An Oracle White Paper
April 2010

Deploying Web 2.0 Applications on Oracle® Servers and the OpenSolaris Operating System

ORACLE

Introduction 1
Typical Web 2.0 Architecture 2
Solution Overview 3
Solution Configuration 4
  Hardware Configuration 4
  Software Configuration 6
  Olio Configuration 6
Configuration and Tuning 7
  Apache HTTP Server 8
  PHP 8
  Memcached 9
  Database Server 9
Test Results 10
  Throughput and Response Time 10
  System Performance 10
  Web Tier 10
  Object Store 11
  Database Servers 12
Scaling a Web 2.0 Deployment 12
Best Practices for Deployment 13
Conclusion 14
About the Authors 14
References 15

Introduction

The rise of the social networking phenomenon has spurred demand for a highly scalable and flexible solution for hosting applications. Many larger sites are growing at 100% a year, and smaller sites are expanding at an even more rapid pace, doubling every few months. Such rapid growth in user population requires a highly flexible and scalable infrastructure. Today, many sites are using open source solutions and platforms as a way to ensure applications can scale as needed.

Many developers and deployers use extremely simple static file retrieval tests to evaluate Web servers, often leading to erroneous conclusions. Understanding the challenges faced by Web 2.0 customers, Oracle created Olio. Developed by engineers with years of experience and understanding related to the complexities associated with creating multitier workloads, Olio provides a realistic workload that can help test the performance and scalability of Web 2.0 technologies. This technical white paper describes a reasonably sized, scalable Web 2.0 deployment. Olio helped validate and tune the solution, resulting in several best practices for deployment.

Typical Web 2.0 Architecture

The widespread use of open source technologies in Web 2.0 deployments is resulting in the standardization of a horizontally-scaled architecture (Figure 1). Typical Web 2.0 and enterprise 2.0 sites support clients that make requests over the internet. These requests are first processed by a set of load balancers or reverse proxy servers that handle static requests and forward dynamic requests to back-end Web and application servers. The Web tier accesses structured data from back-end database servers and unstructured data from object stores as needed. The Web application also accesses external resources—Web services that are external to the site—such as a geocoder.

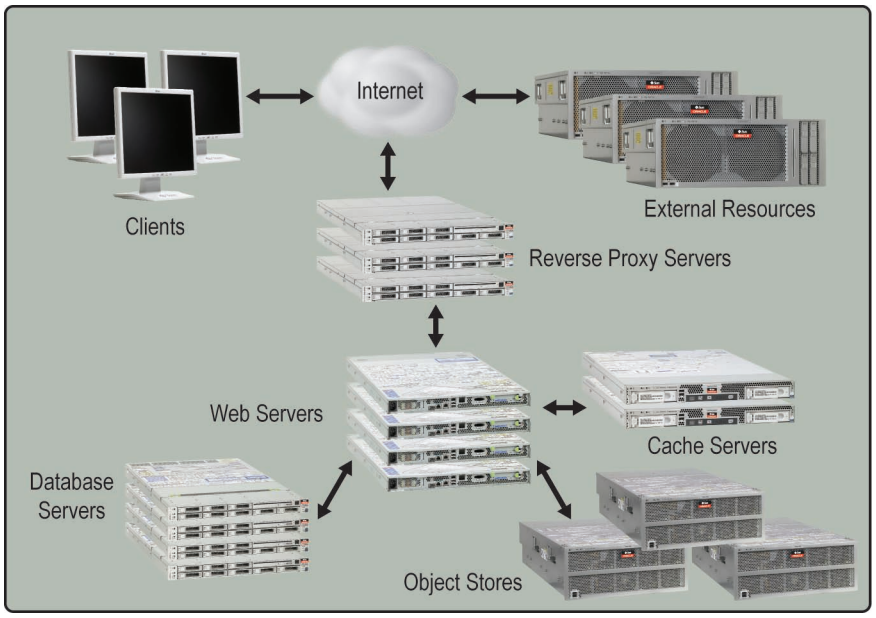

Figure 1. Solution overview

A caching tier is deployed to off-load the database. The Web application first looks for data in the cache servers. If the data is not found, or the cache timeout has expired, data is fetched from the back-end database. The data is obtained, manipulated if necessary, and saved in the cache. Finally, the Web page is generated and returned to the client browser. As a result, the caching tier also helps to reduce the load on the Web application tier by caching page fragments and modified data sets, eliminating the need for the application to recompute these for every request. Similarly, load balancers are used in the front end to multiplex requests from many users onto a smaller pool of connections to the Web tier.

