An Oracle White Paper
May 2011

Debunking the NoSQL Hype

ORACLE

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 NoSQL database space is getting pretty crowded and includes Cassandra, MongoDB, Voldermort, CouchDB, Redis, and SimpleDB.[1] Much of the hype behind these databases is due to the involvement of well known large internet companies like Facebook, LinkedIn, Amazon.com and others. The key attributes of these databases is that they are supposed to be "internet scale" with "elasticity", but they also have significant limitations and tradeoffs. With many rapidly evolving alternatives in the NoSQL space, providing a paper that compares them will be out of date as soon as it is published, but some blogs have.[2] [3] [4] Performance comparisons are similarly premature, but the numbers that are published aren't very impressive.[5] Comparisons in the related area of public clouds are also premature, but also raise questions.[6] [7] Before making a strategic decision to use a NoSQL database, you should take many issues into consideration.

There is currently no clear leader among the NoSQL databases. The market is getting fragmented, which is a problem for open source because you need a critical mass of developers to succeed. With so many choices with new options being regularly introduced, the NoSQL databases are beginning to feel like an ice cream store that entices you with a new flavor of the month. However, you shouldn't get too attached to any of the flavors, because it may not be around for that long. The Free Open Source Software (FOSS) databases have evolved from Berkeley DB, to MySQL, to manual functional sharding of MySQL, to Dynamo clone or big-table clones that feature some form of automated sharding. Interestingly, each of these FOSS databases features their own non-standard interfaces that require significant investment to use and commit to. Due to the specialization, a company might have to install more than one of these databases. Some

[1] http://nosql-database.org/ ; All URL references in this document were accurate as of April 2011.

[2] http://www.metabrew.com/article/anti-rdbms-a-list-of-distributed-key-value-stores

[3] http://randomfoo.net/2009/04/20/some-notes-on-distributed-key-stores

[4] http://blog.boxedice.com/2009/07/25/choosing-a-non-relational-database-why-we-migrated-from-mysql-to-mongodb/

[5] http://www.brianfrankcooper.net/pubs/ycsb.pdf

[6] http://www.cs.berkeley.edu/~kraska/talks/cloudbench_ms.pdf

[7] http://people.csail.mit.edu/tromer/papers/cloudsec.pdf

describe the NoSQL market as monopolistically competitive where NoSQL firms are doomed to make zero economic profit in the long term.[8]

NoSQL databases are often key-value stores, where the values are often "schemaless" blobs of unstructured or semi-structured data. This design allows NoSQL databases to partition their data across commodity hardware via hashing on the key. As a result, NoSQL databases may only have partial or no sql support. For example, group-by and join operations are typically performed at the application layer, resulting in many more messages and round trips compared to performing a join on a consolidated relational database. To reduce these round trips, the data may be denormalized such that a parent-child relationship can be stored within the same record. For hierarchical data, such denormalization can be acceptable, but not for complex data relationships. As the data is partitioned either functionally or horizontally, integrity of the data cannot be checked at the database level. There are no uniqueness, no foreign key, no referential and no integrity constraints. Features such as secondary indexes, transactions, triggers, sequences, ad-hoc queries (especially those for business intelligence) and support for database-level programming languages (like PL/SQL) are often sacrificed. Each NoSQL database will have its own peculiarities; with MongoDB, for example, one company avoided long column names due to the schemaless implementation, avoided per-customer databases due to space considerations, discovered unexpected locking and blocking issues, and found problems recovering from corruptions.[9] Although the NoSQL databases are evolving, some of their limitations are fundamental to their scalability model. Furthermore, the enhanced NoSQL database versions are often not backwardly compatibile.[10]

One Size Fits All *versus* Tools in a Toolbox

"One size fits all" has been criticized by FOSS or small startups, because FOSS can't do it. The NoSQL mantra of "use the right tool from the toolbox" may backfire as it is resulting in a lot of fragile, special-purpose tools, none of which is complete enough to provide full functionality. When you don't have the right tool with the right feature, you claim you don't need it (e.g., indexing or secondary indexes.) However, developers of the tools are busily adding features as quickly as they can because in actuality, none of these tools is sufficient for general-purpose deployments. You can use a Phillips screwdriver as an Allen wrench, but it's not recommended. Frequently, multiple NoSQL solutions are required to

