Data Grids and Service-Oriented Architecture

An Oracle White Paper
Updated May 2007

ORACLE

Service-oriented architecture (SOA) provides a means of integrating disparate applications within an enterprise, improving reuse of application logic while eliminating duplication of production environments.

INTRODUCTION

Service-oriented architecture (SOA) provides a means of integrating disparate applications within an enterprise, improving reuse of application logic while eliminating duplication of production environments. An SOA avoids silos of disconnected information within the enterprise that make it difficult to service customers, meet production demands, and manage large volumes of information. Developing an SOA that guarantees service performance, scalable throughput, high availability, and reliability is both a critical imperative and a huge challenge for today's large enterprises.

The increasing rate of change in the modern business environment demands greater agility in an organization's technology infrastructure, which has a direct impact on data management. SOA offers the promise of less interdependence between projects and, thus, greater responsiveness to business challenges. But it also raises many questions for enterprise architects:

- How will data access services be affected by the increasing number of services and applications that depend on them?
- How can I ensure that my services don't fail when underlying services fail?
- What happens when the database server reaches full capacity? And how can I ensure the availability of reliable data services even when the database becomes unavailable?

When choosing an SOA strategy, corporations must rely on solutions that ensure data availability, reliability, performance, and scalability. They must also avoid "weak link" vulnerabilities that can sabotage SOA strategies.

A data grid infrastructure, built with clustered caching, addresses these concerns. It provides a framework for improved data access that can create a competitive edge, improve the financial performance of corporations, and sustain customer loyalty.

This paper looks at the challenges of selecting an SOA strategy, how an SOA can improve data availability and reliability, and how clustered caching can improve SOA performance and ensure scalability for very large-scale transaction volumes.

SOA CHALLENGES

The following should be taken into consideration when selecting an SOA strategy:

The Structure of an SOA Environment

In an SOA environment, there are several types of components to consider. In order of increasing consolidation, these can be grouped into data services, business services, and business processes. Data services provide consolidated access to data. Business services contain business logic for specific, well-defined tasks and perform business transactions via data services. Business processes coordinate multiple business services within the context of a workflow.

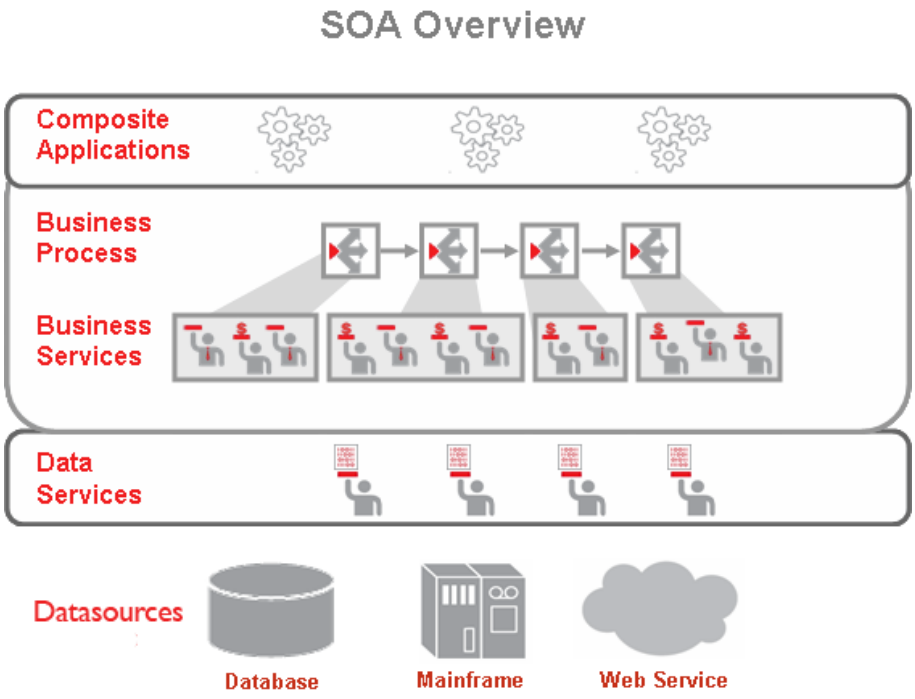

Figure 1: SOA environments typically comprise three types of components: data services, business services, and business processes.