Solution Overview

This technical white paper uses the Olio application to showcase a deployment. Olio is a Web 2.0 toolkit for evaluating the suitability, functionality, and performance of Web technologies[1]. The application provides an events site similar to popular social networking sites on the internet, featuring photos, calendars, and shared comment feeds. It includes common Web 2.0 features, such as Asynchronous JavaScript and XML (AJAX), tagging, tag clouds, comments and ratings, friends, feeds, and more. In addition to the structured data in a relational database, Olio uses a large unstructured dataset with items such as photos and documents, as well as a caching tier that is typical of Web applications. The Olio toolkit also includes the tools needed to drive load against the application in order to measure performance.

An open source Apache software incubator project, Olio provides three implementations of the same application using three different technology stacks: PHP, Rails, and the Java™ Platform, Enterprise Edition (Java EE platform). The deployment described in this article uses the PHP implementation of Olio deployed on commonly used open source technologies that are popular with Web 2.0 sites. These technologies include

- **Apache HTTP Server.** The Apache HTTP Server is a widely used, open source Web server that powers many small to medium sized Web sites. The Apache HTTP Server typically is deployed in a multiprocess mode (MPM-prefork) with the `mod_php` software module for PHP-based applications.
- **PHP.** PHP is a scripting language used for Web applications. An open source version of the PHP runtime is available and used with the Apache HTTP Server as part of the popular Apache HTTP Server, MySQL database, and PHP (AMP) stack.
- **APC.** APC is a widely used, open source implementation of a caching and optimizing accelerator for PHP that can help improve PHP performance.
- **Memcached.** Memcached is an open source distributed caching system that is used to reduce the load on back-end databases, and is the de-facto standard for Web applications.
- **MySQL database.** MySQL database server is a popular open source relational database management system that is developed, distributed, and supported by Oracle. The MySQL software delivers a fast, multithreaded, multiuser, and robust SQL database server that can be used to deliver e-commerce, online transaction processing (OLTP), and multiterabyte data warehousing (DW) applications. A fully integrated, transaction-safe, ACID-compliant database with full commit, rollback, crash recovery, and row-level locking capabilities, MySQL database provides ease of use, scalability, and performance for Web 2.0 applications.

[1] See *http://incubator.apache.org/olio*.

Solution Configuration

Figure 2 depicts the configuration used to deploy the Olio application. It differs from the typical Web 2.0 deployment shown in Figure 1—it does not use load balancers or reverse proxy servers, and the caching tier is combined with the Web server tier.

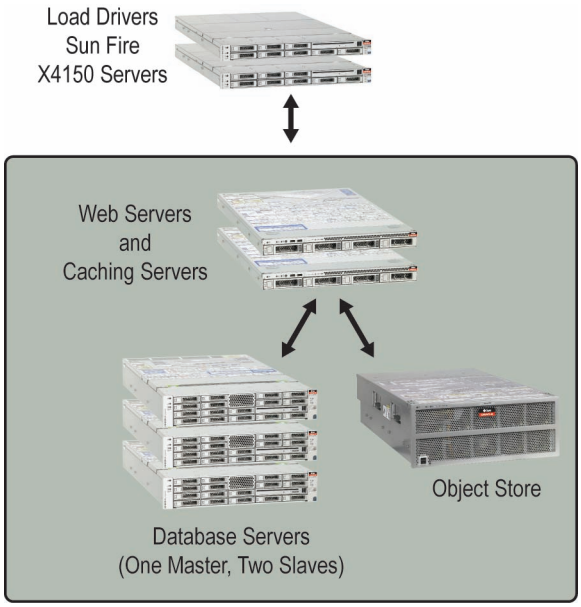

Figure 2. Solution configuration

- The deployment architecture uses two systems for the Web tier. Each driver system was configured to route requests to a separate 1 Gigabit Ethernet interface on each of the Web server systems to manually load balance the system.
- Since the Olio application contains very little static content, a reverse proxy server is not needed. However, the lack of a load balancing or proxy tier requires connections to be made to the Web tier for every active user. As a result, the system must be able to support a very large number of Apache software processes, each with their own connection to the back-end database. These factors influence the tuning required, such as configuring sufficient swap space.
- Memcached uses few CPU resources. Rather than implementing a separate caching tier, memcached is deployed on the systems running the Web servers. By using high-performance systems, the tiers can be combined on fewer systems to reduce hardware costs and simplify deployment.