[8] http://queue.acm.org/detail.cfm?id=1961297

[9] http://blog.boxedice.com/2010/10/23/on-shortened-field-names-in-mongodb/

[10] http://sujitpal.blogspot.com/2010/11/cassandra-dao-with-pelops.html

be deployed because each one is best suited for a specific set of use cases. One well known company is using four NoSQL databases and paying a high "pioneer tax."[11] Each of your NoSQL deployments needs its own set of expertise and its own set of machines. Each NoSQL database typically has its own non-standard API which requires changing the applications if the NoSQL database doesn't satisfy their needs or if a new NoSQL database comes along. Even with the plethora of NoSQL databases, you may not find one that fits your needs, and you need to implement your own.[12] You'll probably need to migrate to different NoSQL databases.[13] Do you want to be in the database development business?

The "One size fits all" phrase is really a vague reference to the Oracle database. Yet the Oracle Database is still the gold standard, including for the trivial key-value use case. Oracle has gone through many previous threats, including those that are now making a NoSQL resurgence such as object databases, hierarchical databases, document/XML databases, columnar databases, graph databases (e.g., RDF support), and network databases. The relational model is general enough to have been able to absorb the best of these ideas, and Oracle has been around long enough to incorporate them all. In addition, Oracle has all the bells and whistles including ETL, proven application development model, a full ecosystem of tools (ETL, BI, monitoring, administration, testing, etc.) high availability, performance, security, and interoperability. It's misleading that some of the papers on NoSQL databases use MySQL as the representative relational database, and then point out the scalability or functional limitations of the database. Oracle's supporting ecosystem did not materialize instantaneously. Oracle has been developing database-centric software for thirty years. Building any database is hard, to build one as good as Oracle's takes a long, long time.

Scalability

The NoSQL databases scale out by spreading their data across commodity machines, evoking visions of Google-like deployments of warehouses of machines. With a ton of machines comes a ton of potential failures. Google has presented statistics that a cluster with 1800 machines should expect an overheating incident where you have to power down machines, a power distribution unit failure, 20 rack failures of 40-80 machines,

[11] http://techblog.netflix.com/2011/01/nosql-at-netflix.html

[12] http://www.slideshare.net/nkallen/q-con-3770885 , http://qconlondon.com/london-2011/presentation/Data+Architecture+at+Twitter+Scale

[13] http://ria101.wordpress.com/2010/02/24/hbase-vs-cassandra-why-we-moved/

1000 individual machine failures, thousands of hardware failures, and other failures within the first year.[14] Werner Vogels summarized earlier Google disk studies as "high disk utilization or age of the disk have no significant impact on the probability that it will fail. They did find a strong correlation between manufacturer/model and failure rates. Basically you get what you pay when you talk about disk reliability."[15] Public clouds are not immune to catastrophic failure.[16]

NoSQL databases require a specific application development model and depend on replication to tolerate hardware failures, major failures and machine churn (see below.) While in theory, NoSQL databases can be deployed on hundreds or thousands of machines, in practice the deployments are much smaller and can typically easily fit within a single Oracle database with fewer and more reliable components. The largest production Cassandra cluster has over 100 TB of data in over 150 machines.[17]

Scale-out also requires the ability to gracefully add and remove sites to the distributed database environment. Thorough testing of this functionality is recommended as it may not perform to expectations.[18] Such sharding issues have caused an outage at Foursquare.[19] The scale-out approach is promoted as a way to handle unpredictable or exponential business growth or needs, but scale-out is often easier said than done due to the number of moving parts. It's typically hard to instantaneously build a new warehouse. Using a public cloud raises security and SLA concerns. Public clouds may also be temporarily nearing their capacity. "Cloudbursting" of less important applications to a public cloud requires rearchitecting existing application

