



**THE PROMISE AND CHALLENGE OF SOA**

# GemFire Enterprise

*Boosting performance of SOA applications*

## OVERVIEW

Enterprises are seeing continued operational and deployment benefits of Service Oriented Architectures (SOA). The ability to integrate existing applications in a loosely coupled infrastructure is compelling drivers to adopt SOA. Systems that appeared disconnected in the past are brought together under a service orchestration umbrella through SOA. However, this concept does place new requirements on IT infrastructure and on data management in particular. This paper examines the relevance of an Enterprise Data Fabric (EDF) such as GemStone's GemFire Enterprise in achieving the goals of a typical SOA implementation.

## THE PROMISE AND CHALLENGE OF SOA

As highlighted in Figure 1, SOA can be defined as an architecture that enables scalable (easy to add new members) and flexible (easy to integrate) deployment of composite applications that perform different business functions. It does this by orchestrating a well-defined set of shared services (typically Web Services).

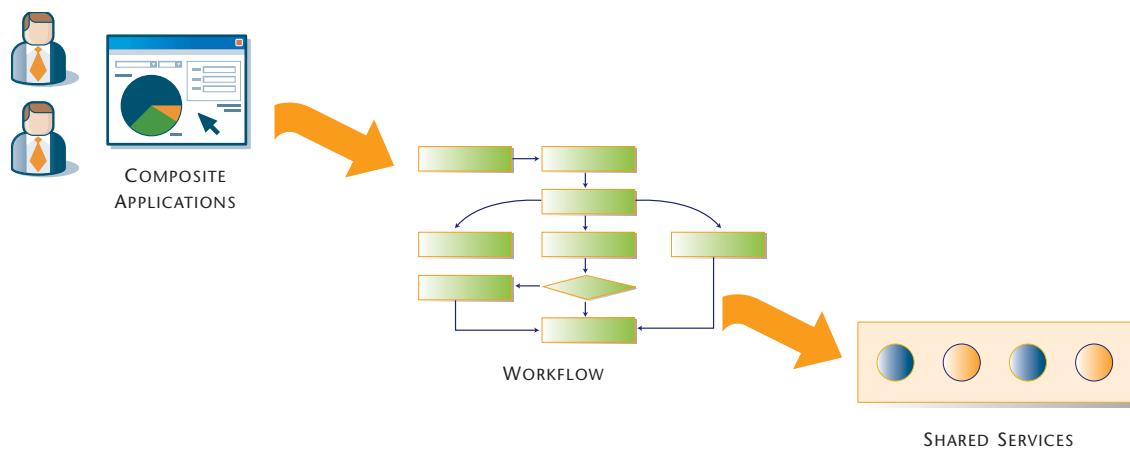


FIGURE 1: SOA FRAMEWORK

Through the Web Services paradigm, SOA achieves separation of application 'function' from application 'structure,' enabling workflows that make use of services with differing underlying structures or implementations. Such a capability becomes extremely important in most large organizations, where application, process and information integration needs to occur across heterogeneous platforms, and between new and legacy systems. Furthermore, SOA fosters a reusable development model where services become shared components that can be used in new application development, leading to significantly lower programming and maintenance costs. Most importantly, SOA makes possible a customer-centric application development model that aims to improve the customer experience and deliver complex Service Oriented Business Applications (SOBAs) through the simple orchestration of services or composite process.

While these benefits seem compelling, there are inevitable hurdles to overcome. These issues include:

- Unpredictable access loads on the individual services which introduce scalability and performance challenges and can cause non-compliance with Server Level Agreements (SLAs).
- Latencies in retrieving commonly used data which slow down individual services
- Latencies in performing Identification, Authentication and Authorization checks multiple times (once per step) in a composite process which slow down the overall composite process.
- Challenges in managing session state across a composite process without check-pointing each step in a database causing significant overall performance degradation.