Hardware Configuration

The testing deployment solution consisted of load drivers, Web servers, database servers, and an object store configured as follows.

Load Drivers

Load drivers generate the workload to drive the Web application, and sit outside the actual application infrastructure. Two Sun Fire X4150 servers from Oracle were used to run the load generators, each on a separate subnet. Each system incorporated one quad-core Intel® Xeon® 5440 processor and 32 GB of memory.

Web Servers and Caching Servers

The deployment configuration took advantage of the high performance of Oracle's Sun Fire X2270 servers and combined the Web server and caching tiers on the same system. Offering a compact 1U form factor, the powerful performance of Intel Xeon Processor 5500 Series with HyperThreading, optional Sun Flash Modules from Oracle, and the versatility to execute most any x86 or x64 application, the Sun Fire X2270 server is an excellent building block for Web 2.0 infrastructure environments.

The Sun Fire X2270 servers used for the testing effort included two Intel Xeon X5570 processors, two standard Gigabit Ethernet interfaces, and 48 GB of main memory. Two of these servers ran the Web application and memcached software. One network interface on each system was used to communicate with the driver, and the other network interface was used for sending all traffic to the database and object store.

Database Servers

The database was deployed using MySQL database replication on three Sun Fire X4170 servers from Oracle, configured as one master and two slaves. The Sun Fire X4170 server delivers density, high performance, and energy efficiency in a compact 1U form factor. Incorporating two Intel Xeon Processor 5500 Series processors, HyperThreading and QuickPath Interconnect technology, up to eight 2.5-inch disk drives, four Gigabit Ethernet ports, energy efficiency enhancements, and enterprise-class reliability, availability, and serviceability (RAS) features, these servers are well suited for database and Web 2.0 infrastructure deployments.

The Sun Fire X4170 servers used for the testing effort included two quad-core Intel Xeon E5540 processors running at 2.53 GHz and configured with 8 MB cache, 24 GB of memory, and eight 146 GB capacity SAS disk drives. One network interface was used for all traffic between the Web application and the database.

Object Store

The object store was deployed on a network-attached storage (NAS) appliance. Oracle's Sun Storage 7210 system provides 44 TB of capacity in an energy-efficient 4U package. Incorporating general-purpose x64-based servers, Oracle storage products, write-optimized solid state devices (SSDs), and built-in analytics, the Sun Storage 7210 system helps to simplify enterprise storage management.

The appliances used for the testing effort included two SSDs to buffer storage writes and dramatically improve performance. A single Gigabit Ethernet interface was used to route all traffic to and from the Web server systems.

Software Configuration

The systems in the deployment utilized the following software components.

OpenSolaris Operating System

Based on the Oracle Solaris operating system, the popular OpenSolaris operating system gives developers and enterprises access to the latest features before they are integrated into the Oracle Solaris 10 OS. By offering organizations advanced technology, access to the latest innovations from the OpenSolaris community, innovations in packaging, and the assurance of Oracle's proven, extensive testing processes and worldwide support, the OpenSolaris operating system makes it faster and easier to develop and deploy enterprise applications and services.

Every system in the testing configuration ran the OpenSolaris 2008.11 release, which is available with production support. No changes were made to any operating system tunables on any of the systems.

Sun Web Stack

Sun Web Stack is a complete LAMP or SAMP cross-platform portfolio of Web tier technologies. It allows developers and administrators to leverage the technologies they already know, including Apache HTTP Server, Tomcat, MySQL database, and PHP, and enables enterprises to standardize their existing LAMP or SAMP deployments and strategies. Several components from the integrated Sun Web Stack were utilized for the testing effort, including Apache HTTP Server 2.2.9, PHP 5.2.6, memcached 1.2.5, and MySQL database 5.0 client software. The database servers ran the 64-bit version of MySQL database 5.1 with patches.

Object Store

The unstructured object store was deployed on the Sun Storage 7210 system by using the integrated Oracle Solaris Zettabyte File System (ZFS) Hybrid Storage Pool—a mechanism that places Flash devices into a new storage hierarchy to assist disk drives by locating the ZFS Intent Log (ZIL) on write-optimized SSDs to help improve write performance. All configuration and monitoring is performed using a browser interface.

