Scalable Caching in a Java Enterprise Environment with WebSphere eXtreme Scale

September 2009

WebSphere software

John Stecher,

Adam Stolle,

Robert Wisniewski

Executive Summary

The paradox of completing transactions faster with bigger and bigger data sets has been solved. IBM® WebSphere® eXtreme Scale V7 and IBM System x® hardware shattered the performance barrier that limited application scalability by introducing a linearly scalable high performance in-line cache to the database system. With this breakthrough new technology from IBM, developers can rapidly build a seamless, flexible in-memory data grid that scales out as their application scales unlocking them from being limited by the performance of the database.

This article reports a benchmark where a synchronous in-line eXtreme Scale database cache improved client performance by 365% over direct database access. This had a peak throughput of over 2.46 million transactions per minute. Furthermore, CPU consumption on the database system was reduced by 75%, enabling other critical applications to use the newly freed database capacity to improve their performance as well.

This new benchmark (called Fantasy Sports XS) simulates a world-wide, sports website where users create and manage a fantasy sports team and update personal preferences and options. An application of this scale and size presents significant challenge for administrators and developers because there is a need to scale in response to client demand increases. Utilizing WebSphere eXtreme Scale in these cases eliminates the concern of scaling the underlying database infrastructure as the eXtreme Scale grid scales in unison with the application independently of the database.

Three unique benchmark configurations were used to firmly prove out these conclusions. First, a traditional OLTP topology is utilized where the application uses JPA persistence APIs to directly access a traditional database. Secondly, an eXtreme Scale grid is introduced, with the application accessing the grid directly, and the grid using the same JPA persistence APIs to keep the data store updated in a synchronous manner. Finally, the advanced eXtreme Scale asynchronous write-behind technology is used to illustrate the benefits afforded by this type of optimization.

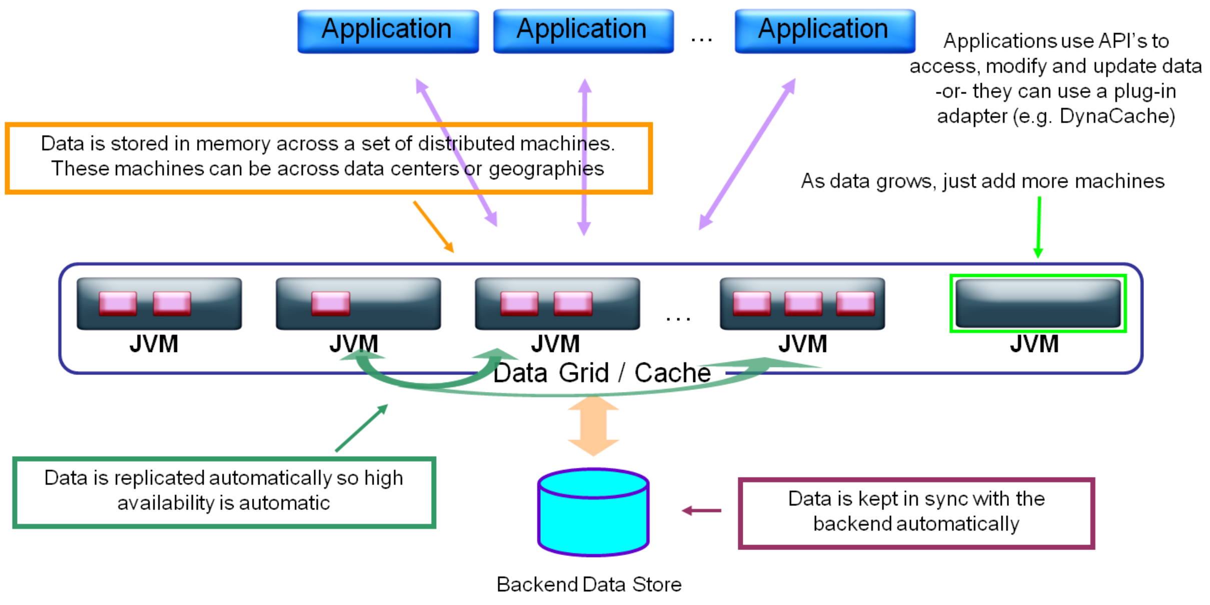
In-Line Data Caching with WebSphere eXtreme Scale

IBM WebSphere eXtreme Scale is a product which solves many design and implementation patterns in the enterprise environment. eXtreme Scale acts as an elastic data grid capable of creating a single virtualized memory space for a collection of member Java processes so that data can be easily, quickly, and transparently accessed regardless of location. To ensure that the data is consistent and will not be corrupted, the data is isolated and interacted with in a transactional matter just like a database. To provide fault tolerance and self healing characteristics the data can also be replicated in a variety of ways, which can also speed up access. All of this is accomplished with little to no administration or management of the grid itself. For example, an eXtreme Scale grid can dynamically expand or contract to utilize additional java processes provided.

