

An Oracle White Paper
March 2011

Oracle Exalogic Elastic Cloud: Software Overview



Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Introduction

The Oracle Exalogic Elastic Cloud is an engineered system comprised of both hardware and software components. Software, such as the operating system or embedded firmware, is very tightly integrated with the physical devices in each Oracle Exalogic Elastic Cloud system. The Oracle Exalogic Elastic Cloud Software, by contrast, is a set of enhancements made to the core Oracle Fusion Middleware products. These software enhancements greatly increase the performance of Fusion Middleware based applications when deployed on Exalogic Elastic Cloud hardware.

The Oracle Exalogic Elastic Cloud is designed for maximum performance and reliability and most applications will experience a significant performance improvement even when only a subset of the optimizations that exist at different layers in the system, are leveraged. The primary sources of application performance benefits can be attributed to the fact that all Exalogic hardware systems:

- Incorporate the most powerful processors available
- Include very large amounts of very low latency memory
- Are built on an extremely high bandwidth and low-latency InfiniBand I/O fabric that connects all major system components
- Feature high-throughput, low latency shared storage directly attached to the I/O fabric
- Include tuned and optimized Oracle Solaris and Oracle Linux
- Support the Exalogic Elastic Cloud Software, which features enhanced Multi-core processing, InfiniBand networking and Oracle Database integration

As a result, the Exalogic system will improve performance of applications which:

- Are highly distributed, such as a modern SOA-based application or applications that make extensive use of enterprise messaging
- Use in-memory session state replication
- Handle large volumes of HTTP requests, such as e-commerce and social media applications
- Have large memory requirements or are highly multi-threaded
- Execute large volumes of transactional interaction with Oracle 11g Database and/or Real Application Cluster instances

Ultimate Enterprise Application Consolidation Platform

Although virtually all applications will demonstrate a significant improvement, only applications which utilize Oracle's latest Fusion Middleware products and the Exalogic Elastic Cloud Software can achieve the maximum possible performance gains an Exalogic system. The Exalogic Elastic Cloud Software (figure 1), includes enhancements made to Oracle WebLogic Server, the Oracle JRockit and Oracle HotSpot Java Virtual Machines (JVM) and the Oracle Coherence in-memory data grid. These software enhancements and new features are included in the standard installer binaries for the latest 11gR1 versions of Fusion Middleware, and are intended for use exclusively within an Exalogic system deployment. The Exalogic Elastic Cloud Software optimizations leverage the specific versions of Oracle Linux and Oracle Solaris that are incorporated with the Exalogic Elastic Cloud hardware.