Traditional architectures such as client-server systems were designed with a specific set of use-cases in mind, and are often unsuited for an SOA-based fluid application environment. In an SOA environment, clients accessing a system are completely decoupled from the system itself, which has no knowledge of the user access patterns or the duration of access. For instance, a client application might try to access a back end system during a planned downtime for that system. As new services and clients are added there can be an unexpected increase in data access volumes that may overwhelm existing systems. This affects system reliability and performance, leading to poorer Quality of Service (QoS) and non-compliance with Service Level Agreements (SLAs). Hardware upgrades can alleviate these problems to some extent, but are by no means a complete solution.

Traditional architectures also manage the overall end-to-end business process in a single application process state. This has a number of efficiencies that are lost in a composite process environment. Single process environments perform identification, authentication, and authorization only once at the beginning of a process. In a SOA environment each service must perform this work. Establishing a security context can be compute intensive relative to the rest of the work the service may perform. Single process environments share process state by simply passing a reference to data in memory from one step to the next. Unless a similar technique can be developed for SOAs, they are reduced to either passing large XML documents between steps (which can be computer and I/O intensive) or writing intermediate state to a database which adds the latency of disk I/O and created data management challenges if a process is abandoned before it is finished. This can be a frequent occurrence. Think about the number of times you started a process (such as ticketing) on a website, and then just didn't finish.

Another difficulty may be encountered with long-running business processes, where managing state across multiple Web Service calls presents challenges in dealing with session time-outs. XML is often the preferred data standard in SOA, mainly due to its elegance from an application interoperability standpoint. Unfortunately the flexibility comes with associated performance inefficiencies in storing, marshalling/unmarshalling and manipulating XML documents.

These challenges in installing and operating a Service Oriented Architecture are to a great extent a manifestation of the lack of a robust operational data management framework.

### THE RELEVANCE OF A MIDDLE-TIER DATA FABRIC

To achieve the real promise of Service Oriented Architectures, IT organizations need to move from a 'centralized' data management system, characterized by back-office databases and warehouses, to a more distributed middle-tier data fabric that provides a scalable way to access and distribute data across multiple services. Simply put, data must be accurate, timely, highly available, and instantly accessible to any business process. SOA environments require state-of-the-art data management capabilities:

- From a performance viewpoint, services need data locality-the proximity of data to computing resources-to speed up applications, avoid network bottlenecks, and to enable timely information aggregation.
- From an accessibility viewpoint, services running on any enterprise node must be able to discover and access data from any other node in a location-independent manner. SOA applications should be completely shielded from having to know physical location, format, and custom naming syntax for shared data.
- From a scalability viewpoint, services need ways to keep large datasets operationally ready for access by distributed business processes. SOA infrastructures must scale transparently and handle incremental additions of new data sources as well as additional volume of current sources. Furthermore, access to these scalable data resources must avoid bandwidth bottlenecks like slow disk I/O at all costs, lest computational resources idle uselessly while waiting for working dataset transfers.
- From a reliability viewpoint, the high-availability of data is crucial to SOA processes that depend on datasets lying in the critical path to service fulfillment. Without guaranteed data access reliability, businesses cannot offer competitive service level agreements based on the delivery of consistent quality-of-service.
- From an operational viewpoint, SOA applications require fine-grained access to data. Not only is elemental access and delivery of objects (located within files) essential for minimizing network bandwidth and speeding responsiveness, fine-grained data mining operations enable more precise decision-making and meaningful interpretation of analytic data.

- From an integration viewpoint, SOA applications must be capable of representing data and processing information in a neutral, open, and extensible manner that eliminates reliance on proprietary integration technologies. At the same time, SOA applications must be able to merge disparate formats through data transformations and allow application access through a wide variety of access mechanisms.

A middle-tier data fabric overcomes the latency inherent in most large monolithic systems by operating with data in a service-oriented manner. Service orientation abstracts individual data sources into discrete business-oriented units referred to as data services. Data services can be thought of as network enabled resources that establish a common representation for identifying, discovering, invoking, and provisioning any IT data resource, including relational databases, packaged applications, and unstructured content repositories. Since data services share a common representation, data services pools can easily be aggregated and activated based on instantaneous loads. SOA environments with data services pools can dynamically provision data resources as needed for load-balancing and data locality purposes, thus delivering consistent quality-of-service.