Data within an SOA generally falls into one of two categories:

- **Conversational state** – The conversational state is managed by business services and processes and corresponds to currently executing operations, processes, sessions, and workflows.
- **Persistent data** – Persistent data is managed by data services and is usually stored in databases.

Consolidation of Data Services Raises Scale and Performance Issues

The role of data services is to provide access to enterprise data by expressing the data in terms of the business without requiring any external knowledge of how the data is actually managed. The value of data services lies in the consolidation that

they bring, allowing centralized control of data without the proliferation of data silos throughout the enterprise. Unfortunately, this centralization also brings significant scalability and performance challenges. Scalability issues arise when many business services depend on a single data service, overwhelming back-end datasources. Performance issues result directly from scalability limitations, because poorly scaling data services will become bottlenecks and requests to those services will queue. Performance is also influenced significantly by the granularity of an SOA data service, which often provides either too little or too much data. Data services built around a specific use case will provide too much data for simpler use cases, and more-complex use cases will need more data, resulting in more service invocations. In either case, performance will be affected, and with application service level agreement (SLA) requirements moving toward response times measured in milliseconds, every data service request can represent a significant portion of the application response time.

Reliability and Availability Can Be Compromised by Complex Workflows

If a business process depends on six services, each of which achieves 99 percent uptime, the business process itself will have a *maximum* of 94 percent uptime, meaning more than 500 hours of unplanned downtime each year.

Reliability and availability may also be affected. As business services are integrated into increasingly complex workflows, the added dependencies *decrease* availability. If a business process depends on several services, the availability of the process is actually the product of the weaknesses of all the composed services. For example, if a business process depends on six services, each of which achieves 99 percent uptime, the business process itself will have a *maximum* of 94 percent uptime, meaning more than 500 hours of unplanned downtime each year.

SOA Environments Differ from Traditional User-Centric Applications

Conversational state, such as the hypertext transfer protocol (HTTP) session state utilized by Web services, is often short-lived, rapidly modified, and repeatedly used. The life span of the data may be a matter of seconds, spanning a dozen requests, each of which may need to read or update the data. Moving from traditional user-centric applications to an SOA environment means that, in addition to users, machines are now accessing services—at machine speed. This means that the "user count" increases dramatically while the average "think time" decreases to almost nothing, causing the maximum sustained request rate to *far* exceed the original specification. The result is that technologies that were capable of handling traditional user loads are almost inevitably crushed by the increased load associated with an SOA deployment.

Ensuring the reliability and integrity of conversational state is critical, but its rapid churn rate and transient nature make it particularly difficult to manage by traditional means. Using database servers is the traditional solution for scalable data services, but they cannot cost-effectively meet the throughput and latency requirements of modern large-scale SOA environments. Most in-memory solutions depend on compromises such as queued (asynchronous) updates, master/slave high-availability (HA) solutions, and static partitioning to hide scalability issues, all at the

cost of substantially reduced reliability and scalability. Most SOA vendors go as far as to strongly recommend avoiding stateful services if at all possible, due to these scaling and performance challenges.

DATA RELIABILITY AND AVAILABILITY IN AN SOA

The stakes for data reliability and availability in mission-critical environments are high: crucial business decisions, financial results, customer satisfaction, employee productivity, and a company's reputation all depend on it.

SOA Demands High Data Reliability and Availability

Making sure that services have a consistent, coherent view of data is critical to ensure reliability and availability. Transactionally consistent data services are essential for scalable, reliable data processing. Products used to manage data must have data integrity "in their genes," supporting both optimistic and pessimistic transactions, synchronous server redundancy, and reliable configurations.

Data-management products for SOA must prioritize availability and reliability over features, because SOA adoption results in enterprise systems that are *more* prone to outage as the number of service dependencies increases.

Data-management products for SOA must prioritize availability and reliability over features, because SOA adoption results in enterprise systems that are *more* prone to outage as the number of service-dependencies increases. This is the natural consequence of compositional complexity and represents an engineering trade-off resulting from the elimination of application silos. This risk becomes further pronounced as systems are consolidated, because service interruptions will have an increasingly greater impact on the organization.