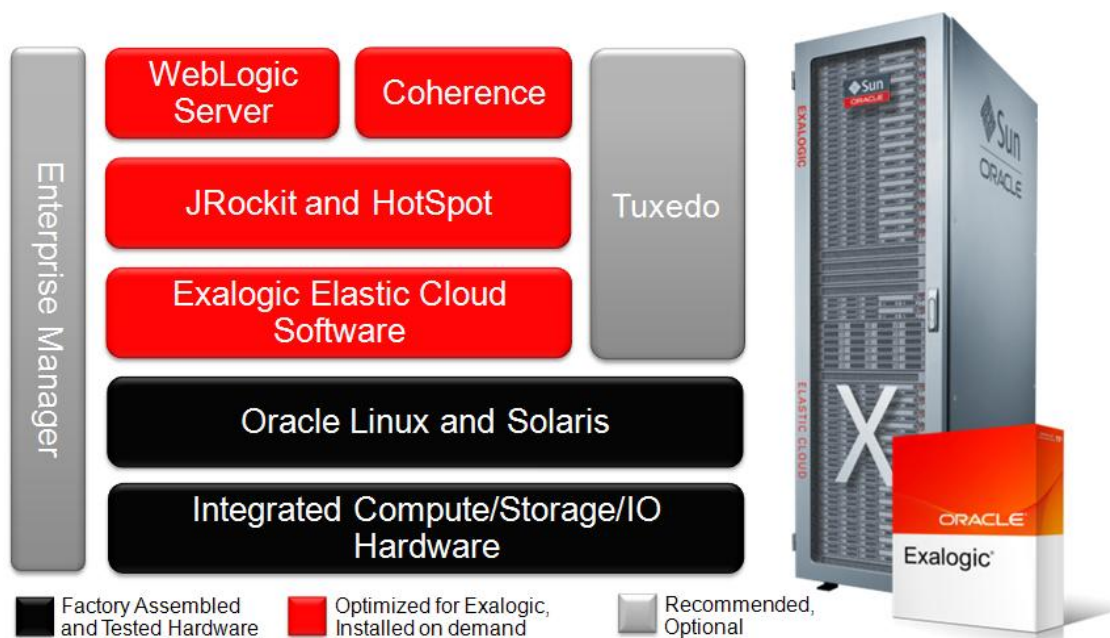


Figure (1) Oracle Exalogic Elastic Cloud Software

The key benefits provided by the Exalogic Elastic Cloud Software enhancements can be grouped into four primary categories:

1. Increased WebLogic Server scalability, throughput and responsiveness

Improvements to WebLogic Server's networking, request handling and thread management mechanisms enable it to scale better on the multi-core compute nodes that are connected to the fast InfiniBand fabric that ties all the Exalogic components together. WebLogic Server, with assistance from the underlying JVM and operating system, is able handle more of its request

processing work in parallel, with less thread locking, and is able to reduce the latency of the network communication that occurs between different WebLogic instances running on different compute nodes. When generating responses, WebLogic Server works in conjunction with the JVM to reduce the amount of costly data copying that usually occurs between system layers, reducing the burden on system memory. The net effect is each WebLogic server can handle more client requests while at the same time reducing the time taken to respond to each individual request.

2. Superior WebLogic Server session replication performance

Many Web applications keep track of the current state of an end user's interaction using a 'session object' managed by the application server (a.k.a. an HTTP Session). To avoid losing a user's session data in the event of a server failure, WebLogic Server can automatically replicate this session data from the 'primary' server to a 'secondary' server in the cluster each time session data changes, leading to increased high availability. However, this session replication process does come with an inevitable performance cost, especially if a large amount of user data is being held in each session. For Exalogic, WebLogic Server's replication mechanism is improved to utilize very high I/O bandwidth (InfiniBand 40Gb/s), available for inter-process communication between servers. WebLogic Server replicates more of the session data in parallel, over the network to the secondary server, using concurrent connections, (figure 2).