### GEMFIRE™ ENTERPRISE

GemFire Enterprise, an Enterprise Data Fabric (EDF) offering from GemStone Systems, Inc., is a distributed operational data management platform that bolsters SOA architectures by offering high performance data access and distribution in a scalable and reliable fashion. Enterprise data that is locked in multiple disparate systems is made immediately available and accessible through a middle-tier data backbone to which the different services and applications connect. Based on open Java standards, XML and Web Services, GemFire Enterprise manages operational, analytical and historic data through a highly scalable distributed cache. By aggregating data on demand and co-locating it with business process applications, GemFire Enterprise solves problems of network latency and data source bottlenecks while improving resiliency to server failure.

The following GemFire Enterprise features help enhance the quality of SOA environments:

**Data Virtualization:** GemFire Enterprise creates a layer of abstraction between enterprise data sources and the applications and services within the SOA. By enabling data virtualization, GemFire Enterprise facilitates:

- a) Location transparency - access to data without worrying about the location of the data
- b) Format neutrality - sharing of data across Java, XML, C++ or SQL applications through a common infrastructure
- c) Data source heterogeneity - representing data from multiple disconnected backend data sources such as databases, files and mainframes.

Data virtualization can be thought of as a logical extension to the service virtualization that is typical of an SOA. Just as composite applications or business processes invoke Web Service operations on portTypes without any knowledge of the underlying service implementation, data access becomes indifferent to the underlying source. From an application's perspective, all data 'appears' to be available in one huge data repository.

**High Performance Data Access:** Through GemFire Enterprise's efficient in-memory data management, applications are guaranteed data access at extremely high speeds. The responsiveness of these applications increases considerably as disk I/O or CPU-intensive queries against databases no longer hinders performance. This obviously impacts the end-user experience in a positive manner. Furthermore, GemFire Enterprise supports multiple caching topologies that can scale to large datasets by using the RAM available on multiple nodes or by overflowing to disk to accommodate excess data volume. GemFire Enterprise is unique in its ability to guarantee high performance in conjunction with data virtualization, as these two concepts have often been seen as conflicting objectives.

**High Availability:** Ensuring service availability is often a significant challenge in SOA environments. GemFire Enterprise offers sophisticated high availability features that provide fail-over capabilities upon application failure and also ensure data availability even during network outages or when the back end data source cannot be reached. Data that is held in memory can be replicated to standby server nodes that can service client requests when the primary application fails. To the end-user or application this transfer is invisible. GemFire Enterprise can also be configured to persist data to disk for complete data recovery, even when

all applications have gone down. Upon restart an application can regain its state either by obtaining a data copy from a replicated node or from disk. Both replication and disk persistence can be configured to be synchronous or asynchronous. Since GemFire Enterprise holds data in memory, applications are shielded from temporary network outages or data source unavailability. These high availability options are critical in complex business processes where the process would go into an irrecoverable failure mode if the data and state was not available to the underlying services. Thus GemFire Enterprise's high availability is important in ensuring the necessary quality of service for SOA-based applications.

**Low Latency Data Distribution:** Since SOA relies on a shared services model, data movement across distributed services is of paramount importance. GemFire Enterprise supports a nimble distribution model that can move data on demand between services, especially when there is shared information between the services. In these kinds of scenarios, data that is stored in the distributed GemFire Enterprise layer is instantaneously available to any application that needs it. Multiple data distribution policies can be used based upon the scenario. In some cases a pullbased distribution model is used, where the data element is moved only upon a client request. In other cases, a push-based approach makes more sense, where any updates are pushed to all relevant services that may be affected by the change.

### ENHANCING SOA WITH GEMFIRE ENTERPRISE

Enterprise data is often fragmented in multiple systems, making it unusable in SOA scenarios. As can be seen in Figure 2, GemFire Enterprise can provide a middle-tier platform that acts as a data backbone for the services offered.

