



GemFire Enterprise Data Fabric and EII: A New Approach to Distributed Data Management

OVERVIEW

The widespread adoption of distributed applications and architectures on the one hand has helped enterprises achieve significant benefits, while on the other has necessitated a fresh look at data infrastructure needed to support these applications. Enterprise data in most organizations resides in several different systems across various platforms. Information consumers too are spread across the network and need information instantaneously in a format that their application can comprehend. The ability to ensure high-performance and reliable access to data irrespective of its locations in a reliable (24/7) and scalable fashion (thousands of users) to information stakeholders is a significant competitive advantage that IT organizations in several industries desire.

SOLUTION OVERVIEW

Enterprise Information Integration (EII) solutions offer application developers a uniform facade for information access across multiple data sources. Further, certain vendors offer meta-modeling solutions that help create models of information sources and present a virtual database to IT developers. The meta-data mapping avoids the need for applications to connect to any physical sources directly and hence allows the physical data sources to change without affecting any end-user applications. Also, EII tools support the notion of federated queries, which are SQL-like queries executed over multiple data sources. Such queries can be executed via a JDBC interface and materialized as result sets in the client applications.

GemFire Enterprise, a component of the GemFire Enterprise Data Fabric, is a middle-tier high-performance data/object management solution that works synergistically with EII tools to cache and distribute result sets from federated queries to different client applications across an application network. Introduction of the GemFire Enterprise data layer on top of an EII server enables applications to manage relevant data locally within their process eliminating data latency. By doing so, data can also be distributed on demand to other applications that require similar information. And avoids the need for multiple executions of expensive federated queries. GemFire Enterprise caches can also complement the caching functions that an EII server may provide and deliver a hierarchical topology that includes a centralized EII server cache and a system of distributed GemFire Enterprise caches that are close to the end-user applications.

GemFire Enterprise caches can also provide a caching layer behind an EII server, in order to cache data from individual data sources that are queried by a server. Many times a similar subset of data is used across many federated queries. If this data comes from a source that is slow, overburdened, computationally expensive to access, or physically remote from the EII server, caching data close to the EII server can result in significant performance gains.

Such a solution that combines an EII server and GemFire Enterprise's data management capabilities will be of great relevance in several use-cases where real-time data integration and data distribution are important. Examples of such scenarios include comprehensive real-time intelligence for battlefield operations managed by federal intelligence agencies or reference data management in financial services, wherein there is a need for reliable access to large volumes of data and distribution to client applications without latency. In all these scenarios, a solution like this provides comprehensive access to critical information for all applications and users within a distributed enterprise.

Figure 1 illustrates the combined solution architecture involving GemFire Enterprise and an EII server. As shown in the figure, the EII server provides data access, federated querying and aggregation services. These services are complemented by the high-performance data caching and distribution services of the GemFire Enterprise data layer. Further, the connectivity between GemFire Enterprise and the EII server is handled through an adapter/interceptor that GemFire Enterprise provides for JDBC compliant data sources. This interceptor provides intelligent ResultSet caching capabilities and is discussed in further detail in the following section. Client applications can access the cached result sets through a variety of interfaces (Java, C/C++, SOAP, and XML) and information can be shared across different application nodes. For instance, if application A has cached the results of query A, and application executes query A,