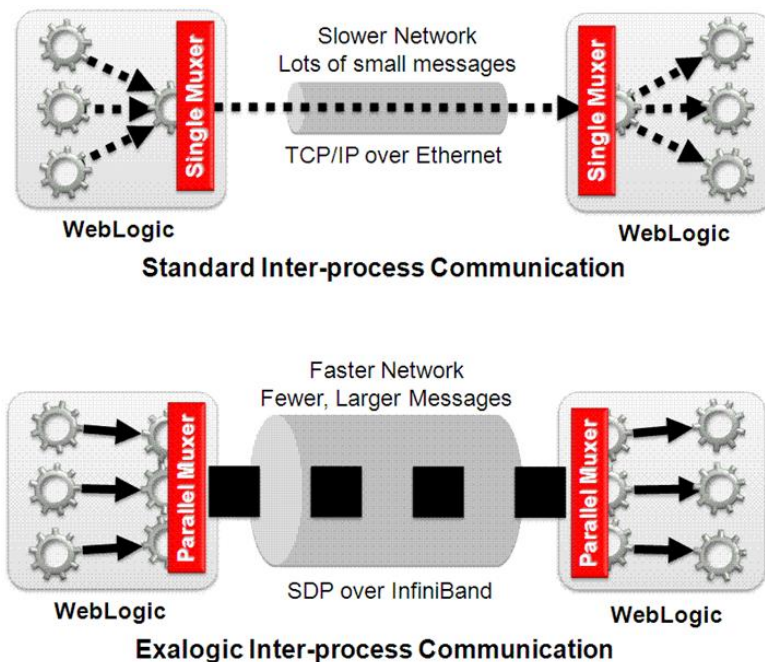


Figure (2) WebLogic Inter-Process Communication

WebLogic Server also avoids a lot of the unnecessary session processing work that usually takes place when the secondary server receives a copy of the session data. With the help of the underlying JVM, WebLogic Server uses InfiniBand's fast 'native' networking protocol, called Sockets Direct Protocol (SDP). The use of SDP enables session payloads to be sent over the network with lower latency. The net effect is, for stateful web applications requiring high availability, end-user requests are processed far more quickly.

3. Tighter Oracle RAC integration for faster and more reliable database interaction

A new technology component has been added to WebLogic Server on Exalogic called 'Active GridLink for RAC (Real Application Clusters)'. Active GridLink for RAC provides optimized WebLogic-to-Oracle RAC database connectivity by interfacing WebLogic Server directly with the Oracle RAC protocols. This new technology supersedes the existing WebLogic Server capability commonly referred to as 'Multi-Data-Sources'. Active GridLink for RAC provides more intelligent load-balancing across RAC nodes based on the current workload of each RAC node, faster connection failover if a RAC node fails and more transparent RAC node location management. Active GridLink is also able to handle global transactions more optimally, where multiple individual database operations have been encapsulated into a single atomic transaction, by the enterprise Java application. The net effect is, for enterprise Java applications involving intensive database work, applications achieve a higher level of availability with better throughput with reduced and more consistent response times. In addition to the performance benefits, Active GridLink for RAC (figure 3), also simplifies the configuration of Data Sources in WebLogic Server, reducing the number of configuration artifacts required for Oracle RAC connectivity and mostly eliminating RAC Service configuration changes from having a corresponding change in WebLogic Server.

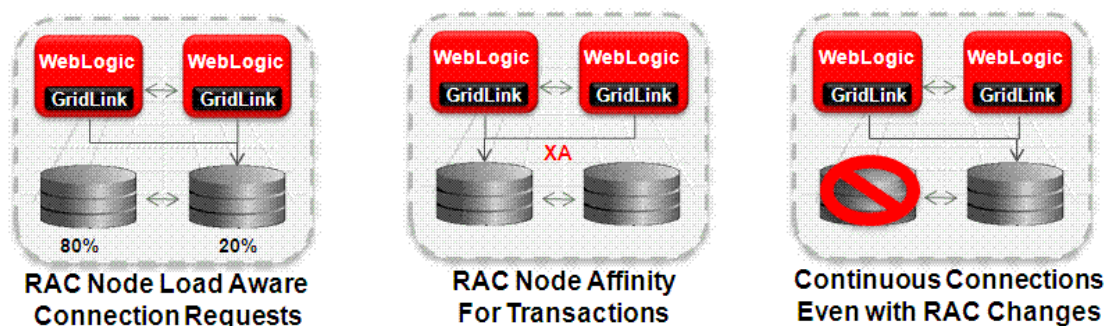


Figure (3) Active GridLink for RAC

4. Reduced Exalogic to Exadata response times

In situations where an Exalogic system is directly connected to an Exadata system, using InfiniBand, the Active GridLink for RAC mechanism provides an additional feature enabling WebLogic Server to leverage InfiniBand's fast “native” SDP networking protocol, to interact with the Oracle RAC database on Exadata. This results in lower latency for request-response times for calls between WebLogic Server and the database. The performance gain is most significant when large results sets are returned from the database. The net effect is, for an enterprise Java application that interacts with Exadata, the application is able to respond to client requests more quickly.

