GigaSpaces Peer-to-Peer Cluster Patterns

White Paper
June 2004

USA Office
GigaSpaces™ Technologies Ltd.
257 Park Avenue South 8fl.
New York, NY 10010
Tel: 646-862-6935
Fax: 212-308-7886

EMEA Office
GigaSpaces™ Technologies Ltd.
8 Maskit P.O. Box 12897
Herzelia, 46733, Israel
Tel: +972-9-952-6751
Fax: +972-9-957-6780

E-mail
info@gigaspaces.com

Technical Support
support@gigaspaces.com

Sales
sales@gigaspaces.com

Restricted Rights Legend
This document may not, in whole or in part, be photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from GigaSpaces™ Technologies Ltd. Information in this document is subject to change without notice and does not represent a commitment on the part of GigaSpaces™ Technologies Ltd.

Trademarks
GigaSpaces™ and GigaSpaces™ Platform are registered trademarks of GigaSpaces™ Technologies Ltd. All other trademarks are the properties of their respective owners.

Table of Contents

1. INTRODUCTION 3
2. THE GIGASPACES P2P CLUSTERED SPACE 3
   2.1. *P2P Space Benefits* 4
3. ACHIEVING CONSISTENCY IN A P2P CLUSTER 5
   3.1. *Synchronize on Commit* 5
   3.2. *Single Master-Space Configuration* 5
   3.3. *Master Per Entry Type or Hash Code ("RAID" Space)* 5
   3.4. *Single System Image* 5
4. ACHIEVING PERSISTENCY IN A P2P CLUSTER 6
   4.1. *Direct Persistency Using a JDBC Storage Adapter* 6
   4.2. *Buffering Update Operations* 6
   4.3. *Using a Backup Persistent Space Cluster Member* 7
5. UNLIMITED PROCESSING NETWORK (MASTER-WORKER) 8
   5.1. *Master-Worker Benefits* 8
6. GLOBAL DIRECTORY SERVICE 9
   6.1. *Clustering Architecture* 9
7. COMMUNICATION PLATFORM 10
8. CODE EXAMPLES 11
9. SUMMARY 12
10. ABOUT GIGASPACES 12

Table of Figures

Figure 1 - Agent communication through communication spaces 3
Figure 2 - Backup Persistent Space Architecture 7
Figure 3 – Processing Network 8
Figure 4 - Global Security Directory Service 9
Figure 5 - Cluster of Site-Clusters 10
Figure 6 - The Virtual Real-Time Enterprise 11

1. Introduction

Being a pioneer in the field of Peer-to-Peer (P2P) clustered space, GigaSpaces has gained valuable experience in and knowledge of many issues that current technologies have either failed to address, or were too complex to implement.

This document explains how the GigaSpaces architecture provides an extremely scalable and efficient solution for handling millions of entries and thousands of concurrent users with thousands of operations per second - to meet the needs of the real-time enterprise.

2. The GigaSpaces P2P Clustered Space

Most current messaging applications use a centralized server approach, in which a centralized exchange manages communication between the information producer and the information consumer. The main drawback of this approach is that the messaging exchange becomes a major performance bottleneck.

The GigaSpaces P2P clustered space solves this problem by establishing a virtual communication bus, using a set of embedded, replicated spaces. With the GigaSpaces approach, it is possible to achieve high throughput between peer nodes in a clustered environment.

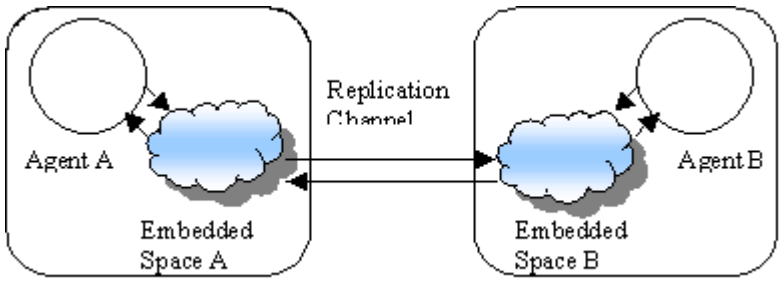

Figure 1 - Agent communication through communication spaces

In the above scenario, agents A and B communicate through a pair of embedded spaces, which are connected to each other by means of a replication policy.

Both agents perform the space operations on their local space without direct network calls. The information is passed to the other agent through replication, which occurs in the background. The replication policy specifies the rate at which this information will be passed.

The communication rate is dependent on:

- **Time interval** - the intervals at which replication is triggered.
- **Size -** the maximum amount of entries that are buffered before a replication is triggered.
- **Chunk Size -** the number of entries that are passed in each replication call.