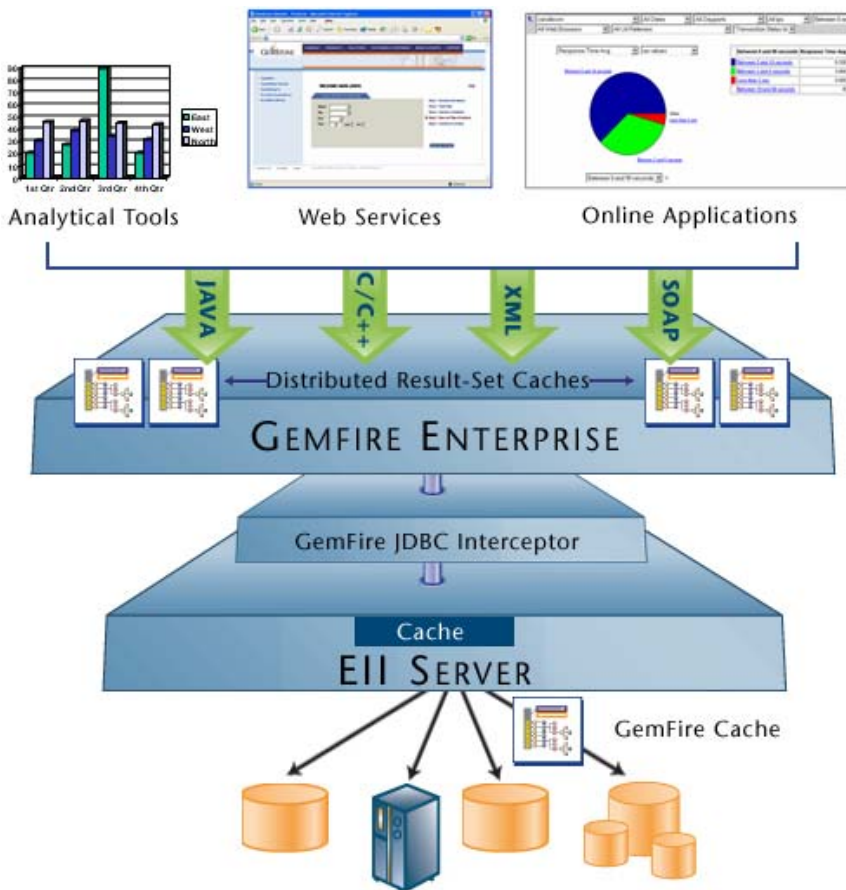


Figure 1: GemFire and EII Solution Architecture

then data is pushed from application A to application B and the request is instantaneously satisfied without the need to invoke the JDBC layer. In addition to the caching and distribution services, the GemFire Enterprise layer also offers high-performance transformation services for XML and Java applications to share data via Object-to-XML bindings and vice-versa. Hence, client applications can be integrated with one another and with backend information sources in an effective manner.

In addition to the ResultSet caching, GemFire can also cache SOAP responses that the EII server may return based on SOAP requests from a Web Service client for instance. This prevents the expensive marshalling and unmarshalling of XML data and consequently improves performance.

GEMFIRE JDBC INTERCEPTOR

GemFire JDBC Interceptor is a pluggable adapter that can be used with an EII server (as shown in Figure 1) or directly with a database for caching result sets. A specialized JDBC compliant driver removes the need for any additional code to enable caching or detect query performance. The driver, through the use of a simple and intuitive graphical user interface allows the user to monitor any application for expensive SQL/federated queries. Users can then configure individual queries or patterns of queries as being cacheable and dynamically cache and distribute query result-sets in memory and/or disk across multiple nodes. The Interceptor also commits data to the database when an update occurs to ensure data integrity. When a query result set is not found in the cache, the driver fetches it from the EII server and stores it in the cache before returning it to the user. Retrieving results from the in-memory cache can greatly enhance the performance of Java/J2EE applications, by avoiding database roundtrip times as well as improving performance of the database.

By allowing data within a database system to be cached in local memory, GemFire JDBC Interceptor eliminates the overhead of executing a query, and returning results across a network connection, thus enhancing application performance. Caching in this manner ensures one or more significant improvements in data transfer performance and/or a reduction in EII server/database loading as the case may be. A majority of Java/J2EE applications are deployed in a distributed or clustered environment where applications running on multiple nodes access a single EII server. In such situations, it becomes essential to understand the data source access patterns across the entire system, not just on a single server to effectively tune performance. GemFire JDBC Interceptor permits the application developer to tune the complete distributed application by collectively showing the SQL statements executed across all the nodes making the job of tuning the distributed system very easy.

For expensive SQL queries that are frequently executed on clustered nodes, applications have the choice to replicate data in the cache to all the other nodes in the system (push), or retrieve data from other nodes on demand (pull), dramatically decreasing the traffic to the underlying data source. GemFire JDBC Interceptor provides multiple options to maintain freshness of the cache data. Applications can configure "Time To Live" (TTL) for individual queries or sets of queries to refresh the data from the data source. The TTL can also be configured at a table/column level for which result sets are cached. Applications can alternatively configure an absolute time when the query result set gets invalidated in the cache. This interceptor also has enough intelligence built-in to interpret data manipulation statements (inserts, updates and deletes) and invalidate result sets. Any change made to a local cache is synchronized with other members in the system, providing maximum data consistency throughout the system.