Exalogic Elastic Cloud Software: Technical Detail

A technical description of the product optimizations, under-pinning the four key benefits, follows.

Increased WebLogic Server scalability, throughput and responsiveness

WebLogic Server

- Use of a new Java NIO based “parallel muxer” (figure 4) for more efficient use of threads and greater throughput. This new muxer uses an improved non-blocking mechanism for request processing and uses multiple Java NIO channel selectors to maintain 3 different lists of open sockets, resulting in less lock contention. With this in place, fewer threads are now able to process larger number of requests, thus reducing the amount of costly context switching that inevitably occurs when there are many competing threads.

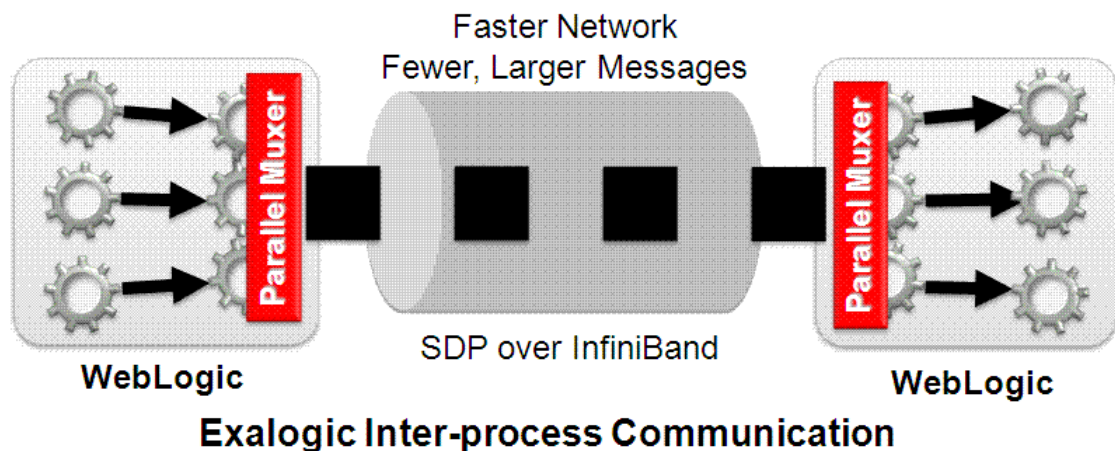


Figure (4) Parallel Muxer for WebLogic Inter-Process Communication

- An optimized work scheduler is employed, providing improvements to the Increment Advisor used to manage the size of WebLogic Server's Self-Tuning Thread Pool, (figure 5). The Increment Advisor makes more significant adjustments, if necessary, every several seconds, by incrementing the number of pooled threads by the number of hardware threads on the host machine. Previously, an increment of 1 was always used. For Exalogic, this increment becomes 24 (12 cores with hyper-threading). This enables the server to reach its steady state more quickly, where there is an optimum balance between the number of pooled threads for peak-load request processing and the number of cores available on the host machine.