The object store was configured as a single pool using mirroring for redundancy. A file system was created on the pool and mounted on the */export/filestore* mount point.

Olio Configuration

The Olio application has built-in support for MySQL database replication. Users can specify the names of the master and slave hosts in the application's *etc/config.php* configuration file. Each session picks a slave to connect to, and uses that slave for the entire session. All read requests are sent to slaves, and all write requests (create, update, delete) are sent to the master. Read requests in the same transaction as a write request are executed on the master for transaction consistency.

Two instances of memcached were run on the two Web server systems. Olio automatically configures the array of memcached instances specified in the configuration file. It relies on the PHP Extension Community Library (PECL) memcached client to distribute cached objects between the instances.

The Web tier uses the Network File System (NFS) to mount the file system containing the object store. As a result, the filestore appears as a local file system to the Olio application.

Olio uses Faban to drive a workload against the application. The Faban harness is infrastructure for hosting workloads and automating execution, and was installed on all systems in the configuration. Designed to drive the application and to simulate real world interactions by a number of concurrent users, Faban gathers data from a test run and presents it to users in graphical form through a Web interface. It can manage the systems used in a workload test, including the system under test, load generation systems, and even database and cache servers commonly used in complex workload testing. Two Sun Fire X4150 servers were used to run the Faban driver during the testing effort. One of these systems also ran the Faban master. See *http://faban.sunsource.net/0.9/docs/guide/harness/overview.html* for a more detailed explanation of how Faban works.

The Olio workload tests seven operations, and has requirements for average and 90% response times for each operation. Each operation includes several HTTP requests that execute the logic needed to generate a Web page. On average, each operation involves six HTTP requests and 10 database queries. The driver controls the selection of an operation (based on a pre-configured mix), the login and logout of users, the selection of cycle times, and more. It also keeps track of metrics. At the end of a run, the driver prints out a summary and detailed report.

The PHP implementation of Olio version 0.1 was downloaded and installed from *http://incubator.apache.org/olio/downloads.html* following the setup instructions documented with the toolkit.

Workload Scaling

The Olio application is scaled using a single parameter: the number of active users that connect to the Web application and perform various operations repeatedly. This parameter is referred to as scale or users. The size of the database and object store are derived from this parameter. The database is sized to hold 100 times as many users as the number of active users, with the cardinality of all other tables set appropriately to maintain key ratios. The sizing is intended to reflect the usage pattern found in many Web 2.0 sites that boast millions of users but on average have a much smaller number of active users. The size of the object store is derived from the cardinality of the events and persons tables—each event has an image, thumbnail, and literature, while every person has an image and thumbnail.

The workload was scaled to 10,000 concurrent users for the testing effort. This created one million users in the database and over 400,000 files consuming 1 TB of space in the object store.

Configuration and Tuning

Several changes were made to the software stack for the tested deployment configuration. These changes were not intended to create a highly optimized environment or to maximize performance. Instead, the changes focused on configuring the systems in a reasonable manner with the minimal changes necessary to support the scale of users and operations executed during the testing effort. As shown in the results below, considerable headroom remains in the configuration to support additional loads or periodic spikes in usage.

Apache HTTP Server

The OpenSolaris operating system 2008.11 release includes version 2.2.9 of the Apache HTTP Server. By default, the Apache software runs in MPM pre-fork in 32-bit mode when using `mod_php` to access PHP files on the OpenSolaris operating system. The following parameters were changed in the Apache software *httpd.conf* configuration file that resides in the */etc/apache2/2.2* directory. These changes were made to support the large number of connections and requests tested.

```
<IfModule prefork.c>
ListenBacklog 16384
ServerLimit 8192
MaxClients 8192
MaxRequestsPerChild 0
StartServers 128
MinSpareServers 20
MaxSpareServers 128
</IfModule
```

PHP

PHP version 5.2.6 was used for the testing effort. The Olio application uses the following PHP extensions: `apc`, `curl`, `gd`, `memcache`, and `pdo_mysql`. These are included and enabled by default in the OpenSolaris operating system. The following changes were made to the *php.ini* configuration file located in */etc/php/5.2/php.ini* to support the application.