CASE STUDY

Using measurements taken from a highly demanding real-time application in the telecom sector, we compared the performance aspects of a centralized server approach and other messaging technology with GigaSpaces P2P approach.

According these measurements, the P2P approach improved performance by up to 500%. Additionally, GigaSpaces was able to process thousands of operations per second between two remote agents, using standard, single-CPU, Intel-based machines.

2.1. P2P Space Benefits

Performance

The main benefit of a P2P space configuration lies in its performance. Replication performs transparent batching of network calls, reducing the number of network calls between agents.

Concurrent Agents

A P2P configuration provides linear scalability. That is, if one group of agents can handle 100 concurrent agents, two groups of agents can handle 200 concurrent agents – while still maintaining the same rate of response - and so on.

In a P2P configuration, the application can bundle a group of agents within a single VM, using an embedded space. This group of agents communicates through the space, with no network calls. Further, via replication, the group communicates with another group of agents on a remote VM. As opposed to a centralized configuration, in a P2P scenario each group of agents manages its own communication internally, with no network calls - lowering processing overhead and increasing capacity.

CPU Scalability

GigaSpaces highly parallel architecture allows for linear scalability and performance increase relative to the system's CPU power. The non-centralized P2P approach makes it possible to use the CPU power in each of the groups, and thus increase processing power relative to the number of space peers in the network.

Offline / Online Capabilities

Since the space is embedded with each agent application, the agent can submit its tasks to the space even while in offline mode. These tasks are sent automatically to the other spaces in the network when the agent reconnects.

Simple Deployment and Maintenance

The embedded space is a set of libraries, and becomes an integral part of the agent application. The agent provider is relieved from the responsibility of maintaining additional processing and installation. Moreover, the footprint in such a configuration is minimal.

Copyright © 2004 GigaSpaces Technologies

Transparent Access to Embedded or Remote Space

From the application perspective, working with an embedded or remote space is completely transparent. This enables a flexible configuration of "thin" and "fat" agents. The configuration can be changed on deployment based on the specific environment requirements.

3. Achieving Consistency in a P2P Cluster

Consistency and performance in a clustered environment are often conflicting requirements. Setting the synchronization policy based on a specific scenario is the key for achieving the best performance while maintaining consistency. GigaSpaces provides several options for achieving synchronization and consistency in a P2P cluster.

3.1. Synchronize on Commit

With this option, every update on the cluster is transactional. Upon commit, the content of the transaction is transmitted to all cluster members, and the client application is blocked until all cluster members commit the transaction.

This feature can be used as a "checkpoint," whereby applications update their local spaces and synchronize the state of their work with other space nodes by committing the transaction. This guarantees that information stored under the transaction will be recoverable upon failure. This method is extremely efficient in cases where the transaction boundaries are relatively high (x10, x100 of operations per transaction), since such an operation would otherwise be very expensive in terms of performance.

3.2. Single Master-Space Configuration

A master-space configuration means that a specific space member in the cluster maintains all the updates of the shared information. This option ensures that all the take/write operations (destructive operations) are routed to the master space, and all the read operations are spread around the cluster members according to the load-balancing policy. When the agent contains an embedded-space, the default behavior is to route all the read operations to the local space.

3.3. Master Per Entry Type or Hash Code ("RAID" Space)

In some cases, each member in a cluster may have different entry ownerships. The ownership may be determined arbitrarily or by performance requirements. In such cases, every node is expected to get fast "update access" to the entries owned by the node, and slower "update access" to entries not owned by the node. This policy can be achieved with a hash-based load balancing policy. This takes the entries index field and routes the update operations (take, write) to the space node that matches the entry hash % spaceNum algorithm.

3.4. Single System Image

With Single System Image (SSI), every space member is a master of all the entries. There are scenarios in which entry ownership cannot be split into individual space nodes. In these cases, the cluster must maintain the consistency level of the SSI.

A normal update operation on a space involves the combination of a take operation and a write operation. In a cluster that supports SSI, a take operation needs to succeed only once, regardless of the take operation target space. A read operation attempted on an entry currently being updated should fail or block until

the update is completed. To achieve this level of consistency, all take operations are done under a transaction on all the cluster members. The take is blocked until it has completed on all the nodes (the actual algorithm is more complex, but the concept is the same). Concurrent take operations on the same entry are blocked till the entry is written back. The write/read operations are executed locally on the cluster member and propagated through replication to all the cluster members. To maintain consistency between concurrent take, the order in which the take is performed must be consistent across the space nodes. That is, the take must begin at the same space and end at the same space.