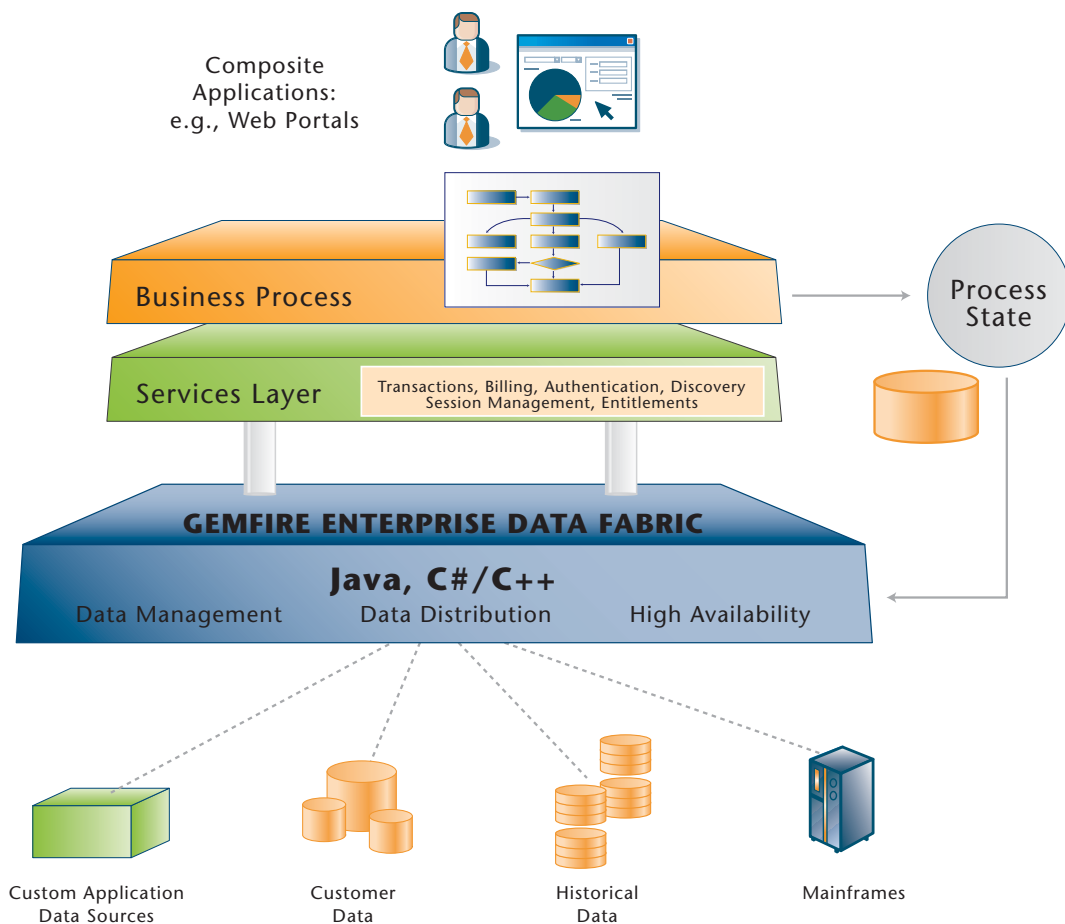


FIGURE 2

GemFire Enterprise thus becomes an enterprise-wide data layer that services can use to access, manipulate and distribute enterprise data efficiently. Within an SOA ecosystem, it can help in dealing with the following data entities:

**Business Application Data:** Most business workflows require access to business data, such as customer data or reference data, which is stored in databases or other back end systems. The operational or working dataset is often a small fraction of the entire source data. GemFire Enterprise provides an intelligent distributed caching mechanism, by which operational data can be held in memory for low latency access and increased system performance and throughput. The subset of data held in memory is swapped intelligently based on client request patterns. In certain cases, operational data volumes may stretch the physical limits of RAM available on a machine. In such cases data can either be partitioned across nodes or moved to disk (with indexes to the disk storage areas) to accommodate surging data loads. Service implementations that use JDBC to access relational databases can use the GemFire JDBC interceptor - an extension module with GemFire Enterprise - to monitor expensive SQL queries and cache result-sets of such queries to avoid expensive round-trips to the database.