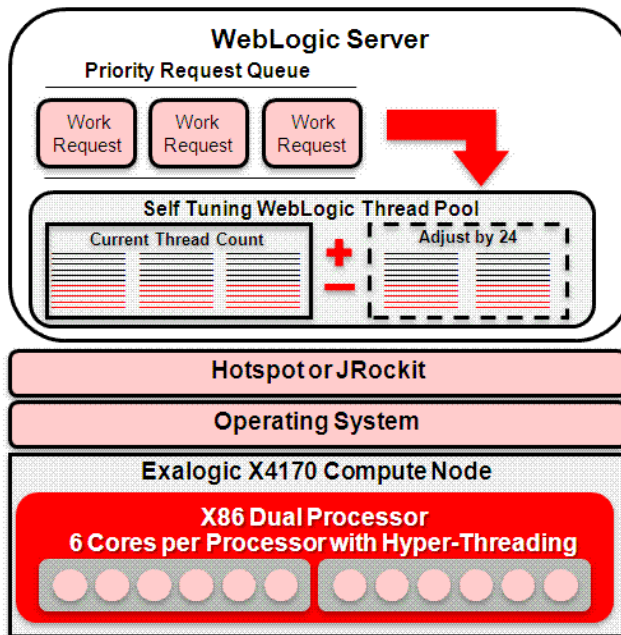


Figure (5) Self Tuning WebLogic Thread Pool

- Modifications to reduce the number of data buffer copies (figure 6), that occur have been incorporated. WebLogic Server has changed to use byte buffers to collect data responses. These buffers are shared between the WebLogic's sub-system layers, in contrast to the old behavior where copies of data arrays were made and then passed between the sub-system layers. The layers include the WebLogic JSP compiler, WebLogic Servlet Container and WebLogic core system. The byte buffers are heap based (i.e. non-direct Java NIO ByteBuffers) and are arranged as 4k chunks, scattered in the JVM heap memory. As a result of the enhancements, there is a significant reduction in the amount of objects created for request/response processing. This reduced heap usage also means that less frequent expensive garbage collections occur.

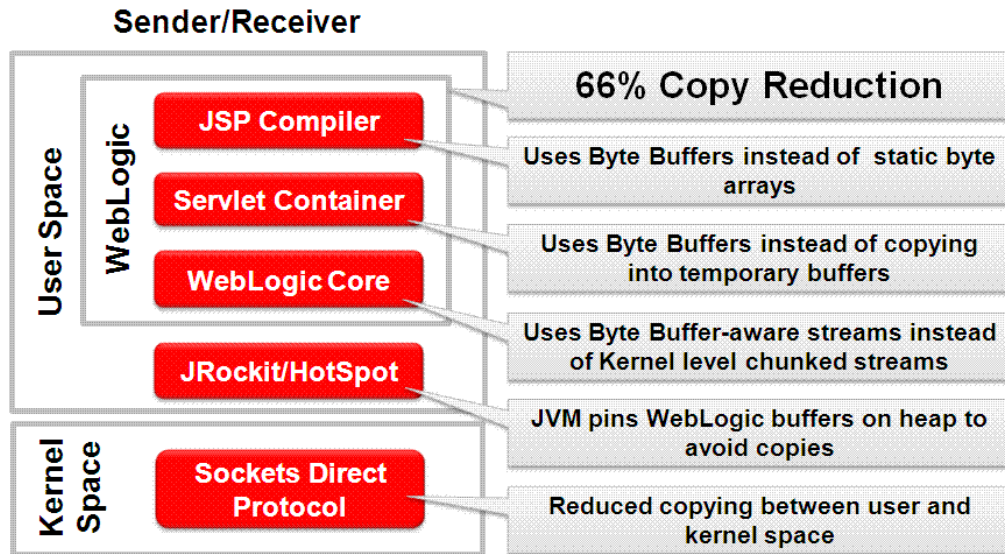


Figure (6) Reduced Data Buffer Copies

Java Virtual Machine

- Use of the Vektored I/O pattern (a.k.a Scatter/Gather I/O) between WebLogic and the JVM (figure 7), reduces the overhead incurred when sending data over the network. When the JVM receives the Java NIO based network API call, from WebLogic Server (see above), it performs a 'gather' operation on the scattered set of byte buffer chunks and copies them into a single heap-based, non-direct, byte buffer. The JVM is then able to make a single and more efficient network call to the InfiniBand user-space libraries, provided by the operating system, to send the data over the network.

