Best Practices Guide

Sybase: Maximizing Performance through Solid State File-Caching

May 2000

Solid Data Systems, Inc. ◆ 2945 Oakmead Village Court ◆ Santa Clara, CA 95051 USA ◆ Tel +1 408.845.5700 ◆ Fax +1 408.727.5496 ◆ www.soliddata.com

Sybase:
Maximizing Performance through Solid State File-Caching

Summary

When a Sybase application server is I/O-bound on a transaction-intensive workload, the database administrator or system administrator should consider moving the most I/O-intensive "hot files" from hard disk drive (HDD) storage to solid-state file cache. This hardware architecture change can multiply server performance, throughput and scalability by a factor of 2x, 4x, 8x or more.

Applications: Highly transaction oriented Sybase databases where performance is inadequate despite extensive tuning efforts. Examples include billing, on-line trading, decision support systems (DSS), batch jobs that exceed available window, etc.

Target Environments: Billing applications take days to complete.
System response after a data entry are reaching 8 seconds or longer.
Queries are getting progressively longer as more table entries are added.
Batch jobs are taking 8 hours or longer.

Hot Files: Transaction logs, tempdb, highly accessed tables & indexes.

Performance Gains: Typical gains are 200% to 400% or more, with very little time investment.

Some real-world examples are:
– Billing application taken from 5 days to 2 days
– System response time after data entry from 15 seconds to 3 seconds
– Batch job taken from 8 hours to 2 hours
– Batch job taken from 72 hours to 8 hours

Business Results: Customers can experience reduced capital expenditures when compared to purchasing a commensurate number of servers to equal the performance gains of solid-state file-caching. Additional savings result by re-deploying highly skilled DBA resources to new development rather than tuning efforts on existing applications.

In many cases, customers have been able to generate more revenue when the database transaction is a billable event (on-line trading). Other times, customers have been able to add new clients because their systems can now handle the load. And still others have found that customer service ratings have improved by having faster response times during data entry (banking, customer support, CRM).

Problem &Current Practice

There are numerous database development tools available today that assist the DBA in database design. ER diagrams aid in defining relationships of objects in tables. Application tools help generate various GUIs and the associated views. Dynamic query optimizers assist in reducing unnecessary table scans. And with various levels of normalization, storage efficiencies can be achieved – at a trade off. While DBA can make generalizations that a particular schema will be "faster" or "slower," concrete database performance predictions remain elusive.

Real-World performance is difficult to predict accurately in the development stages for many reasons. Under what conditions, what server platform, which storage? How many simultaneous users? What queries will be running? Will it be "fast enough?" and by whose definition of "fast enough?" Development timelines are tight and don't allow for stress testing with extremely high transaction rates.

So in reality, DBA do the best possible under the given circumstances. And then the application goes "live." When the performance isn't "fast enough" by some arbitrary measure, the DBA then begins "tuning" the database. This can be a very time consuming effort but is likely necessary due to new understandings the DBA will gain from seeing how the application is being used in the real world.

However, once all the tuning has been done and performance still doesn't meet the "fast enough" mark, alternative measures must be investigated. These measures can be as extensive as adding new servers or as simple as moving hot files and tables to a faster storage device. The fastest storage device available is a solid-state file cache.

Identifying the Hot Files in Sybase Applications
In highly transaction-oriented Sybase applications, the following data types are typically the best ones to consider moving from hard disk drive (HDD) storage to solid-state file cache:

1. *transaction logs*
2. *tempdb*
3. *highly accessed tables/index structures*

A *transaction log* or *syslog* is a good candidate for solid-state file cache technology as it is always written to, during modification procedures, and is used to "roll back" transactions in the event of a system crash. The *transaction logs* can be written to a separate device within Sybase and can be easily moved to file cache.

Tempdb is another good candidate for solid-state file-caching, particularly when the application locks certain system tables, or where there is contention for them. Most Sybase applications use *tempdb* as the temporary working space when creating new tables, running queries, etc., so it is typically very I/0-intensive. For more on this topic see your Sybase SQL Server Performance and Tuning Guide or look online at: http://sybooks.sybase.com/onlinebooks/group-as/asg1200e/aseperf/@Generic__BookTextView/59331;cs=default;ts=default;pt=59331?DwebQuery=%22solid+state%22#X.

Highly accessed tables or index structures can also be good candidates for file-caching. Selection of tables and indexes for placement on solid-state file cache will depend on the way these data segments are accessed in a specific application. A Sybase white paper helpful in identifying hot tables can be found at http://www.sybase.com/products/adaptiveserver/migration/config.html. It discusses using the *dbcc memusage* command to display I/O statistics on a per object basis.

Raw Partitions vs. UNIX File Systems
For high performance applications, administrators typically configure Sybase applications with the data segments on raw disk partitions. In contrast to a raw partitions, a typical UNIX file system uses a read-ahead buffer cache -- which can actually slow down the I/O performance if the database has its own buffer cache. This is a particularly important point when the hot files are placed on solid-state file cache. Because a solid-state file cache can retrieve a random data block with no mechanical delays, access to the data is virtually instantaneous. Therefore a read-ahead cache offers no benefit; in fact, read-ahead caching can reduce the apparent performance of the solid-state file cache, by forcing the system to wait for movement of data that the application does not need and has not requested.

There is also a resiliency benefit for using raw partitions with Sybase devices. A normal UNIX file system uses a buffer cache for disk I/O. Since the Sybase database does a write-ahead buffering scheme, it assumes that the write has been committed when there is a possibility that data is still in the buffer waiting to be written to disk. In this scenario, if there were a system/disk failure before the buffer cache is written to disk, there would be no way to recover or rollback the data.

By using raw partitions, the server can then manage the disk I/O without any buffering from a file system so that if there were any system/disk failures, the system would know what part of the transaction completed and could recover or rollback the data.

Solid State File-Caching
Solid-state file-caching systems have no moving parts, so they experience no mechanical delays when accessing data. They can support random I/O rates measured in the thousands per second. This compares to rotating disk products than can support I/O rates on the order of one hundred per second. File caching delivers an order of magnitude advantage at small block sizes. For more information on solid-state storage and file-caching see the Solid Data white paper titled "Solid State File-Caching for Performance and Scalability," at http://www.soliddata.com/whitepapers/file_caching.html.

Business Results
A number of customers have been able to increase overall system performance by 200%, 400% 800% or more by placing the most I/O-intensive data segments on a Solid Data solid-state file cache. This enables them to scale a growing application workload on their existing servers, or to meet their business growth objectives with less costly investments in new sever hardware, data center floor space and skilled IT staff.