An eXtreme Scale grid can be very effectively utilized as a data cache for a database or other data sources, which are generally slower to respond due to the need to access data on a hard disk. In addition, these data sources are generally more complicated and expensive to scale beyond a single instance and so have a limit on the total throughput that can be achieved. An eXtreme Scale grid has no such limitation and when properly used can scale with a linear response time without any reasonable upper bound. Applications have predictable scaling characteristics at a predictable cost as the need for resources grows. These optimizations for the data read and write operations significantly increase performance, sometimes by orders of magnitude. In addition, the overall load on the data source itself is drastically reduced along with response times.

A worldwide Fantasy Sports XS application is used as a new benchmark to demonstrate how eXtreme Scale can reduce database hardware needs while speeding up overall data access. This benchmark simulates the user management aspects of a worldwide fantasy sports website. The user session management is the key aspect here since game engines would handle the visualization aspects. Common user session operations such as retrieving and changing the core user's data, team members and athletes of interest form the workload.

The benchmark uses eXtreme Scale as an 'in-line cache' which is a common usage pattern. In this case the application retrieves and writes all data from the data source directly to and from the eXtreme Scale grid. The grid is configured with a 'Loader' which implements basic primitive operations to the data source such as create, update, delete, and get. The grid uses this implementation to keep the data source in sync with the information being requested from and written to the grid. By default all updates to the grid are written synchronously to the backend data source as part of the transaction so the cache is never out of sync with the database (a 'write-through' configuration). Optionally, the grid can be configured to buffer changes to the database for some period of time before asynchronously grouping all of the changes into a batch transaction and sending them in one large transaction. Generally this large transaction is more efficient, and additionally if a given object is updated multiple times in the asynchronous window, only one update with the latest version of the object is eventually written to the database. This avoidance of updates is called conflation. The buffer of updates is itself replicated within the grid and is also fault tolerant like the data itself, so even in the case of a container failure the database will continue to be in a coherent state. This asynchronous update configuration is called a 'write-behind' cache. The write-behind optimization provides the best performance results since the slower write operations are taken out of the client's critical path for the transaction.


Application
Application
…
Application
Applications use API's to access, modify and update data -or- they can use a plug-in adapter (e.g. DynaCache)
Data is stored in memory across a set of distributed machines. These machines can be across data centers or geographies
As data grows, just add more machines
JVM
JVM
JVM
…
JVM
JVM
Data Grid / Cache
Data is replicated automatically so high availability is automatic
Data is kept in sync with the backend automatically
Backend Data Store

Fantasy Sports XS

In this benchmark the user session information is stored in memory as a Java Object. This session can either be persisted through relational mapping via the JPA APIs or simply stored into the eXtreme Scale grid as a POJO using the eXtreme Scale ObjectMap APIs. (More information on persistence options can be found on the WebSphere eXtreme Scale Wiki.)

The Fantasy Sports XS application can be configured in multiple modes with its baseline case to use the traditional JPA data access model and its optional cases to be configured using the grid's APIs. When the application is configured to use the grid instead of the database via JPA, an eXtreme Scale Loader is provided to the grid deployment. This loader also uses JPA to retrieve and write data in the database, keeping it updated and retrieving any information the grid may not have currently loaded.

Benchmark Runtime Configuration Scenarios

The scenarios which have been tested in this paper represent common deployment patterns for applications of this type both with and without a data caching layer. They are outlined as follows:

1. **Traditional Direct Data Access Deployment**

The application simply uses JPA APIs to retrieve, manipulate and write session information. The mapping between the user session POJO and the relational data tables as well as associated SQL queries is accomplished using the JPA framework.

2. **Synchronous In-Line Data Cache (Write Through)**

   The application interacts only directly with the eXtreme Scale data grid via the ObjectMap APIs which is stored as a serialized Java object. When data is requested that is not currently in the grid, or when data is created or written, an eXtreme Scale Loader utilizing the JPA framework is used to then interact with the database. Any updates to the grid are written to the database synchronously within the active transaction. The update is only committed to the grid if a successful update to the data source is achieved.