4. Achieving Persistency in a P2P Cluster

Persistency is required to maintain the durability and reliability of the space information. Persistency often involves saving information to disk, but this operation is considered very expensive in terms of performance. In this section, we will examine how persistency can be achieved in a P2P cluster.

4.1. Direct Persistency Using a JDBC Storage Adapter

The storage adapter interface provided with the GigaSpaces platform provides a generic interface for integrating with various persistency backends. With GigaSpaces, a persistent space is created by simply attaching a space with a specific storage adapter. A generic storage adapter that integrates through JDBC is provided as part of the platform.

The JDBC storage adapter leverages the proven scalability and stability of RDBMS implementations. GigaSpaces takes advantage of indexing capabilities to provide flat responsiveness while dealing with millions of entries. It also uses this approach to scale beyond the physical memory capacity through LRU-based caching.

The storage adapter provides the flexibility needed to choose the most cost-effective of the various databases available - a light RDBMS, such as PointBase or CloudScape; a free RDBMS, such as MySQL; or a more scalable RDBMS such as Oracle, DB2, etc.

Performance is the main problem when using RDBMS to store object data. Direct access to persistent storage creates two performance-related problems:

- **Write/Read Access is Relatively Slow Compared to In-Memory Access -** there is a strong dependency between the space performance and the underlying RDBMS performance. In the following section, we will examine various ways to deal with these problems and improve the performance of a persistent space.

- **LRU-Based Caching Reduces the Read Overhead -** GigaSpaces uses an LRU-based caching mechanism that stores the most recent information in memory, avoiding the necessity of accessing the underlying RDBMS in read operations.

4.2. Buffering Update Operations

With this approach, all update operations are sent first to the space cache and then to the underlying RDBMS through a log file. Here, the performance overhead will be limited the overhead of writing to the space cache and the log file – which is much smaller than committing an INSERT operation to a RDBMS.

With this option, the overall performance of a persistent space is not dependent on the performance of the underlying RDBMS. The update to the RDBMS is done as a background process.

4.3. Using a Backup Persistent Space Cluster Member

This option uses GigaSpaces' clustering capabilities, providing in-memory access to the space while maintaining the persistency in the background using a backup persistent space.

The diagram below illustrates how this is achieved. Spaces A and B are memory-based embedded spaces. Space C is a persistent space. Agents A and B communicate through the in-memory spaces. The information stored in spaces A and B is replicated in the background to persistent space C. When space A or B is re-started, their data is automatically recovered from space C.

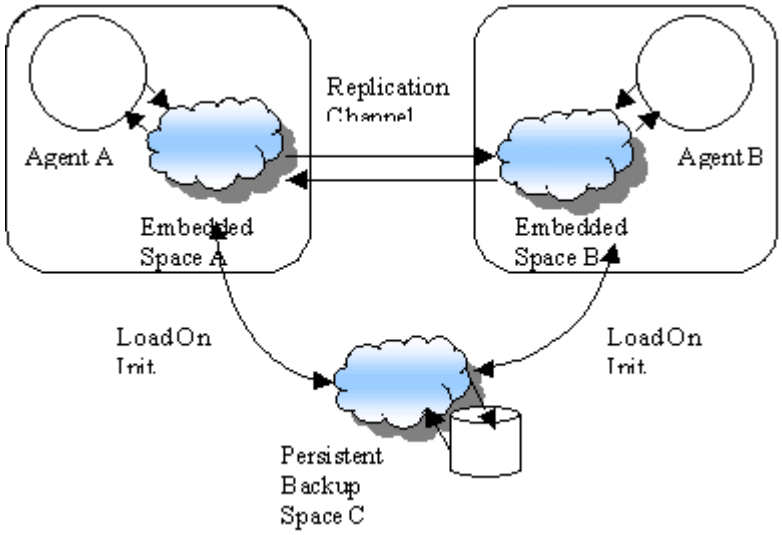

Figure 2 - Backup Persistent Space Architecture

This persistency mechanism allows in-memory application access to the space, while maintaining recovery capabilities using the backup persistent space. The backup persistent space can be used to monitor space content. Since the space is not involved with the actual interaction between agents, it can also be used as a basic query/reporting engine via which other management or reporting services can get real-time information on the state of the space.

