



GemFire Enterprise Data Fabric Turbo-Charges a Program Trading Environment

BENEFITS FROM GEMFIRE:

- Up to 6000 trades/sec (cache writes with synchronous persistence) and 14000 trades/sec (cache writes with asynchronous persistence) and over 200,000 cache reads/second. This represents approximately a 600% increase in performance
- Ability to handle sporadic loads due to large customer orders and increase revenues
- Increased reliability due to mirrored order/execution data caches and persistence
- Representation and distribution of complex order and trading data in object formats without the need for relational or other formats

A leading Investment banking, securities trading and brokerage firm is deploying GemFire Trading Reference Architecture as an integral part of their program-trading infrastructure (basket-trading).

BUSINESS PROBLEMS

Their business workflow involves receiving bulk orders from institutional clients as well as signals from market data feeds. These orders are broken down into "child" orders/baskets and then routed to the respective exchanges for execution. The trade executions are tracked, orders filled and notifications regarding failed trades are relayed back to the client. The "parent" orders, "child orders" and execution information are persisted in a relational database and retrieved as necessary during the order management process.

With the existing architecture, the following business problems occur:

- High Latency issues with an RDBMS based architecture
- Inability to handle the necessary trading volumes to support large orders at the end of trading day
- Lack of Reliable fail-over mechanisms

Consequences:

- Loss of revenue from turning down orders/consumption from their own positions
- Lack of reliable mechanisms to ensure business continuity

THE GEMFIRE SOLUTION

GemFire Enterprise, which is the underlying component of the reference architecture for trading, makes data available on-demand to applications regardless of the underlying data sources or formats. GemFire Enterprise is built on the industry's fastest and most reliable high-performance data distribution and caching system.

Figure 1 illustrates the GemFire based architecture for this program trading environment. The key technical features and services provided by this solution include:

High Performance Caching: Order and trade information is held in the distributed GemFire caches that server multiple order execution engines and instantly accessed/updated during the order routing and execution process. Data regions (caches) can be defined based on business logic to

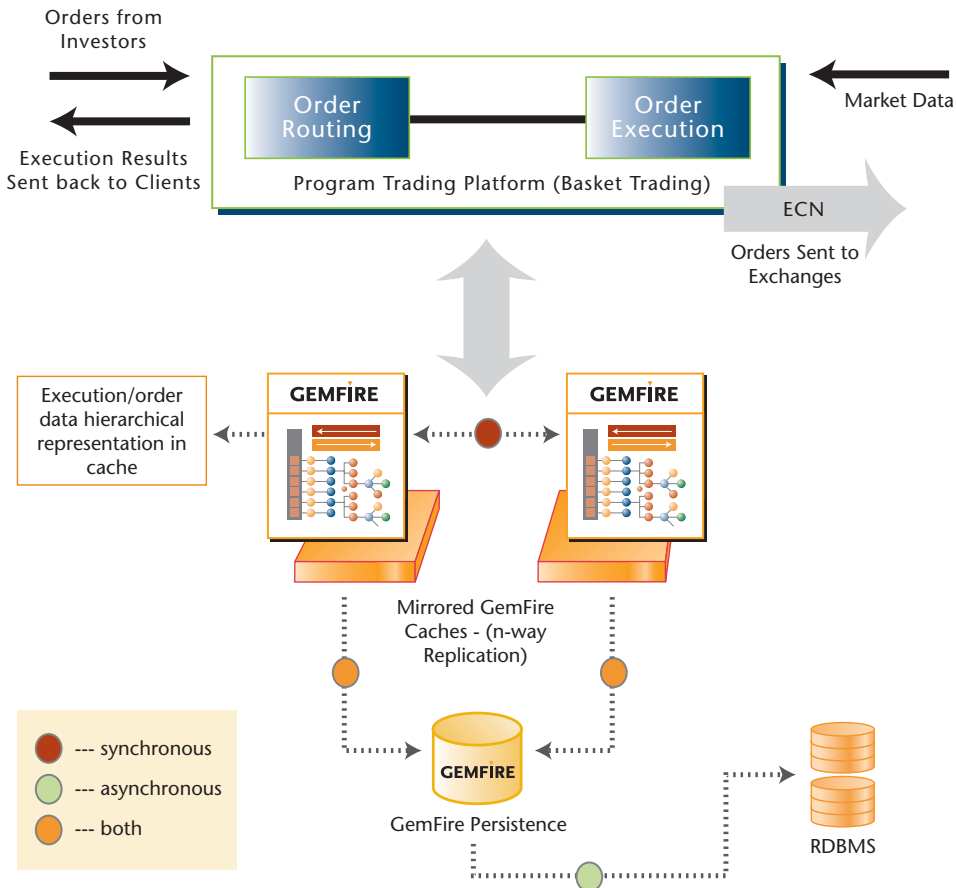


Figure 1: GemFire Based Program Trading Architecture

hold related data in a single cache instance. Extremely high-speed reads and writes, and consequently high order-processing volumes, are ensured through this mechanism and data latency issues are successfully resolved. New orders that enter the system and order execution data are both recorded on a transaction log with checkpoints that enables recovery from failure as discussed later.

Replication: GemFire data caches are replicated across multiple nodes, with synchronous data propagation to mirror nodes, to ensure 100% data backups. This eliminates single points of failures and balances client loads. When a failed application restarts, it can automatically reload its cache from a mirror node and seamlessly process client requests. This feature is extremely critical for this program trading environment, where even a few moments of downtime represents a

significant monetary loss. In certain scenarios, a failed application may also recapture its state by recovering data from the transaction log based on checkpoints that avoid reloading of the entire transaction log.

Persistence: In order to ensure complete resilience to system failure and data recovery especially in cases where a significant number of the replicated servers are down, GemFire can be configured to persist the cached order and execution information to disk. As illustrated in Figure 1, this persistence can be configured to be synchronous (as and when the cache is updated) or asynchronous (batch writes to disk). Data stored on disk can automatically be reloaded to a cache on application restart. Persisted data can also be asynchronously archived in a relational database. When an overflow condition is encountered in the in-memory cache, the system can

be configured to overflow to disk and dynamically scale to increasing trade volumes.

Data Distribution: GemFire enables agile distribution of data across the different components of the order management/trading platform. The distributed caching features of this solution enables a portion of data to be held in memory within an application and make that immediately available to another application that needs it, either on-demand or in a push-mode. This enables applications and system components to share data effectively.

Object representation: Parent orders (aggregate orders), child orders (basket orders sent to exchanges) and execution and order fill information are represented in object formats, including their relationships, in the GemFire Enterprise Data Fabric. This enables applications to work with GemFire directly without the need for any complex transformations. Avoiding these transformations further improves the performance of the trading platform.

**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.