,600, configured with 1GB of cache and 327 MB of disk. Disk arrays are available at lower list prices, e.g. \$9,000 for a Sun A-1000 with 24MB of cache and 72GB of disk. However, each RAID model would require separate verification of its performance and cache persistence.

Figure 4. Alternative Configurations Compared

Persistence is also assessed, for each configuration, in comparison with the reference architecture - configuration C - in which the TimesTen log files are placed on Solid Data file cache. On this dimension, write-through RAID cache (configuration A) gets comparable marks, since the RAID controller writes the data to magnetic disk media before acknowledging write completion to the server.

In contrast, the persistence of log file data in a write-behind RAID cache (configuration B) will vary depending on the RAID product chosen, and the way it is configured and managed. As mentioned previously, write-behind RAID caching requires careful evaluation and testing of each specific RAID product model. In contrast, Solid Data file cache - a key component of the reference architecture - is a standard, proven solution that delivers high speed and data persistence across a wide range of server platforms.

Performance vs. Cost and Risk

Figure 4 also illustrates the *costs* of the required storage devices, for each alternative approach. While the cached-RAID performance benchmarks were conducted using a Sun T-3 array with 256MB of cache, the midrange RAID cost estimates are based on the minimum T-3 configuration currently offered by Sun Microsystems - a 327 Mbyte disk array with 1Gbyte of RAID controller cache. To reflect the existence of lower-priced disk arrays at the entry level, the RAID configurations in Figure 4 also include price and capacity information for a Sun A-1000 array.

Configuration A - a disk array with *write-through* cache - is included for completeness, though it is clear that it cannot deliver adequate throughput for high-performance messaging and transaction processing. The cost of this configuration is comparable to the other choices, and it offers good data persistence, but its performance just doesn't make the grade.

Configuration B - for disk arrays with *write-behind* cache - shows a separate JBOD (just a bunch of disks) subsystem for non-logfile data, in the case of the *dedicated* write-behind RAID configuration. The additional cost of the JBOD box is avoided in the *shared* RAID configuration by using the RAID array to store other application files in addition to the TimesTen log files.

Shared RAID: As illustrated for the T-3 array in Figure 3, the performance of the *shared* RAID configuration degrades when the server accesses those other files. Moreover, the performance degradation depends on the size of the additional file accesses, relative to the size of the RAID array's block cache. The A-1000 array, for example, provides only 24 MB of controller cache, so it is unlikely to match the throughput of the tested T-3 array. And, even if we assume that the A-1000 would provide performance comparable to the T-3 array, the low end of the configuration B price range merely matches the pricing of the reference configuration - while introducing significant performance penalties, management complexity and risk.

Dedicated RAID: Figure 4 also illustrates the dedicated-RAID alternative for configuration B. It includes a dedicated RAID array that serves as a write-behind caching subsystem for the TimesTen log files, and a small JBOD subsystem for other application files. While the test results were obtained with a Sun T-3 RAID array, the minimum list price for the T-3 array is \$38,600. This would be an expensive choice, especially when the array is dedicated to caching the small TimesTen log files. Other cached RAID choices are available in the marketplace at lower prices, and the cost analysis in Figure 4 uses a Sun A-1000 array priced at \$9,000. This brings the cost of configuration B into line with the cost of the file-cache alternatives. However, we do not have test results for the A-1000, or for the dozens of other cached RAID products available. And the total cost of configuration B would have to include the overhead costs of evaluating, testing and configuring specific RAID products to be sure that cached data will be protected under all possible power-fail conditions.

Configuration C in Figure 4 represents the reference configuration. It includes a 1GB Solid Data file cache - priced in mid-2002 at \$7,500 - dedicated for caching the TimesTen log files. It also includes a \$2,300 JBOD (just a bunch of disks) subsystem that supplies 36GB of disk storage for other data files. This example configuration uses a Sun JBOD subsystem but other choices for this component would not materially impact the cost comparisons among the alternatives considered here. As the benchmark tests have demonstrated, this configuration delivers maximum performance for message and transaction-processing applications that require durable commits.

Overall Conclusion: RAID-based solutions may be comparable to the file-cache-based reference architecture in terms of purchase price, but they are less effective and predictable in terms of performance. For high-performance operation, RAID-based solutions must dedicate the RAID array's cache for log file caching, and must also be operated in write-behind caching mode - which requires users to carefully evaluate, test and configure specific RAID products to minimize the risk of data loss when persistent logging is required for durable commits. In contrast, the reference architecture using file cache provides a simple, standardized and proven solution for speed and persistence.

Application Examples

Applications that can benefit from a transaction server solution using TimesTen software and Solid Data hardware include telecom industry applications such as prepaid calling services and financial industry applications such as electronic trading and middleware.