In contrast, you could fit your hardware to the needs of the database. In just the last couple of years, new performance capabilities have revolutionized the scalability that database systems can achieve. These include the use of very large, multi-terabyte flash memories, compression which can dramatically increase the amount of data cached in memory and the speed of disk scans for large queries, and the movement of select database logic into storage to speed functions such as row selection, compression, encryption, and more. For Oracle, this has culminated in Exadata which is a pre-packaged software and hardware product incorporating all of these new innovations in an

[14] http://hosted.mediasite.com/mediasite/Viewer/?peid=1330ca0a008f4394917c2b7eb3163f1b1d

[15] http://www.allthingsdistributed.com/2007/02/on_the_reliability_of_hard_dis.html

[16] http://aws.amazon.com/message/65648/

[17] http://cassandra.apache.org/

[18] http://www.brianfrankcooper.net/pubs/ycsb.pdf

[19] http://highscalability.com/blog/2010/10/15/troubles-with-sharding-what-can-we-learn-from-the-foursquare.html

architecture that embraces the scale-out philosophy to effectively provide unlimited scalability for all types of applications. [20]

Application Development Models

Using a NoSQL database typically implies using a fundamentally different application development model than that of the traditional 3-tier architecture. Hence, an existing 3-tier application cannot be simply converted to NoSQL databases. They'd have to be rewritten. It is not easy to re-architect your systems to not run join queries, or not rely on read-after-write consistency.[21] [22]

Randy Shoup[23] from EBAY succinctly outlined the top best practices for this approach to scalable web sites:

1. partition everywhere
2. embrace asynchrony
3. automate everything
4. everything fails
5. embrace inconsistency

Pat Helland[24] formalized this philosophy of building large-scale applications when he "introduced and named a couple of formalisms emerging in large-scale applications:

- *Entities* are collections of named (keyed) data which may be atomically updated within the entity but never atomically updated across entities.
- *Activities* comprise the collection of state within the entities used to manage messaging relationships with a single partner entity.

Workflow to reach decisions functions *within* activities *within* entities. It is the fine-grained nature of workflow that is surprising as one looks at almost-infinite scaling."

Pat Helland[25] further embraced apologies as a way for code and businesses to deal with this asynchronous processing and the resulting inconsistencies with the real world. The

[20] http://www.oracle.com/technetwork/database/hadoop-nosql-oracle-twp-398488.pdf
[21] http://media.amazonwebservices.com/Netflix_Transition_to_a_Key_v3.pdf
[22] http://perfcap.blogspot.com/
[23] http://qconsf.com/dl/qcon-sanfran-2010/slides/RandyShoup_MoreBestPracticesForLargeScaleWebSitesLessonsFromEBay.pdf
[24] http://www.ics.uci.edu/~cs223/papers/cidr07p15.pdf
[25] http://blogs.msdn.com/b/pathelland/archive/2007/05/15/memories-guesses-and-apologies.aspx

NoSQL development model is designed to tolerate partial failures of hardware and temporary loss of data. Yet, many of the details behind this application development model remains part of the "secret sauce" of large internet sites, unlike the well documented, mature 3-tier application development model. Applications for large internet sites often evolve on a continuous basis.

James Hamilton of Amazon.com loves eventual consistency but there are some applications that are much easier to implement with strong consistency.[26] SimpleDB users have complained about not being able to rely on querying shortly after writing to SimpleDB; retries may be required to find the writes due to eventual consistency.[27]

The 3-tier model is exemplified by the Oracle stack. Oracle Fusion Applications are layered on Oracle Fusion Middleware and the Oracle database. Thousands of the largest and most mission critical systems in the world today have been running on Oracle RAC and ASM technology with great success for years. One of the growing trends within the customer usage is to consolidate many corporate databases into fewer and fewer databases. Consolidation reduces the overall administration overhead of a data center. Furthermore, consolidated applications can more effectively integrate into more differentiated joint services, allowing customers to take better advantage of their crown jewels - their data. Hardware such as Exadata can handle these large databases. Large internet sites have also used the Oracle database in large installations - SalesForce.com uses a multi-tenant Oracle architecture and memcache-d to achieve scalability.[28] Clearly the traditional 3-tier application development model scales well and the desired development model for established non-internet companies.