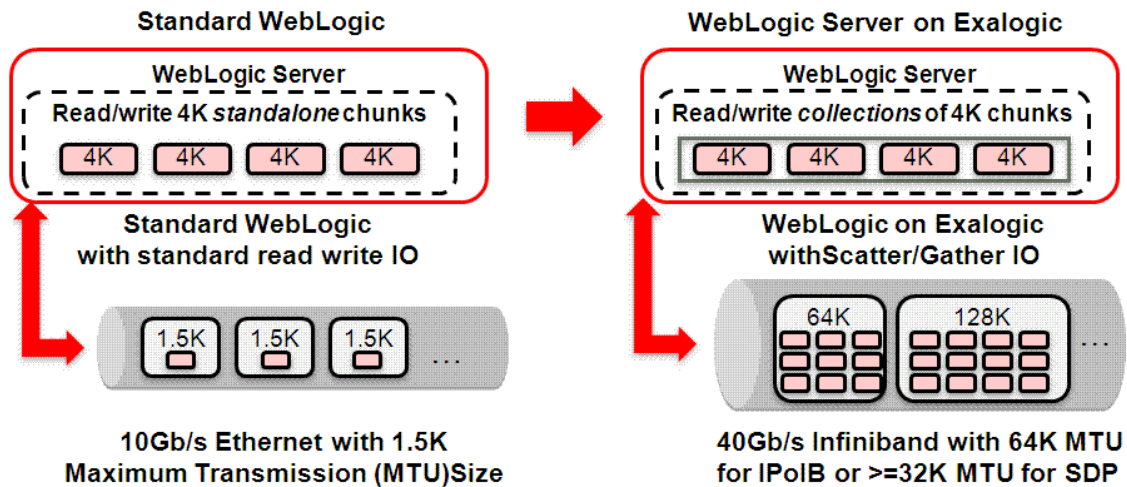


Figure (7) WebLogic on Exalogic with Scatter/Gather IO

- The JVM pins the memory address position of the new single byte buffer, in place on the heap, for the lifetime of the network I/O call. This avoids the need for an extra copy of the data to a safe native memory buffer. Normally, making a native copy is necessary because a JVM cannot guarantee that the memory address of an object in the heap will remain the same over time. Without this enhancement, JVM procedures, such as garbage collection, will frequently result in the object being moved to a different memory address in the heap (e.g. due to the promotion of an object from the nursery to the old space, or due to defragmentation).

Note: With the combined copy reduction optimizations in WebLogic Server and the JVM, at least one copy operation of data still occurs.

Coherence

Coherence has had a number of changes made to its networking behavior to better run on Exalogic. For example, it now chooses the host's network device that has the highest MTU specified, to use for communication with other cluster members. Coherence does not support the use of its TCMP protocol over SDP rather than IP. Coherence is usually network I/O bound and already demonstrates far superior performance on Exalogic when compared to commodity hardware. This is mainly due to the presence of the InfiniBand based network rather than Ethernet.

Oracle Linux and Oracle Solaris

- The default IP over InfiniBand (IPoIB) network settings for the host compute nodes are tuned to have a Maximum Transmission Unit (MTU) set to a higher value than is used by traditional Ethernet networking. These are set to 64k on Exalogic. Given that a single homogeneous IPoIB network is present inside Exalogic, it is possible to assume a much higher known ceiling value for network packet sizes. This enables more efficient transmission of fewer, larger packets, between WebLogic Server instances. The

optimization is especially useful when large data-sets need to be passed between servers. Often, in heterogeneous network environments, the MTU ends up being reduced to a lowest common denominator value, such as 1.5k with Ethernet, in an attempt to avoid fragmentation of network packets.