Prepaid Calling Service

Prepaid telephone services are data driven - constantly accessing subscriber balances and service profiles. The speed at which this critical subscriber data can be accessed is a key differentiator, determining the scalability and quality of service offered by the prepaid solution.

Packet-based data transfers will generate a huge increase in the amount of billing data generated by 2.5G and 3G networks. This data will need to be received, stored and processed in real-time for all customers.

A combined TimesTen/Solid Data solution - as illustrated in Figure 5 - can provide the ultimate in high-performance application speed and non-volatile persistent data stores, to support current and future convergent billing services with the most cost-effective infrastructure.

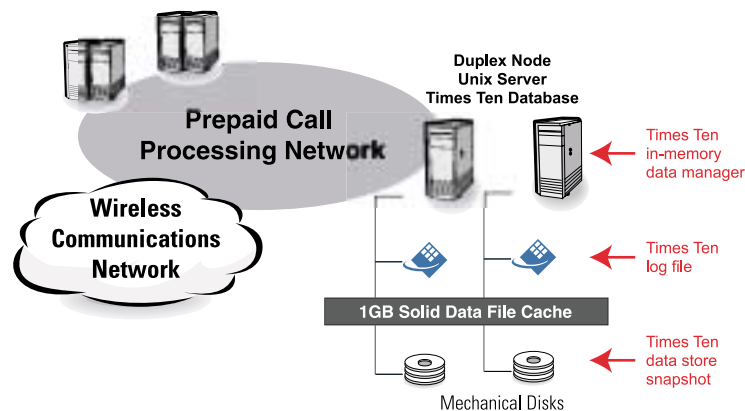


Figure 5. Reference Architecture applied to Prepaid Calling Services

Electronic Trading and Middleware

Large financial trading systems must handle very high volumes - thousands of trades per second - while ensuring persistent capture of every transaction. Also, front-office trading applications are increasingly integrated with back-office systems through messaging middleware, and all these applications require high-volume performance with persistent event logging.

A combined TimesTen/Solid Data solution, by combining high throughput with a persistent data store, enables financial institutions and service providers to deliver powerful business solutions with the most cost-effective infrastructure.

Summary

On-line, real-time applications - no matter what market they address - often have very similar performance and availability requirements. These include high-performance processing of very random transactions, and the need to durably commit each transaction to non-volatile, persistent storage. There are several ways to address these requirements, and these approaches have different costs and benefits.

The combination of TimesTen software and Solid Data hardware delivers the ultimate architecture for speed and persistence for applications that require durable commits of transactions. TimesTen benchmark tests demonstrate that this reference architecture delivers seven to eight times faster throughput on durable commits, compared with rotating disk (RAID with write-through cache). While cached disk arrays may achieve up to 80% of the reference architecture's performance - if an array is *dedicated* to the log files and operated in *write-behind* caching mode - those alternative solutions are typically more expensive, more risky and more complex to manage. The reference architecture provides a simple, standardized and proven solution for speed and persistence.

When every transaction counts, application architects and administrators now have a way to deliver the ultimate high-performance solution for real-time transaction and messaging servers with TimesTen software and Solid Data hardware.

About TimesTen

TimesTen, Inc. provides software for real-time event processing - a fundamental requirement of applications used by Wall Street firms and global telecom providers. Systems built with TimesTen inside are instantly responsive, highly reliable, and able to process massive transaction volumes. Deutsche Bank, Sprint, Nokia and Cisco are among the customer who rely on TimesTen's software and expertise to create a unique advantage in a rapidly changing market.

For more information, see www.timesten.com.

Corporate Headquarters:

1991 Landings Drive
Mountain View, CA 94043
USA
Tel: +1.800.970.1248 ° +1.650.526.5100
Fax: +1.650.526.5199

EMEA Headquarters:

500 Chiswick High Road
London, W4 5RG
UK
Tel: +44 (0) 20 8956 2532
Fax: +44 (0) 20 8956 2536

About Solid Data

Solid Data Systems is the leading global provider of file-caching systems that multiply the throughput of business-critical, transaction-intensive applications. Solid Data's proprietary file-cache products, based on solid-state memory technology, combine the speed of main memory with the persistence (non-volatility) of disk. Customers include global telecom operators and financial service providers.

Visit Solid Data on the web at <http://www.soliddata.com>

Corporate Headquarters:

3542 Bassett Street
Santa Clara, CA 95054
USA
Tel: +1.800.287.0373 ° +1.408.845.5700
Fax: +1.408.727.5496



The Solid Data logo is a registered trademark in the United States and Japan. TimesTen and the TimesTen icon are registered trademarks of TimesTen. All other trademarks or registered trademarks are the property of their respective owners.