Replication and the CAP Theorem

All the NoSQL databases use replication to provide their high-availability solution. The CAP theorem is often used to justify the replication design of the NoSQL databases.[29] The CAP theorem is often over simplified to "Choose 2 out of 3: Consistency, Availability, Partition tolerance." The use of the CAP algorithm to justify using weak consistency to tolerate partition tolerance has sparked a debate within the research

[26] http://perspectives.mvdirona.com/2010/02/24/ILoveEventualConsistencyBut.aspx

[27] http://buzzpressure.com/2008/07/29/eventual-consistency-eventually-a-pain-in-the-ass/

[28] http://hosted.mediasite.com/mediasite/Viewer/?peid=8810d5467ba54623b2c97e92038d36b41d

[29] http://queue.acm.org/detail.cfm?id=1466448

community. Some of CAP theorem directly feeds into the application development philosophy discussed above, especially embracing inconsistency and apologies. Yet, not everybody in the community is in complete agreement. For example, Jeff Dean from Google viewed building simple abstractions to build applications on top of weakly consistent storage systems one of his top challenges for future work because using weak consistency is difficult.[30]

Optimizing your design for partition tolerance is often not a good tradeoff. Michael Stonebraker[31] [32] has pointed out that there are many types of failures including:

1. Application errors.
2. Repeatable DBMS errors.
3. Unrepeatable DBMS errors.
4. Operating system errors.
5. A hardware failure in a local cluster.
6. A network partition in a local cluster.
7. A disaster.
8. A network failure in the WAN connecting clusters together.

Daniel Abadi[33] has pointed out that many NoSQL databases actually use PACEL - "if there is a partition (P) how does the system tradeoff between availability and consistency (A and C); else (E) when the system is running as normal in the absence of partitions, how does the system tradeoff between latency (L) and consistency (C)?"

Some NoSQL databases now support both strong and weak consistency models.[34] [35] Like other features, justifications are being used to justify the lowest-common denominator feature (e.g., weak consistency replication), but support for more advanced features like replication with strong consistency guarantees are added as the NoSQL database area matures. The trade-offs of when to use strong or weak consistency are not always clear.[36] Cassandra's eventual consistency model wasn't a good match for Facebook's new real-time Messages product.[37]

---

[30] http://hosted.mediasite.com/mediasite/Viewer/?peid=1330ca0a008f4394917c2b7eb3163f1b1d

[31] http://cacm.acm.org/blogs/blog-cacm/83396-errors-in-database-systems-eventual-consistency-and-the-cap-theorem/fulltext

[32] http://en.wikipedia.org/wiki/Michael_Stonebraker

[33] http://dbmsmusings.blogspot.com/2010/04/problems-with-cap-and-yahoos-little.html

[34] http://aws.amazon.com/articles/3572?_encoding=UTF8&jiveRedirect=1

[35] http://wiki.apache.org/cassandra/ArchitectureOverview

[36] http://www.cidrdb.org/cidr2011/Papers/CIDR11_Paper15.pdf

[37] http://www.facebook.com/note.php?note_id=454991608919#

The Oracle database has a more complete high-availability solution. While replication is the single high availability solution for most NoSQL databases, Oracle dynamically switches to different solutions optimized for different failure modes:

- Fast-Start Failover and Fast-Application Notification (FAN) for computer failures.
- FAN, Data Guard and Automatic Storage Management (ASM) for Storage failures.
- Automatic validation of redo blocks before they are applied and fast failover to an uncorrupted standby database upon production database corruption to handle data corruption.
- Fast-Start Failover and Fast-Application Notification (FAN) for site failure.
- Real Application Clusters for instance failures.