Superior WebLogic Server session replication performance

WebLogic Server

- Use multiple replication channels for synchronous in-memory session replication between servers in a WebLogic cluster. Multiple socket connections exist between each pair of WebLogic instances that are acting as the primary and replica memory stores for a set of HTTP sessions. Rather than there being a single pair of 'RJVMs', between two communicating servers, multiple RJVMs are utilized with Exalogic. This helps reduce the socket handling lock contention that usually occurs in the sending and receiving servers, when there is just a single server socket present. As a result, much higher parallelism is achieved when replicating many simultaneous sessions. This greatly increases throughput and ultimately reduces the latency of the user request-response times landing on the primary server. This optimization does not apply to all types of T3 based traffic; it is only used for the replication channels.
- Deferred de-serialization of session data on the replica server until required (a.k.a. Lazy De-serialization). In the vast majority of cases, replicated session data will not actually be needed on a replica server, and will be over-written by subsequent updates. In non-Exalogic systems, WebLogic Server always de-serializes the session data into a set of Java objects in the replica server's heap, even though it is unlikely to be used. With this optimization, the time-consuming de-serialization process is avoided and the burden on the replica server's heap, CPU and garbage collection process is reduced. If failover does occur, the replica's Servlet container can just de-serialize the latest session data on demand.
- Use of the InfiniBand 'native' network protocol, Sockets Direct Protocol (SDP), for replicating all session data between two servers is incorporated. This avoids the use of the operating system's TCP/IP stack and the added latency that it would otherwise incur. Using SDP rather than TCP/IP over InfiniBand yields better response times especially in cases where the data payload is large, as is often the case for HTTP Sessions. This optimization does not apply to all types of T3 based traffic; it is only used for the replication channel. SDP characteristics also enable the adoption of 1-way RMI calls for session replication, which avoid the overhead of a synchronous 2-way RMI call.

Java Virtual Machine

- The JVM provides an implementation of the InfiniBand "native" network protocol, Sockets Direct Protocol (SDP), and exposes a Java API for this protocol, for use by other Oracle technologies running on the JVM (specifically WebLogic and JDBC clients). An operating system's TCP/IP network software stack includes some capabilities that InfiniBand already caters for at a lower level (e.g. message loss avoidance). As a result, the latency added by such redundant TCP/IP features, can be avoided. SDP is a highly efficient stream sockets based wire protocol used for more efficiently controlling InfiniBand's Remote Direct Memory Access (RDMA). With RDMA, one compute node writes, over-the-wire, to the memory of another target compute node, with only minor intervention required by either

compute node's processors. By using SDP, lower latency and higher throughput is achieved for session replication and the compute node's CPUs have more free cycles to perform other work.

Oracle Linux and Oracle Solaris

- Where SDP is used by WebLogic Server and the JVM, larger Message Transmission Units (MTU) are adopted (e.g. 128k in many cases) to enable more efficient transmission of fewer, larger packets, between the WebLogic servers on different compute nodes in the same InfiniBand fabric.

Tighter Oracle RAC integration for faster and more reliable database interaction

WebLogic Server

- Provides the new “Active GridLink for Oracle RAC” data-source capability for connecting WebLogic to an Oracle RAC cluster. This is an Exalogic-only feature of WebLogic and is intended to be used in preference to the existing “Multi-Data-Sources” feature of WebLogic. With Active GridLink, WebLogic subscribes to the database's Fast Application Notification (FAN) events using Oracle Notification Services (ONS). This allows the WebLogic connection pool to make more informed runtime load-balancing decisions based on the current workload of each RAC node. By sending database calls to the node with lowest current workload, the average latency of user request/response times is reduced. Additionally, Active GridLink uses Fast Connection Failover (FCF), which also uses the notification events, to enable rapid database failure detection for greater application resilience. This feature also eases the system management overhead by allowing WebLogic to automatically reconfigure its connection pool, in the event of planned additions or removals of RAC nodes. For multiple database interactions in the same global transaction, Active GridLink uses XA Transaction Affinity to pin all related database calls to the same RAC node. This avoids the overhead that occurs when a global transaction is spread across multiple RAC nodes. The outcome is improved database processing efficiency, whilst still allowing for transactions to be recovered on a different RAC node, if the original RAC node happens to fail. If the target Oracle RAC database is version 11gR2 or greater, Active GridLink for RAC allows a Single Client Access Name (SCAN) to be used. With SCAN, a simple JDBC URL can be specified for the WebLogic connection pool and, as a result, it is not necessary to change the URL, each time a RAC node is added to or removed from the database cluster.