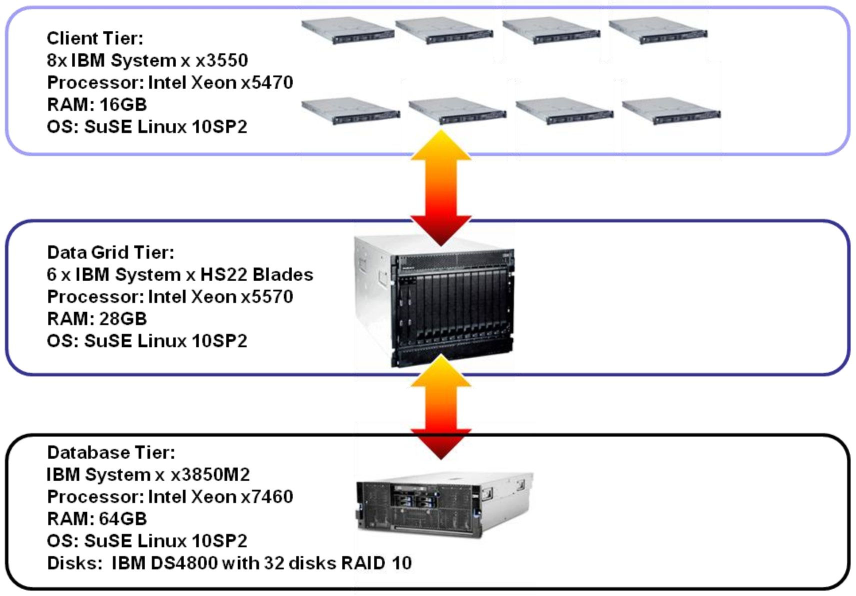
3. **Asynchronous In-Line Data Cache (Write Behind)**

   The application code is identical to the synchronous scenario and interacts once again only with the eXtreme Scale data grid. Any updates or new data is written only to the grid within the transaction itself, and the need for an update to the database is buffered in an update queue. This is accomplished with a simple grid configuration change. Updates are buffered for a maximum of two minutes before being written to the data.

Basic System and Software Configuration Parameters

Hardware Topology

State of the art hardware representative of what would be found in most high end high volume websites was used for this benchmark. We selected IBM's System x line of products because of its performance and reliability at all layers of a modern IT infrastructure.



The benchmark client hardware consisted of eight IBM System x x3550 machines each with two Intel x5470 3.33Ghz processors and 16GB of RAM running SuSE Linux 10SP2.

The hardware topology for benchmark testing where WebSphere eXtreme Scale was used consisted of six IBM System x HS22 blade servers featuring two Intel x5570 2.93Ghz processors and 28GB RAM also running SuSE Linux 10SP2.

The database tier consisted of a single IBM System x x3850M2 with six Intel x7460 processors and 64GB of RAM running SuSE Linux 10SP2. Backing the database system was an IBM DS4800 disk array with 32 disks combined into a single RAID-10 storage system for optimal IO performance.

Application Client Configuration

The application client side of the Fantasy Sports XS benchmark can be thought of as the code that typically resides inside of a servlet displaying a rich user interface or inside of a standalone JVM for simplified testing. The term client for the purposes of this benchmark is not the end user that would be browsing the website from their web enabled phone or laptop; it is instead the application code that is attached to the database or WebSphere eXtreme Scale grid.

For this benchmark case we wanted to ensure that the client was not a performance bottleneck, hence we took the approach of using the stand alone JVM client coupled with IBM System x s x3550 systems sporting 3.33Ghz Intel Xeon x5470 processors. Each client JVM ran eight threads executing the client code to correspond with the eight hardware threads in the system thus minimizing contention on shared data structures in the client code and focusing the benchmark's performance on the backend systems.

The application client tier was sized with an appropriate number of machines such that there were no network bottlenecks or CPU bottlenecks during testing. In a real world deployment the client code would be designed in the same manner for optimal performance and availability.

WebSphere eXtreme Scale Configuration

The middle tier of the configuration is the tier containing the WebSphere eXtreme Scale grid. This tier given its use case had to use hardware with a significant amount of physical memory capacity as well as network bandwidth to the client and database tiers. We selected IBM System x HS22 blades with Intel x5570 2.93Ghz processors to take advantage of their memory density as well as performance.

On each HS22 blade we ran sixteen WebSphere eXtreme Scale instances each instance having a 1GB JVM heap. This gave us the ability to fit the whole 2 million user database inside of the grid as would be expected for optimal performance in a production environment. As the database grew in users given WebSphere eXtreme Scales linear scalability more blades could be added to the cluster to maintain response times.

Database Configuration

The database tier for this benchmark uses IBM's DB2 9.7 as the backing persistent datastore for Fantasy Sports XS. It resides on an IBM System x x3850M2 with Intel x7460 processors running at 2.66Ghz with 64GB of RAM. Backing the physical server infrastructure is a set of IBM DS4800 disk arrays complete with 32 disks configured for RAID-10.