**Security Context:** SOA environments perform security checks and look up a user's authorization with each invocation of a service. This means that a service invocation results in a check with a policy server or other security mechanism. These mechanisms inevitably look in a database to find this information. For globally distributed organizations this often means a trip across a WAN as well. All of this work adds latency and significantly slows down fast services. For composite processes with N number of steps this work is performed N times. If the security context is cached in GemFire it can be seamlessly shared across all of the steps in the workflow and the services that compose that process avoiding unnecessary delays that impact QoS and SLAs.

**Business Process State:** Business processes in an SOA are often an orchestration of stateless services, but the real-life business processes are usually stateful. In order to resolve this mismatch, databases and messaging systems have often been used as workarounds to manage state across a business process. However such approaches significantly impede performance and make the process ineffective. GemFire Enterprise, because of its inherently distributed nature, can provide a memory-based caching layer for business process state and context. Services that are distributed over a network have local access to relevant contextual data through the GemFire fabric. Benefits of managing state in this fashion become evident when the same service is executed as a part of several different business processes or by several instances of the same business process.

**Web Session Management:** Since most clients access services through a web-tier, session management becomes a critical function within SOA architecture. Traditionally, session state has often been stored in databases, or more recently in web containers such as J2EE application servers. With databases, session storage and retrieval across distributed services becomes an extreme bottleneck causing timeouts and other process inefficiencies. Caching within web containers or other proprietary caching techniques often cannot scale when the user loads increase. Such mechanisms provide only limited high availability features and are plagued with data inconsistency issues in multi-node clusters. GemFire Enterprise offers a clean and efficient model for managing session data in-memory and in a distributed fashion to enable sharing across multiple services. High availability is guaranteed through replication as well as persistence to disk for complete data recovery. GemFire Enterprise can also scale to large numbers of user-sessions, as it can intelligently move the passive sessions to disk and retain only the active ones in memory. These features ensure that the session data is immediately accessible and available at all times.

**Active Data Change Management:** SOA loosely connects composite applications that perform different business functions. Most often, data changed by one application requires actions by the other applications. For example, an application processing customer orders may send message to the Shipping application to start the delivery process. Traditionally, messaging services are used for such communication between loosely connected applications. Use of messaging system for such communication adds an overhead of marshalling and unmarshalling of data. GemFire Enterprise offers data caching, messaging and event notification services under one product. By using GemFire Enterprise as the data backbone layer for SOA applications, the overhead of data marshalling and unmarshalling is reduced. With GemFire Enterprise, disparate applications can register interest in their relevant data and when the underlying data changes, they get event notifications. For example, the Shipping application can register interest in the order data set and events are sent to the Shipping application if this data set is modified by the Order application.

Additionally, since the data is maintained in an object form in memory, unnecessary transformation of data is eliminated resulting in better resource utilization.

## SUMMARY

Traditional data sources are unable to meet the stringent SLAs and Quality of Service (QoS) demanded by SOA based applications. Moving from 'centralized' data stores to distributed in-memory data management is the solution that promises a scalable and flexible data access model. GemFire Enterprise, the leading data fabric offering, complements SOA with an in-memory data management that increases performance and throughput and provides a transparent way of sharing data across services. With its ability to scale on-demand, distribute data without a single point of failure and access data at memory speeds, GemFire Enterprise is an ideal candidate for the data backbone needs of SOA applications.

THE GLOBAL  
#GRIDAWARDS  
04-28-08

Best Vendor Grid Solution



### Corporate Headquarters:

1260 NW Waterhouse Ave., Suite 200 Beaverton, OR 97006  
Phone: 503.533.3000 Fax: 503.629.8556  
info@gemstone.com

### Regional Sales Offices:

New York | 5 Penn Plaza, 23rd Floor New York, NY 10001 | Phone: 646.530.8458  
Washington D.C. | Phone: 301.564.0550  
California | 444 Castro Street Suite 520 Mountain View, CA 94040