- `error_reporting`: The default value of `E_ALL` was changed to add `E_NOTICE` messages for debugging.
- `upload_tmp_dir`: The Olio application performs many file uploads which are copied by the Web application to the object store. Since the uploads are temporary, the */tmp* directory was chosen. The */tmp* directory resides on swap in the OpenSolaris operating system, and therefore uses main memory.
- `session.save_path`: PHP saves user session data in session files, one file per session. If all files are saved in the same directory, then the file system lock that protects that directory from concurrent updates can become a bottleneck. Instead, the `session.save_path` parameter distributes the session files across 32 subdirectories, each protected by their own lock.

```
error_reporting = E_ALL & ~E_NOTICE
register_long_arrays = On
upload_tmp_dir = /tmp
session.save_path = "1;/tmp/http_sessions"
```

Memcached

Memcached version 1.2.5 was run in 32-bit mode. No special tuning was used.

Database Server

The database servers ran MySQL database 5.1 with patches. The database was created on the master using the InnoDB storage engine on an Oracle Solaris ZFS files system. Three internal disks were used to create a zpool, with the database and logs residing on the same zpool. Default options for the OpenSolaris operating system and the Oracle Solaris ZFS file system were used without modification.

After the database was created, it was copied to the slaves and replication was activated as described in the *http://dev.mysql.com/doc/refman/5.1/en/replication-howto.html* page. The following parameters were changed from their default values in the MySQL database *my.cnf* configuration file.

- `innodb_flush_log_at_trx_commit`: The `innodb_flush_log_at_trx_commit` parameter controls transaction commit rates. For full durability, the default value is set to one, which causes InnoDB to flush the log at every commit. Although this is typical behavior for online transaction processing (OLTP) applications, most Web 2.0 applications have more relaxed durability requirements and set this parameter to a value of two. This implies that the log buffer is written to the file at each commit, but the flush to disk operation is performed only once per second (rather than the default of once per transaction).
- `innodb_doublewrite`: The `innodb_doublewrite` variable is enabled by default, causing InnoDB to store all data twice—first to the doublewrite buffer, and then to the actual data files. Since the test database resides on the Oracle Solaris ZFS file system, which maintains its own consistency, the penalty of storing the data twice can be avoided.
- `join_buffer_size`: The `join_buffer_size` parameter sets the size of the buffer that is used for plain index scans, range index scans, and joins that do not use indexes and therefore perform full table scans. The Olio application includes several queries that perform joins. As a result, increasing this parameter helps to reduce latency.
- `table_open_cache`: The `table_open_cache` parameter sets the number of open tables in all threads. The default value of 64 needs to be increased to support the very large number of threads used in the testing effort.

```
innodb_flush_log_at_trx_commit=2
innodb_doublewrite=0
join_buffer_size=8M
table_open_cache=2048
```

Test Results

The following sections describe some of the metrics and statistics from the 10,000 active users (1,000,000 users in the database) Olio application workload test.

Throughput and Response Time

Results show that a throughput of 2008.573 operations/second was achieved for 10,000 concurrent users. Response time metrics are reported in the Faban summary report, and are listed in Table 1.

TABLE 1. RESPONSE TIME RESULTS

OPERATION	AVERAGE RESPONSE TIME (SEC)	MAXIMUM RESPONSE TIME (SEC)	90TH PERCENTILE RT	REQUIRED 90TH PERCENTILE RT	PASS OR FAIL
HomePage	0.074	6.827	0.200	1.000	Pass
Login	0.083	8.740	0.060	1.000	Pass
TagSearch	0.145	6.958	0.460	2.000	Pass
EventDetail	0.038	3.292	0.060	2.000	Pass
PersonDetail	0.240	9.991	0.520	2.000	Pass
AddPerson	0.666	8.863	1.440	2.000	Pass
AddEvent	0.681	7.204	1.540	4.000	Pass

System Performance

Overall system performance is gated by CPU and network performance in the Web tier. Considerable load exists on the object store as well—in terms of both network and disk I/O—since the application creates files as new users and events are added and retrieves stored objects for existing events and users. Since the application uses memcached to cache data, the number of requests sent to the database is greatly reduced. This is important as it allows the Web tier to scale many times over before the database becomes a bottleneck.