These Oracle features all provide high-availability without the threat of inconsistency. Quorum and database reconfiguration is quick and automatic. The CAP theorem doesn't even apply to most of these high availability features because they avoid the distributed "split brain" problem. Of course, Oracle also has a variety of asynchronous replication solutions including Oracle GoldenGate.[38]

The CAP theorem doesn't apply to high availability approaches of many ACID database systems.[39] Fully ACID systems are PC/EC in PACELC. They refuse to give up consistency, and will pay the availability and latency costs to achieve it.[40] However, the availability and latency costs are low in traditional highly available systems. Let's look at mirrored disks as an example. Storage systems implement mirrored disks by writing both disks and reading from one. If a disk fails, writes are unavailable until the failed disk is detected so that no further writes will be issued to the failed disk. Unavailability of reads is confined to some timeout where the system will reissue reads to the good disk. The real-world engineering issue then becomes how quickly the failure can be detected and communicated to all the clients or all the other write stores, and how quickly they can agree on voting the failed write store out of the configuration. In practice, if the failure detection and quorum reconfiguration can be kept relatively fast, then there is no need to give up any consistency at all. The argument for eventual consistency then becomes that quorum reconfiguration takes too long. Telephone companies claim that the

[38] http://www.oracle.com/us/products/middleware/data-integration/goldengate11g-realtime-wp-168153.pdf?ssSourceSiteId=otnen

[39] http://voltdb.com/blog/clarifications-cap-theorem-and-data-related-errors

[40] http://dbmsmusings.blogspot.com/2010/04/problems-with-cap-and-yahoos-little.html

reconfiguration of their internal databases happen in single digit milliseconds. Storage arrays reconfigure failed disks in seconds or even less than a second. Database reconfiguration happens in seconds or small numbers of minutes. These numbers can all be driven down by engineering and configuration attention. It seems like the use case of an eventually consistent architecture is really systems that cannot tolerate seconds of downtime and are willing to go through the trouble to build tolerance for inconsistency into the application to avoid seconds of downtime.[41]

Schemaless Data Models

Once again, NoSQL databases try to spin a mis-feature as a feature in the case of "schemaless" data. Schemaless data can be emulated quite easily and effectively within an Oracle database. For example, Salesforce.com has Flex schemas implemented with a huge table with 500+ varchar2 columns. Indexed datatype-specific denormalized accelerator tables (also known as pivot tables) in Oracle are synchronously maintained.[42] Yet, while this is flexible from a developer's point of view, data is harder to query. Tom Kyte, a well-known Oracle DBA, called using the Entity-Attribute-Value (EAV) extensible schema his worst design decision ever.[43] The EAV model works for programmers who need only key-value queries. However, range queries or more complex queries as required for reporting are much more complicated and slower when using an EAV schema. EAV schema is really for frequently changing schemas. If your data model is relatively stable, then you should not use EAVs.

The problem of dealing with schemaless data is actually pushed to the application layer. The application still needs to know what information is stored where and how. As data evolves, the application must deal with all the differing formats. XML, protocol buffers or other self-describing data methods are used.

Key-value models tend to be denormalized to reduce the number of joins. Denormalized data models are more expensive to update and keep logically consistent, and work best if the data is read-only.

Mature full-function databases like Oracle support online schema changes and provide support for application upgrades. Oracle support includes:

---

[41] http://www.allthingsdistributed.com/2008/12/eventually_consistent.html

[42] http://www.salesforce.com/au/assets/pdf/Force.com_Multitenancy_WP_101508.pdf

[43] http://tkyte.blogspot.com/2009/01/this-should-be-fun-to-watch.html

- rolling upgrades using RAC or using DataGuard.[44]
- many in-situ online DDL[45]
- out-of-place reorganizations of DDL[46]
- Edition-based redefinition for application upgrade support[47]

Some of the less mature NoSQL databases may not gracefully adjust to schema changes. For example, prior to release 0.7 of Cassandra in early 2011, Cassandra required a restart when you changed the column family definition.[48] [49]

ACID *versus* BASE

ACID stands for Atomicity, Consistency, Isolation, and Durability. BASE stands for Basic Availability, Soft-state, Eventual consistency.[50] The ACID vs. BASE argument derives from many of the issues touched on above. The CAP theorem is a poor justification, as it doesn't apply to much of the modern database high availability techniques.