Reduced Exalogic to Exadata response times

WebLogic Server

- Active GridLink for RAC (see above) also includes an option to communicate with Oracle RAC nodes using SDP over InfiniBand (SDPoIB) rather than IP over InfiniBand (IPoIB). This optimization is only achievable when Exalogic is linked to Oracle RAC running on Exadata and doesn't make sense for 'standalone' Oracle RAC on an Ethernet based

network. This enhancement avoids the use of the operating system's TCP/IP stack and the added latency that it would otherwise incur. When the option is enabled, WebLogic configures the Oracle Thin JDBC Driver with a URL and specific JDBC property to indicate to the JDBC Driver that SDP should be used for all JDBC communication with the database. Using SDP rather than TCP/IP over InfiniBand yields better response times especially in cases where the data payload is large, as is often the case for large JDBC result sets.

Thin JDBC Driver

- The Oracle Thin JDBC Driver has been enhanced for Exalogic to optionally allow the use of SDP over InfiniBand (SDPoIB), for JDBC interactions between Java applications (WebLogic in this case) and the remote database. This requires Exadata to be pre-configured to allow SDP based access, in addition to TCP-IP. Oracle's existing OCI "Thick" JDBC Driver already has support for SDP, but WebLogic is only certified and optimized for use of the Thin JDBC Driver.

The Java Virtual Machine

- The JVM provides the common SDP implementation and Java API that the JDBC Driver invokes, under the covers, to communicate with the Exadata database over InfiniBand. An operating system's TCP/IP network software stack includes some capabilities that InfiniBand already caters for at a lower level (e.g. message loss avoidance). As a result, the latency added by such redundant TCP/IP features, can be avoided. SDP is a highly efficient stream sockets based wire protocol used for more efficiently controlling Infiniband's Remote Direct Memory Access (RDMA). With RDMA, one compute node writes, over-the-wire, to the memory of another target compute node, with only minor intervention required by either compute node's processors. By using SDP, lower latency and higher throughput is achieved for JDBC interactions between the application server and the database. RDMA frees up more of the compute node's CPU to perform other work.

Oracle Linux and Oracle Solaris

- Where SDP is used by the JDBC Driver and The JVM, larger Message Transmission Units (MTU) are adopted (e.g. 128k in many cases) to enable more efficient transmission of fewer, larger packets, between WebLogic on Exalogic and Oracle RAC on Exadata.

Conclusion

The need for enterprise IT organizations to provide next-generation cloud features such as elastic capacity while meeting ever more demanding performance and reliability requirements is driving demand for a new approach to infrastructure. Whether workloads are Web-based or thick-client, whether data-intensive or processing-intensive, whether homogeneous or highly heterogeneous, the key to success is hardware and software engineered together for

performance, reliability, and scale. Building or using custom, special purpose systems for different applications is wasteful and expensive. Oracle Exalogic Elastic Cloud, the world's first and only integrated middleware machine, dramatically surpasses alternatives and provides enterprises the best possible foundation for running applications.

Oracle Exalogic Elastic Cloud Software is a specific set of enhancements made to the core Oracle Fusion Middleware products to optimize their performance on Exalogic. The Oracle Exalogic Elastic Cloud is designed for maximum performance and reliability and most applications will experience a significant performance improvement even when only a subset of the optimizations that exist at different layers in the system, are leveraged.



ORACLE EXALOGIC ELASTIC CLOUD
SOFTWARE: TECHNICAL OVERVIEW

March 2011

Author: Paul Done

Contributing Authors: Michael Palmeter, James
Bayer

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



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

Copyright © 2011, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. UNIX is a registered trademark licensed through X/Open Company, Ltd. 1010

Hardware and Software, Engineered to Work Together