Web Tier

Table 2 lists key system statistics measured at the Web tier during the testing effort. Network writes from the Web server systems to the driver systems exceeded the capacity of one Gigabit Ethernet interface. As a result, two interfaces were used—one for each of the Web server and driver system combinations. The second interface on each Web server system was used for traffic to the object store and databases. These results show that the high performance of the server hardware rendered a single Gigabit Ethernet interface insufficient. Depending on the application, two or four 1 Gigabit Ethernet interfaces, or a 10 Gigabit Ethernet interface could be required.

TABLE 2. WEB TIER RESULTS

DESCRIPTION	RESULT
Web tier CPU utilization	58%
Driver to Web server request throughput	92.9 MB/sec
Driver to Web server response throughput	154 MB/sec
Web server to object store response throughput	53.3 MB/sec
Web server to object store request throughput	41.4 MB/sec

Object Store

The Sun Storage 7210 system provides a Web interface for accessing built-in Sun Storage 7000 Unified Storage Systems DTrace Analytics capabilities that provide dynamic instrumentation for real-time performance analysis and debugging. Statistics can be accessed for any duration of any day, saved to a worksheet, and exported to a spreadsheet for further analysis. Figures 3, 4, and 5 show the disk and CPU statistics for the 10,000 user run.

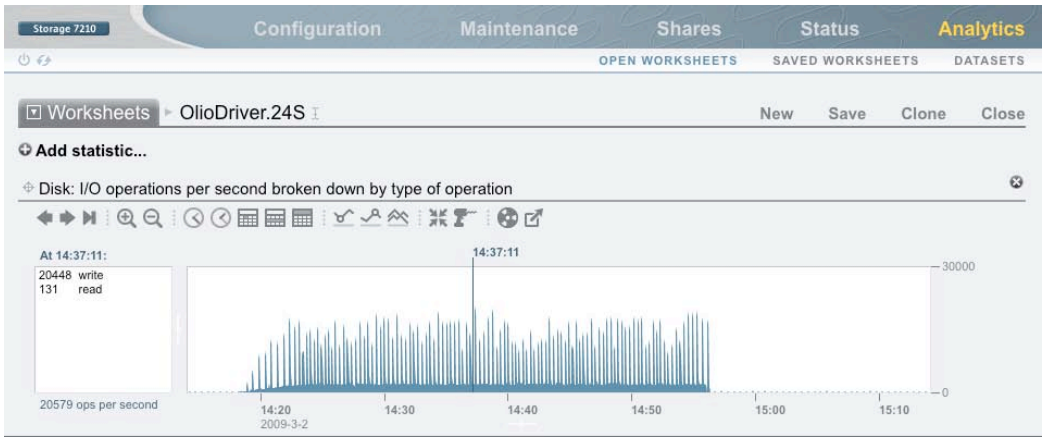

Figure 3. I/O operations per second statistics captured during the testing effort