Eliminating Single Points of Failure

SOA introduces a set of new challenges to the continuous availability of complex systems, but the solutions for both *service* and *system* availability are well understood and proven. Service availability requires the elimination of all single points of failure (SPOFs) within a given service and the insulation—to the maximum extent possible—against failures in the service's natural dependencies. System availability requires similar insulation from the failure of services on which the system depends.

When architecting a service for high availability, it is necessary to ensure that the service host itself is highly available.

Clustering is accepted as the standard approach to increasing availability, but in a traditional clustered architecture, adding servers to a cluster will *decrease* its reliability even as it *increases* its availability. There are several reasons for this, including the likely interruption of service during failover and failback and the increased incidence of server failures in direct proportion to the total number of servers.

Static Partitioning Does Not Increase Data Availability

To achieve scalability, other solutions use static partitioning across a collection of primary servers, each with its own dedicated backup server to ensure availability, but this model is fundamentally crippled:

- Static partitioning makes the service unable to dynamically increase capacity, meaning that it cannot participate in a capacity-on-demand architecture.
- Static partitioning requires massive overprovisioning to prevent peak loads from overwhelming the service.
- Reliance on dedicated backup servers means that the cluster heals much more slowly—or may not heal at all—when a primary server dies and thus increases the window of opportunity for catastrophic data loss by allowing an SPOF to remain within a production environment.
- Static partitioning with dedicated backup tends to make failback processing much more difficult, if not impossible.
- Using dedicated backups for each of the primary servers can significantly increase infrastructure costs and doubles the required number of servers by employing an N+N availability strategy instead of an N+1 strategy.

Clustered Caching Ensures Reliability and Availability

Oracle Coherence is a trusted in-memory data management solution for ensuring reliability and high availability for Java-based service hosts, such as Java Platform, Enterprise Edition (Java EE) application servers.

Oracle Coherence is a trusted in-memory data management solution for ensuring reliability and high availability for Java-based service hosts, such as Java Platform, Enterprise Edition (Java EE) application servers. It makes sharing and managing data in a cluster as simple as on a single server. It accomplishes this by coordinating updates to the data by using clusterwide concurrency control, replicating and distributing data modifications across the cluster by using the highest-performing clustered protocol available, and delivering notifications of data modifications to any servers that request them.

Oracle Coherence, which provides replicated and distributed (partitioned) data management and caching services on top of a reliable, highly scalable peer-to-peer clustering protocol, has no SPOFs. It automatically and transparently fails over and redistributes its clustered data management services when a server becomes inoperative or is disconnected from the network. When a new server is added or when a failed server is restarted, it automatically joins the cluster and Oracle Coherence fails services back to it, transparently redistributing the cluster load. Oracle Coherence includes network-level fault-tolerance features and transparent soft-restart capabilities to enable servers to self-heal.