5. Unlimited Processing Network (Master-Worker)

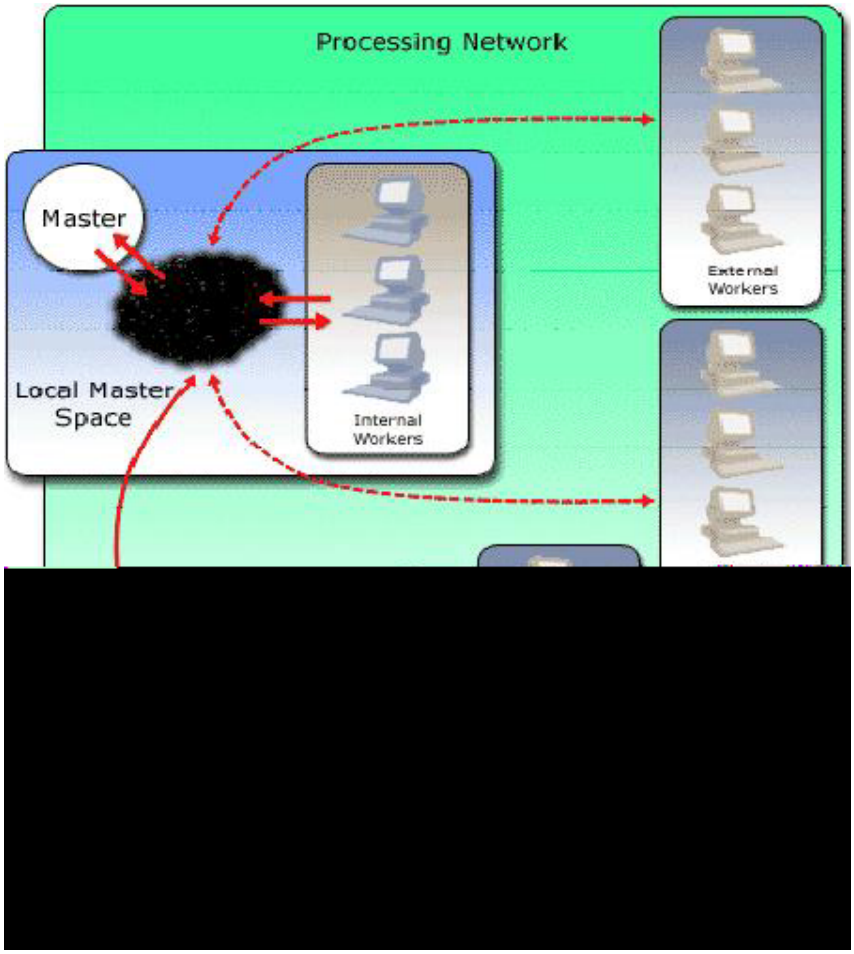

Figure 3 – Processing Network

Master-worker is a known pattern for parallel processing. The master breaks a processing task into a set of sub-tasks. The sub-tasks are performed in parallel, and the results are gathered and brought back to the application. With this approach, a task that used to take N times to process (N = number of sub tasks) now takes only the time it takes to process a *single* sub-task.

A growing number of applications are using the master-worker pattern for computational tasks, simulations, distributed searches, EAI, etc. The GigaSpaces combination of embedded space and clustering provides an extremely scalable and efficient platform for processing master-worker tasks.

5.1. Master-Worker Benefits

Dynamic Unlimited Scalability

With master-worker, an application can scale transparently from a single-process, multithreaded environment to a combination of multi-threaded and multi-processor environment. Load balancing and additional scalability is achieved through a cluster of processing groups. Each processing group consists of a space and a set of workers. The groups are connected to each other through replication, and thus appear to the application to be a single processing unit. With this level

For a full description of the examples, refer to the Example Index file located in the examples directory of GigaSpaces installation.

9. Summary

The P2P clustered space architecture is a totally new approach designed to meet the challenges of the next generation of real-time enterprise. It allows GigaSpaces to achieve the levels of scalability and performance that were unattainable using any of the current available alternatives.

10. About GigaSpaces

GigaSpaces offers the first grid server for real-time distributed transaction processing. Specifically designed to enable the high-speed execution of business applications with high and volatile transaction volumes, GigaSpaces enables grid architecture over low-cost open platforms - such as blade and Linux servers.

Headquartered in New York with an R&D center in Israel, GigaSpaces' technology is used by major financial, telecom and manufacturing organizations worldwide such as AIG, Nortel Networks and Hutchison 3G. The company was founded in 2000, and is a subsidiary of Formula Vision (TASE: FVT), and part of the Formula Group (NASDAQ: FORTY), Israel's largest publicly held IT company.

North America
GigaSpaces Technologies Ltd.
257 Park Avenue South 8fl.
New York, NY 10010
Tel: 646-932-3482
Fax: 212-308-7886

Info: info@gigaspaces.com
Support: support@gigaspaces.com
Sales: sales@gigaspaces.com

EMEA
GigaSpaces Technologies Ltd.
8 Maskit P.O. Box 12897
Herzelia, 46733, Israel
Tel: +972-9-952-6751
Fax: +972-9-957-6780

Copyright © 2004 GigaSpaces Technologies