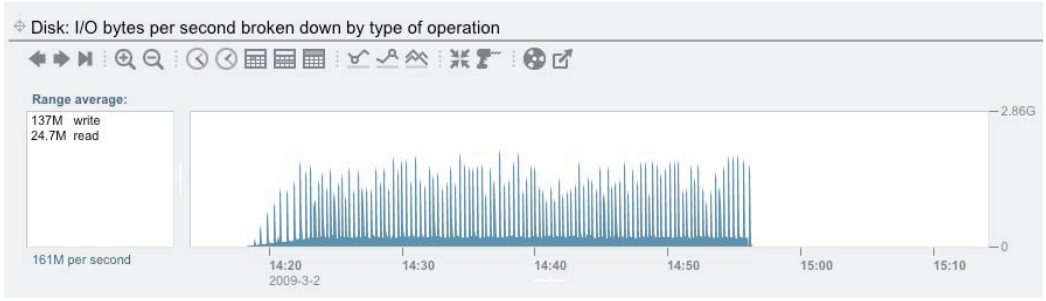

Figure 4. I/O bytes per second statistics captured during the testing effort

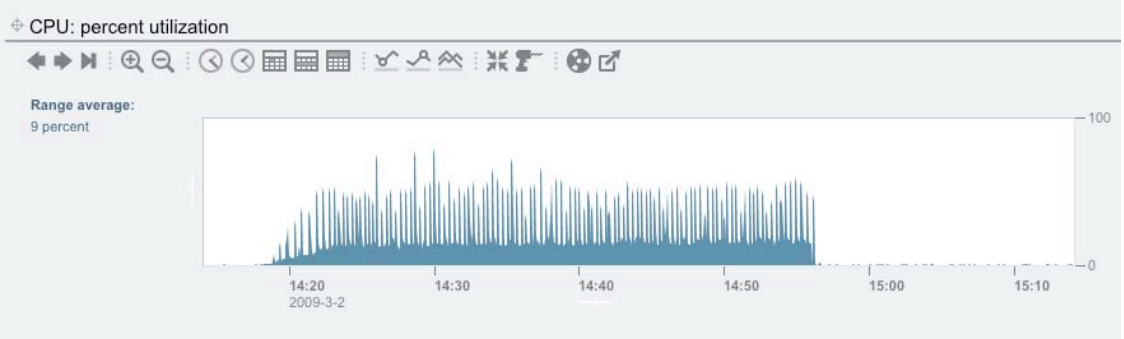

Figure 5. CPU utilization statistics captured during the testing effort

Database Servers

The database servers were lightly stressed in terms of CPU and disk utilization because of the use of the caching tier (memcached) as well as MySQL database replication that spread the load to three servers. Although a large database configuration was not required for the scale of testing, the effort aimed to showcase a solution that still works as the Web tier is scaled many times to add more users.

Database response time also is crucial for this workload, making it important to tune various parameters in the *my.cnf* configuration file so that the master database can handle 20,000 connections and each slave can handle 10,000 connections. A slow response from the database server(s) can cause the response time criteria for the operations to fail. The operations most affected are the database-intensive `AddEvent` and `AddPerson` operations.

Scaling a Web 2.0 Deployment

The solution showcased allows horizontal scaling of the Web application and memcached. Oracle's Sun Blade 6000 family provides a modular way to scale the infrastructure (Figure 6). With chassis designed for investment protection, organizations can cable once and change deployment options as required—mixing and matching server modules as desired. Industry-standard I/O provides flexibility and throughput for individual servers. In addition, transparent networking and management means that Oracle's Sun Blade 6000 Chassis fit into existing network and management infrastructure. Even with this architecture, the single object store and database are likely to become a bottleneck at some point.

Database traffic can be minimized by using aggressive caching techniques in the memcached tier. The object store can be expanded by using the clustering capabilities and expanded storage of the Sun Storage 7410 system. When these options are exhausted, further scaling requires sharding the database and application to split it into multiple horizontal and/or vertically partitioned databases. In addition, unstructured data can be partitioned into multiple file systems distributed across multiple Sun Storage 7000 Unified Storage Systems.

Figure 6. Sun Blade 6000 product family

Best Practices for Deployment

The exercise of deploying, scaling, and performance tuning the Olio application revealed several best practices for deployment, including:

- **Use /tmp for HTTP file uploads.** The default temporary directory in the OpenSolaris operating system is the */var/tmp* directory, which typically resides on the root file system. Besides causing problems such as running out of disk space, this can be a significant performance issue when many files are being uploaded. By using */tmp*, which is located in swap (main memory) by default, both of these issues can be solved.
- **Configure sufficient main memory and swap.** Sun Fire X2270 servers include very fast CPUs and are capable of handling many users. However, sufficient memory must be provided. During testing, the systems were configured with 48 GB of memory, helping to support 5,000 users as well as provide sufficient resources for uploaded files in the */tmp* directory and memcached needs. It is important to ensure that sufficient swap space is configured. For example, prefork MPM shares memory between processes, with only a few pages replicated due to copy-on-write functions. However, the operating system must reserve extra swap space so that the total amount of virtual memory can exceed the working set of physical memory.
- **Use multiple directories in /tmp for saving HTTP sessions.** Session files are stored in the */var/php/5.2/sessions* directory in the OpenSolaris operating system by default. While sessions can survive failures and reboots, performance bottlenecks can result. During the testing effort, sessions were saved to the */tmp* directory, eliminating I/O. This strategy worked well for up to 4,000 users. For larger numbers of users, the session files were distributed into multiple directories using */tmp/http_sessions* directories. Be aware that using the world-readable */tmp* directory could pose a security problem. See *http://us.php.net/manual/en/session.security.php* for more information.