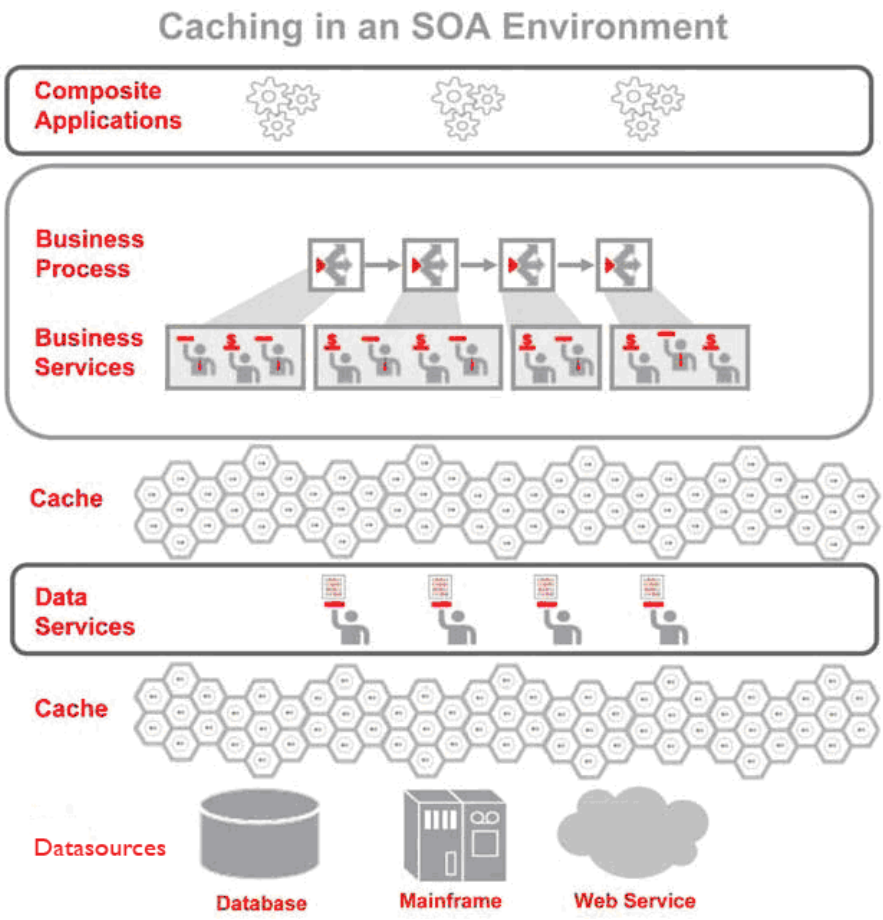

Figure 2: Caching is used to decouple components, yielding increased performance, throughput, and reliability.

Oracle Coherence provides a fully reliable in-memory data store for services, transparently managing server faults and making it unnecessary for the service logic to deal with complicated leasing and retry algorithms.

Without Oracle Coherence, the servers and the service processes that run on those servers each represent an SPOF. With Oracle Coherence, a service is composed as an aggregation of all of those service processes on all those servers, achieving *resiliency by redundancy*. A well-designed service can survive a machine failure without *any* impact on *any* of the service clients, because Oracle Coherence provides continuous service availability, even when servers die. When architected with Oracle Coherence, even a stateful service will survive server failure without any impact on the availability of the service, without any loss of data, and without missing any transactions. Oracle Coherence provides a fully reliable in-memory data store for the service, transparently managing server faults, and making it unnecessary for the service logic to deal with complicated leasing and retry algorithms.

The Oracle Coherence dynamic mesh architecture increases both reliability and availability, by making failover and failback nearly instantaneous. Oracle Coherence illustrates the difference between simple *high availability* and true *fault tolerance*. Moreover, Oracle Coherence supports dynamic capacity on demand by expanding

its resilient data fabric to incorporate additional servers as soon as they come online.

State Management Through Virtualization

Fully stateless services (such as static-content HTTP servers) are very easy to manage for high availability, but very few services are actually stateless. Many services manage conversational state, and even those that do not will usually manage some state, such as caches, internally.

The key to achieving continuous availability and full reliability is to implement stateful services as if they were stateless by delegating all service state management to Oracle Coherence. If the service implementation is stateless, server failure will not be able to cause any loss, thus enabling another server to perform the necessary service request on behalf of the failed server.

Oracle Coherence provides the resilient and reliable state management on which these services are built, with true server location transparency and system fault tolerance. It manages the service state in a manner that completely and dynamically eliminates SPOFs and single points of bottleneck (SPOBs) and fully virtualizes the service state across any number of servers.

Transparent Data Partitioning Achieves Continuous Availability and Reliability

A major factor for service availability is ensuring that any service host can handle any request at any time. Failing to do this will diminish the ability of the service cluster to reliably respond to service requests while a failure is occurring. Not having fully transparent data partitioning means that

- Any delays during failover or failback will reduce reliability.
- Failures will occur *during* the failover process.
- Failures will occur during *failback,* if it is possible to fail back at all.
- Each service implementation will have to include custom fault detection and retry logic to recover from misdirects and redirects.
- Rebalancing after a server failure will be failure-prone or impossible.
- Reliable and dynamic expansion and contraction of the cluster will be impossible.

Oracle Coherence provides fully transparent data partitioning, and the resulting service implementations automatically achieve continuous availability and reliability. No custom fault detection or retry logic is required.

Avoiding Distributed Computing

A key differentiator for clustering products is whether they assume full responsibility for virtualizing the cluster or delegate the difficult responsibilities

back to the application developer. Support for clustering features such as transactions, "no lease" data access, and clusterwide locking is critical in enabling developers to implement cluster-safe services without needing to develop custom distributed computing algorithms for each business process. Oracle Coherence takes responsibility for handling server and network failures.