In future, GemFire will allow for query rewrites with the JDBC interceptor. This technique will be useful in:

- 1) increasing cache hits and further improve system performance
- 2) dealing with data where users may see different rows depending on their security level
- 3) dealing with data where most of the data can be cached but a column or two needs to be retrieved from the EII server

DATA MANAGEMENT POLICIES

In several instances the queries that are executed by an EII server can result in large volumes of data being returned as result sets. One of the challenges in such a scenario is management of the data that is cached - where to hold the data that is cached and how much to cache.

GemFire Enterprise provides several alternatives. Result sets can be cached in the application process that is making the call to the EII server. Alternatively, data can be distributed on multiple machines by creating a server cache that is outside the application process space and holds result sets that are used by multiple applications. In this case there is a local cache at the application level that holds the most recently accessed result sets in the application process space and the rest are held in the server cache.

Multiple applications can also share results sets if needed. GemFire Enterprise can also be configured to limit the amount of result sets that are in-process memory and have the other results that are in the cache to overflow to disk. In this case, GemFire Enterprise executes a lazy write strategy to move data to disk without using a significant amount of CPU resources. References to data on disk are kept in smart hash based indexes so that fetching data on disk is also extremely fast. GemFire Enterprise is neither limited to the 4 GB memory limit of a 32-bit machine nor to the amount of actual physical memory of a 64-bit machine. It can store a con-

figurable but arbitrary amount of cached results. In areas where query results are large or contain large objects, this can be significant. GemFire Enterprise can also be configured to persist data to disk, synchronously or as required, in order to recover in the event of catastrophic system failure. When a failed application restarts, it can reload its cache instantaneously from the data persisted on disk.

Result sets that are held in GemFire Enterprise within an application can also be mirrored to an instance another machine for high-availability. Updates to the primary instance are instantly propagated to the mirror and this way even if the primary application fails unexpectedly no data is lost and the primary instance can reload the data from the mirror. Due to the distributed nature of the cache and the ability to replicate data objects on more than one computer, GemFire Enterprise can enhance the survivability of applications by enabling them to have access some data even if the network disappears. The application enables applications to continue to work with data that was retrieved before the network outage.

Since GemFire Enterprise provides distributed caching outside of the EII server processes, it can be deployed across a wide area network to reduce overall network bandwidth usage and to provide faster results to client applications. With GemFire Enterprise, once data has been transmitted across the WAN for one user, it is available for all users. The reduction in bandwidth usage can be significant in low bandwidth environments both from an application performance viewpoint and from an overall view to maximizing the amount of data/bandwidth available.

HOW GEMFIRE ENTERPRISE COMPLEMENTS EII SERVERS: A SUMMARY

The following points indicate how GemFire Enterprise complements and augments an EII solution-

- Provides a scalable and high-performance data management solution that is not limited by the 4GB in-memory restrictions of a 32-bit architecture
- Facilitates data caching at application level as well as at a server level to accommodate different architectures
- Enables caching of data across a network with absolutely no development effort and is transparent to the application
- Enables sharing of query result sets across multiple applications that rely on data from an EII server (Future support for query splitting that will increase the cached data coherency and further improve performance)
- Offers APIs from C/C++, Java and XML applications to access and collaborate on enterprise data

- Guarantees high availability of data through disk persistence as well as mirroring/replication to backup nodes
- Offers a viable solution even in low-band environments/WAN with remote clients and can provide access to recently accessed data even during network outages

IT organizations can greatly benefit from the synergy that GemFire and EII solutions bring. EII tools provide a convenient mechanism to model, access and query enterprise data sources. GemFire helps IT architects efficiently manage the vast amounts of data that is accessed via EII and deliver it to applications on-demand with a high degree of performance and scalability. This combined solution can be prudently used to deploy nimble, resilient, real-time data infrastructures.

Corporate Headquarters:

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

Regional Sales Offices:

New York | 90 Park Avenue 17th Floor New York, NY 10016 | Phone: 212.786.7328

Washington D.C. | 3 Bethesda Metro Center Suite 778 Bethesda, MD 20814 | Phone: 301.664.8494

Santa Clara | 2880 Lakeside Drive Suite 331 Santa Clara, CA 95054 | Phone: 408.496.0242



Copyright© 2005 by GemStone Systems, Inc. All rights reserved. GemStone®, GemFire™, and the GemStone logo are trademarks or registered trademarks of GemStone Systems, Inc. Information in this document is subject to change without notice.