Some of the tradeoffs aren't always clear or spelled out. In the default configuration of Cassandra, writes may be acknowledged immediately and the CommitLog is synced every 10 seconds. Replication reduces the probability of losing data from a failure after writing the log entry but before it actually reaches the disk. However, data can be lost unless you configure Cassandra to be in batch mode where Cassandra won't acknowledge writes until the commit log has been fsynced to disk. These commit log configurations affect performance. MongoDB has similar considerations.[51] VoltDB relies on persistence

[44] http://www.oracle.com/technetwork/database/features/availability/maa-wp-10gr2-rollingupgradebestprac-1-132006.pdf

[45] http://download.oracle.com/docs/cd/E14072_01/server.112/e10595/indexes003.htm#i1006652

[46] http://download.oracle.com/docs/cd/E14072_01/appdev.112/e10577/d_redefi.htm

[47] http://www.oracle.com/technetwork/database/features/availability/edition-based-redefinition-1-133045.pdf

[48] http://wiki.apache.org/cassandra/DataModel

[49] http://www.datastax.com/dev/blog/whats-new-cassandra-07-live-schema-updates

[50] http://queue.acm.org/detail.cfm?id=1394128

[51] http://groups.google.com/group/akka-dev/msg/f5e8173fc1af7621

through k-safety replication.[52] The administrators of these systems may not even be aware of this issue - until too late.[53] Dealing with eventual consistency can be difficult.

Other Considerations

NoSQL databases typically have their own non-standard API that you'll have to use in your application. If the NoSQL database doesn't meet your needs, you'll need to port your application to another database. SQL is a standard supported by many databases.

Power consumption is now a major issue in data centers.[54] [55] NoSQL databases are really an inefficient way of getting results as they rely upon power-hungry brute force methods to get results for complex queries. A data center is required when one Exadata and one Exalogic machine will suffice.

The NoSQL database has been sponsored by large internet companies. Such sponsorship has added credibility to internet scale claims of the databases and an aura of "coolness." The large internet companies have been motivated by having free database software that is under their complete control. These companies have hired top developers for their scalable database implementations and to keep them running. Is your company prepared to make such an investment in an area which is not your company's core competence? Do you really want to be contributing to an open source effort? Do you really want to bet your project on something that's on the bleeding edge? Do you want to build the superstar team required to make the NoSQL vision a reality?[56] You are not Google.

NoSQL databases are not free to operate. Each NoSQL Databases come with the cost of hardware, administration of the hardware, and support of the software. One NoSQL database may not be enough. Renting machines from a public cloud is likely to be more expensive than running your own data center.

Consider security. Most NoSQL databases assume that they are protected behind a firewall and have only recently begun to address security issues. On the other hand, Oracle is used in critical, high-security installations. Security has been a major focus of

---

[52] http://voltdb.com/content/how-voltdb-works

[53] http://groups.google.com/group/mongodb-user/browse_thread/thread/528a94f287e9d77e

[54] http://www.slideshare.net/infoblog/cidr-2009-james-hamilton-keynote

[55] http://hosted.mediasite.com/mediasite/Viewer/?peid=1330ca0a008f4394917c2b7eb3163f1b1d

[56] http://techblog.netflix.com/2011/01/nosql-at-netflix.html

Oracle for years. Audit Vault[57] and Data Vault[58] are just two of the database security products.

Conclusion

This paper covered a lot of ground. A common theme is that the NoSQL databases are currently not mature. There's no clear winner right now, and new NoSQL databases are on their way. Although the claim is to use different databases for different use cases, these NoSQL databases are adding features as fast as they can and overlap is inevitable. However, compared to relational databases, their feature set is primitive. The database deployments typically aren't very large in terms of data or challenging in terms of performance. Their number of deployments is relatively small. Their application development model is challenging, adding to the complexity of the implementation. Their high availability and SLAs need evaluation.

Go for the tried and true path. Don't be risking your data on NoSQL databases.

[57] http://www.oracle.com/technetwork/database/audit-vault/overview/index.html

[58] http://download.oracle.com/docs/cd/E11882_01/server.112/e16544/toc.htm

Debunking the NoSQL Hype
May 2011
Author: Alan Downing

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