Failover and Failback

Oracle Coherence is designed for lights-out and zero-administration environments and employs self-healing network capabilities to not only survive failures but also repair them.

With the increasing complexity of SOA environments, automated and transparent failover and failback become even more critical. Ensuring that these transitions do not result in an interruption of service means that applications will be not only highly available but also highly reliable.

Oracle Coherence is designed for lights-out and zero-administration environments and employs self-healing network capabilities to not only survive failures but also repair them.

Insulation from Failures in Other Services

Any service that depends on other services must be able to compensate for the lack of availability of those services. This is particularly critical for services composed of other services.

The key to preventing failures in one service from affecting another is to appropriately decouple the two services. For services that support loose coupling, interposing read-through/write-behind caching between the consuming and producing services can provide an effective means of isolating reliability issues.

In many cases, it is possible for a service to use write-behind caching to continue operating—even when an underlying database is unavailable. In such a configuration, Oracle Coherence will queue the transactions for the database until the database is brought online and its transaction logs are recovered. This capability is absolutely crucial for continuously available systems, because system maintenance is inevitable.

SCALABLE PERFORMANCE VITAL IN SOA

Scalability and performance challenges in a services environment are similar in many ways to those faced in traditional applications. The less distance and transformation that is required for using a piece of information, the more efficient the application will be. However, the scale of the problems has been dramatically increased. Furthermore, in addition to the traditional challenges related to conversational state and data access, there is now the added element of increased "distance" between the various services that are working together to handle each incoming request.

Enormous Loads Challenge Scalability

Enterprise systems built on an SOA face a host of challenges relating to unprecedented scale. Request volume is growing in multiple dimensions at once.

There are more requests, more parameters, and more resulting data per request. Additionally, as business processes become increasingly automated, load can increase enormously. In financial services, the use of algorithmic trading systems has increased load in some cases by several orders of magnitude, and growth is anticipated to continue at a breakneck pace. Retailers are seeing similar swells from personalization and closed-loop analytics. The travel and hospitality industries are working to address the need for real-time pricing and the increased load generated by third-party inventory engines. Many other industries are experiencing similar exponential growth in service load.

The most fundamental challenge for large-scale, data-intensive systems is to provide multipoint access to shared data while preserving a true single system image (SSI). Oracle Coherence offers fully coherent caching in a clustered environment, achieving linear scalability on commodity hardware with a fixed worst-case latency. One of the major strengths of the Oracle Coherence peer-to-peer architecture is that it enables data to be pushed and/or pulled within a distributed environment, either on-demand or as the data is updated. It efficiently and directly moves data to where it is needed without depending on time-based invalidation or other artificial means of synchronization. This means that services have the full benefit of instantaneous data access *from any server* without the possibility of accidentally obtaining out-of-date data.

The Data Grid Agent feature in Oracle Coherence ensures ultra low transactional latency without compromising throughput or fault-tolerance. The next step—platform-portable invocation and data services—is a giant leap in Oracle's Fusion Middleware strategy. Together, these capabilities make high-volume, transactionally consistent data and event streams universally available to business services throughout the enterprise.

Large Transaction Volumes Handled Without Compromise

Oracle Coherence is well known for its singular ability to handle enormous transaction volumes (300,000+ transactions per second) for conversational state without compromising read performance or fault tolerance.

Oracle Coherence is well known for its singular ability to handle enormous transaction volumes (300,000+ transactions per second) for conversational state without compromising read performance or fault tolerance. Although there are clustering solutions that support scale-out or high availability, Oracle Coherence remains the only viable option for applications that need to sustain intense read/write data access without resorting to non-fault-tolerant techniques such as asynchronous updates.

Deployment Flexibility Through the Data Grid

Oracle Coherence enables capacity on demand in two key steps. First, it helps move conversational state out of the application and into the Oracle Coherence Data Grid. This enables requests to be routed to any application instance without the need for manual provisioning of data. Oracle Coherence's mesh architecture also means that additional application instances can be started on the fly, without the need for manual repartitioning of data and with minimal delay, because

application state is already prepared in the data grid. This compares admirably to products that depend on static partitioning and "buddy" replication for failover.