The database for the Fantasy Sports XS application consists of 2 million unique users each with 1-4 Fantasy Sports Teams which each in turn have 1-10 members. This database and user load is consistent with many of the leading fantasy sports sites in production today around the globe.

The Results

Baseline Scenario

The baseline scenario for this paper is the common case where the Fantasy Sports XS application is developed using the JPA programming model (OpenJPA in our scenario) to access the backend database system. As client load is ramped up against the Fantasy Sports XS website the database machine becomes the limiting factor for performance.   In this case the database dictates how many users can be supported as the application tier can be scaled out horizontally nearly infinitely given its stateless design, but its dependency on a single database system limits the scalability of the overall infrastructure.

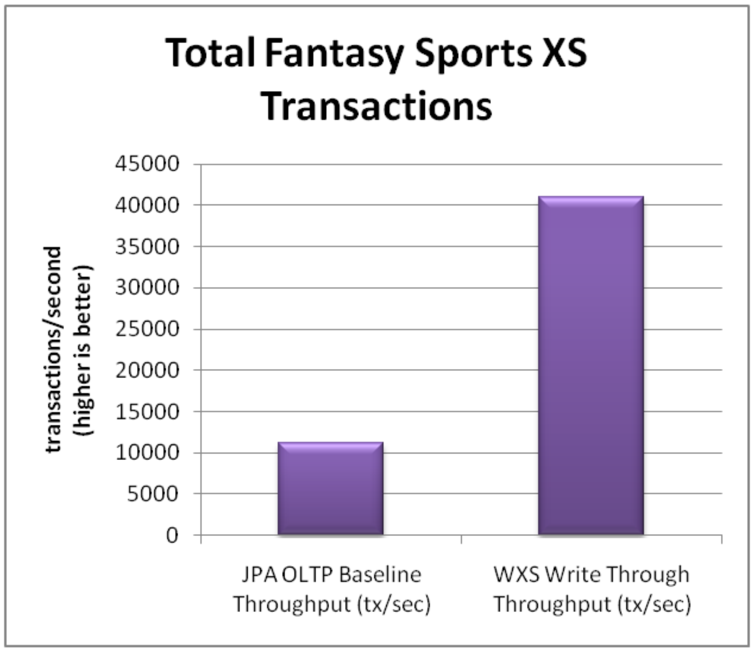
During our testing the JPA based client application running against a highly optimized DB2 9.7 database system produced very respectable throughput results with 8971.4 reads/sec (56ms response time), 2134.6 updates/sec (79ms response time), 66.5 deletes/sec (77ms response time) and 47.2 inserts/sec (95ms response time) which results in nearly 11209.7 transaction/sec.

The application server tier in this case was running at nearly 50% average CPU utilization with the database tier running at 90% average CPU utilization. Adding more client load to the application resulted in exponentially increasing response times thus the above metrics were taken as the best possible performance the topology could yield for the JPA programming model case.

Synchronous Data Cache (Write Through)

Since our database system was the limiting factor for increasing capacity of the application and we cannot grow that tier anymore hardware wise to support additional users, we will offload the database by storing data into WebSphere eXtreme Scale using it as an inline cache between the application tier and the database tier. This scenario illustrates a significant improvement of overall throughput by over 365% from 11209.7 transaction/sec to 41019.7 transaction/sec. In addition the load on the database is reduced by nearly 75%. The response times for read operations went from 56ms to 10ms, a 560% improvement. Update operations improved from 79ms to 57ms, showing an improvement of 139%.

The significance of these massive performance improvements in this scenario has massive real world implications to the simulated business. By being able to offload nearly 75% of the database load (effectively dropping database CPU consumption from 90% to 23% the database system now has CPU cycles available to support the deployment of new applications into the business IT infrastructure. At the same time the performance improvements of the Fantasy Sports XS application being 365% faster mean that the applications user load can now effectively grow 3.65X without impact on the backend database system above and beyond the tested 23% CPU consumption this scenario showed.



Where does this gain come from? Most of the benefit is the offloading of a significant amount of the read operations from the database and into the eXtreme Scale grid. The same update, insert and delete operations are still being handled by the database and thus data integrity is maintained on the backend system but because of the reduction in read traffic the database is able to much more effectively process those specific transactions improving performance.