- **Optimize the memcached tier.** With Oracle's new, powerful Sun Fire X2270, X4170, X4270, and X4275 servers, consider combining the memcached tier with the Web tier. The lightweight memcached server consumes few CPU resources. By configuring sufficient memory in the Web tier for both the Web application and memcached, it is possible to simplify deployment and gain some performance.
- **Use SSDs where performance matters.** The Sun Storage 7210 system was configured with two write-optimized SSDs that stored the ZFS Intent Log. Performance was improved, as the Olio workload creates many new files that generate 20,000 writes/sec (peak) to the storage subsystem. Similarly, SSDs could help speed MySQL database response times. For write-intensive database workloads, placing the InnoDB log on SSDs has the potential to improve performance dramatically. The Sun Flash Analyzer tool can help identify whether SSDs can help improve performance for an application.

  The same is true for the Apache HTTP Server access log. By using larger, more powerful multicore systems, a single system can support thousands or tens of thousands of requests per second, causing the access log to become a choke point. Writes to the access log can take several milliseconds. For critical applications, consider using SSDs.

Conclusion

This technical white paper describes a solution for a Web 2.0 application deployment on Oracle servers and the OpenSolaris operating system. It shows how to scale a medium sized Web 2.0 site with a million registered users with open source technologies, and includes detailed configuration and tuning information for the deployment, along with tips for obtaining the best performance from such a configuration. The solution uses Olio—a Web 2.0 toolkit that Oracle has invested in—to help customers analyze and scale deployments. Olio allows users to performance test new technologies in a stable environment without having to devise expensive, cumbersome tests on production applications.

By combining Oracle's Sun Storage 7210 systems running the OpenSolaris operating system and the integrated Sun Web Stack, the solution is effective for Web 2.0 applications. Oracle's Sun Fire X2270 and X4170 servers running the OpenSolaris operating system have sufficient capacity to combine the Web and caching tiers and handle a significant workload—10,000 concurrent users running a PHP application using two pre-configured Gigabit Ethernet network interfaces.

About the Authors

Shanti Subramanyam is a Senior Staff Engineer at Oracle. She is currently leading a performance project on Web 2.0 applications using open source technologies on Oracle systems. She is a committer on the Apache incubator project Olio, and has been instrumental in the development of several industry-standard and proprietary benchmarks for database and Java technologies.

Richard Smith is a Staff Engineer at Oracle, where he divides his time across a range of projects, including the performance of MySQL database, interpretive languages, and distributed file systems. He

has a particular interest in large-scale scientific parallel computing, and everything that is involved in making things go fast.

Paul van den Bogaard is a Staff Engineer at Oracle. He is a performance expert on Oracle databases, and currently focuses on open source databases like the MySQL database.

Adam Zhang is a Software Engineer at Oracle. His responsibilities include helping ISV partners and open source communities adopt emerging Oracle technologies. In this capacity, Adam works on issues related to system architecture, sizing, performance tuning, and benchmarking.

References

For more information, see the references listed in Table 3.

TABLE 3. REFERENCES FOR MORE INFORMATION

WEB SITES	
OpenSolaris Operating System	http://www.opensolaris.org
Sun Blade Servers	http://www.oracle.com/us/products/servers-storage/servers/blades/index.htm
Sun Fire X2270 Server	http://www.oracle.com/us/products/servers-storage/servers/x86/030679.htm
Sun Fire X4270 Server	http://www.oracle.com/us/products/servers-storage/servers/x86/030682.htm
Sun Servers	http://www.oracle.com/us/products/servers-storage/servers/blades
Sun Open Storage	http://www.oracle.com/us/products/servers-storage/storage/open-storage/index.html
Sun Flash Storage	sun.com/storage/flash/resources.jsp
Sun Web Stack Resources	http://wikis.sun.com/display/BigAdmin/LAMP-SAMP
Sun's Web 2.0 Scalable Performance Toolkit	http://wikis.sun.com/display/BluePrints/Web+2.0+Scalable+Performance+Toolkit
Faban Project	http://faban.sunsource.net
OpenSolaris Project: Web Stack	http://www.opensolaris.org/os/project/webstack
Olio	http://incubator.apache.org/olio
WHITE PAPERS	
MySQL Guide for Sun Storage 7000 Unified Storage System	sun.com/blueprints
Configuring Sun Storage 7000 Systems for Oracle Databases	sun.com/blueprints

Oracle is committed to developing practices and products that help protect the environment

Deploying Web 2.0 Applications on Oracle Servers and the OpenSolaris Operating System
April 2010
Authors: Shanti Subramanyam, Richard Smith, Paul van den Bogaard, Adam Zhang

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

SOFTWARE. HARDWARE. COMPLETE.