Second, the Oracle Coherence Data Grid is designed for lights-out management/zero administration (LOM/ZA), which provides the ability to expand and contract Oracle Coherence almost instantaneously in response to changing demand. The Oracle Coherence mesh architecture becomes increasingly nimble as the cluster size increases, with rebalancing occurring even faster and server failures having smaller and smaller impacts.

By shifting state into Oracle Coherence and using Oracle Coherence's dynamic mesh architecture to dynamically scale data management, applications can achieve near-real-time provisioning without risking loss or abortion of requests.

By shifting state into Oracle Coherence and using Oracle Coherence's dynamic mesh architecture to dynamically scale data management, applications can achieve near-real-time provisioning without risking loss or abortion of requests.

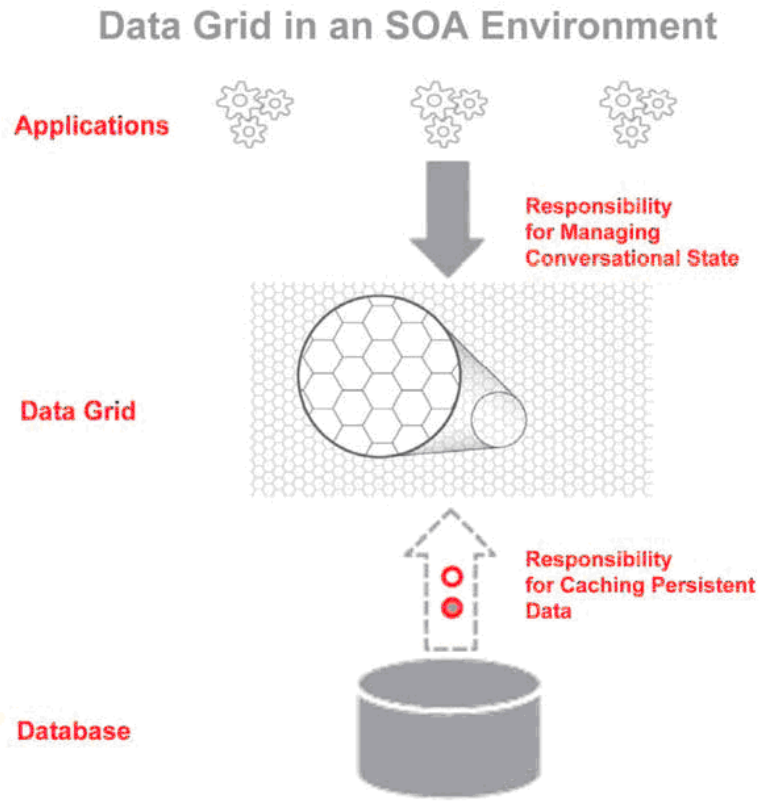

Figure 3: In an SOA environment, state management is the responsibility of the data grid, which easily and cost-effectively scales data access far beyond what can be achieved with a database and also delivers significantly better scalable performance and data integrity for conversational state. At the same time, the data grid results in a substantial increase in deployment agility.

Web Services Require Scalable Performance

SOAs and Web services in general exhibit the same requirements for scalable performance as any other line-of-business or outward-facing application. In the same way that advanced Web applications manage HTTP sessions to provide conversational state on the server on behalf of the user, Web services and other

SOA infrastructures often have to implement stateful conversations and workflow. In fact, many Web services implementations simply use HTTP sessions.

On a request-by-request basis, the data access requirements for Web services appear to be significantly higher than for Web applications, due to the nature of Web services, in which ancillary data is often included to eliminate the need for subsequent requests. In some cases, the request volumes are also significantly higher and growing at a much higher rate, largely because the service clients are no longer humans impeded by think time.

CONCLUSION

Clustered caching and data grid infrastructures ensure availability, reliability, and scalable performance for SOA. SOA environments are adopting these two technologies much more rapidly than earlier architectures, due to their combined value. As the recognized market leader in clustered caching, Oracle has been at the forefront of making SOA a reality, with Oracle Coherence already powering many of the world's largest and most demanding SOA environments.

Data Grids and Service-Oriented Architecture
Updated May 2007
Author: Jonathan Purdy

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