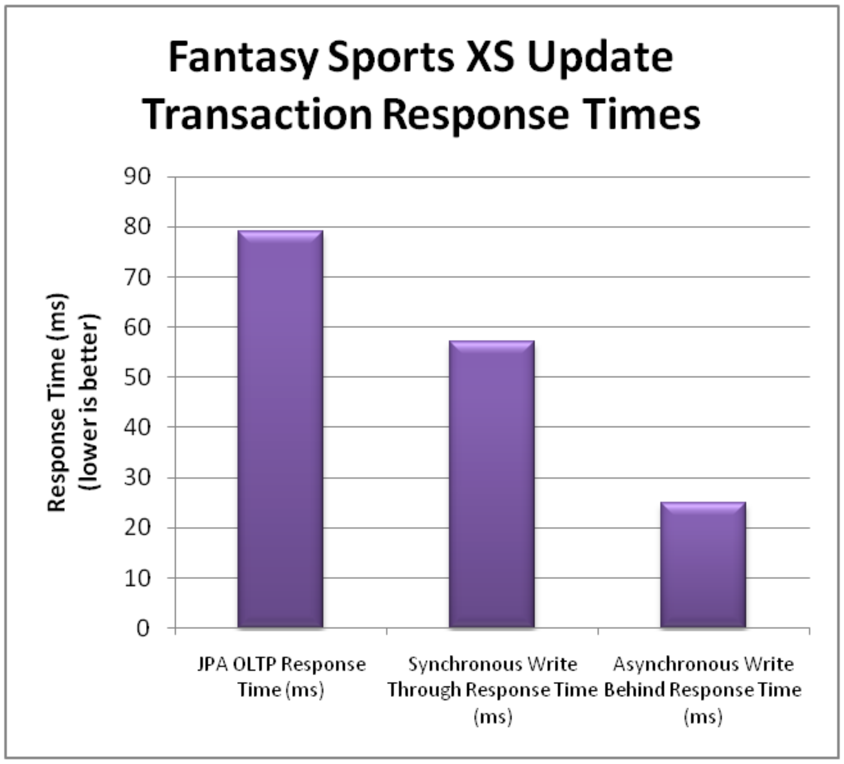
The below table summarizes the actual individual transaction results

Scenario	Baseline Throughput (tx/sec)	Baseline Response Time (ms)	Write Through Throughput (tx/sec)	Write Through Response Time (ms)	Percent Improvement with WXS
Read	8971.4	56	32843.7	10	366%
Update	2124.6	79	7765.6	57	366%
Insert	47.2	77	199.7	47	423%
Delete	66.5	95	210.7	55	315%

Asynchronous Data Cache (Write Behind)

This scenario further improves overall end client response time over the synchronous data cache deployment by 79% or more for data manipulation scenarios (update/insert/delete) while holding constant the performance of the read scenarios. The load on the database is reduced by a nearly 80% over the traditional JPA application deployment, and 50% reduction when compared to the write through cache. The update operation response time dropped significantly from 79ms to 25ms over the traditional deployment.

The further significant performance improvements in this case are due to the removal of the database update from the critical path of those transactions with the update occurring only in the WebSphere eXtreme Scale data grid. Database utilization dropping even further when compared to the write-through deployment is due to the conflation of multiple updates of a single object into a single object within the buffering window as well as the greater efficiency of large batched transactions over individual write operations for each data object.



In the case of Fantasy Sports XS, where the application can tolerate an interval of time without updates to data elements being pushed out to the actual database, significant gains in response time performance can be realized by taking advantage of the asynchronous write-behind feature.

Conclusion

In this paper we presented benchmark testing showing that a synchronous in-line WebSphere eXtreme Scale database cache improved the Fantasy Sports XS scenario performance by 365% over a standard JPA based implementation. This allowed the existing infrastructure with the simple addition of the WebSphere eXtreme Scale layer to grow from supporting an already impressive 672,500 transactions per minute up to nearly 2.45 million transactions per minute. When you couple this impressive throughput performance improvement with the fact that the load on the database system was reduced by nearly 75% allowing for other applications to utilize the freed up cycles on the database you can easily see how injecting WebSphere eXtreme Scale into your IT infrastructure can reduce your TCO significantly and enable your existing backend systems to support significantly more applications.

Moving beyond using WebSphere eXtreme Scale as a pure inline cache with synchronous write-through behavior to an asynchronous write-behind cache with a 2-minute asynchronous window performance was further improved above the synchronous case with response times dropping another 1.79X giving the end user of the Fantasy Sports XS application cutting edge performance no matter the load applied to it while allowing them the opportunity to tune the environment to meet their applications data consistency needs.

Overall WebSphere eXtreme Scale represents a significant leap forward in technology allowing administrators and developers to expand their existing infrastructure significantly while not having to increase the size of their database systems. In fact most will see reductions in database CPU and IO consumption allowing for significantly expanded IT operations.