

Building Distributed Applications

# High-Performance Excel-Based Applications in Financial Services

Nati Shalom, Dekel Tankel, and Alon Lahav  
GigaSpaces Technologies, Inc.

Stevan D. Vidich and Shahar Prish  
Microsoft Corporation

August 2007

Applies to:  
Financial Services Architecture  
.NET Framework

**Summary:** This paper describes a comprehensive solution based on Microsoft and GigaSpaces technologies that addresses scalability and performance challenges with Excel-based applications in securities and capital markets industry. The solution combines the latest Microsoft technologies, including Office Excel, Excel Services in Office SharePoint Server 2007, User Defined Functions, and Windows Compute Cluster Server 2003 with GigaSpaces eXtreme Application Platform to deliver unparalleled usability, performance, and scalability. (16 printed pages)

## Contents

- Introduction
  - Overview
  - Challenges with Existing Applications
  - Addressing the Challenges
- Microsoft Solutions for Mainstream Financial Services Applications
  - User-Defined Functions (UDFs)
  - Excel Services
  - Windows Compute Cluster Server (CCS)
- The GigaSpaces Solution
  - An SLA-Driven Data Grid
  - Space-Based Architecture in a Nutshell
- Combined Solution
  - Solution Overview
  - Integration with Excel 2003 and 2007 Clients
  - Solution Benefits
- Customer Use Case Example: Trading Analysis
- Conclusions
- References

## Introduction

This section presents an overview of the scalability and performance challenges of existing Excel-based applications, and how the new solution addresses these challenges.

## Overview

Microsoft Excel-based applications are widespread in capital markets. Traders and “middle office” users leverage Excel because of its unparalleled usability, flexibility, and rich graphical capabilities.

There is a large and fast-growing segment of financial services applications that need to process high volumes of very low-latency data. In addition to supporting demanding analytical processing, these applications must be able to scale to keep pace with the rapid growth of market data and transaction volumes.

## Challenges with Existing Applications

During the past two decades, Excel has become the most popular spreadsheet application for front and middle-office solutions in capital markets. Excel is a highly flexible and user-friendly tool, and large investments have been made in utilizing for complex analytics.

With the exponential growth of data volumes and adoption of highly distributed environments, Excel-based applications face new challenges in capital markets industry:

- **Need to handle large volumes of data in memory**  
Excel is limited to processing data that fits into the available memory of a single desktop machine. In today’s markets, however, Excel is often required to access data volumes that are larger than what the desktop computer can handle in memory.
- **Scalability and performance**  
With existing architectures, data resides in a central (typically remote) location, such as a database. This data source can become a scalability bottleneck – the number of Excel clients that can access it simultaneously is limited. Moreover, network traffic incurred by round trips to the database, as well as disk I/O access can have significant impact on solution latency.
- **Continuous high availability**  
Because data and logic are processed locally within each Excel workbook, it becomes very hard to provide continuous high-availability without forcing application downtime. This issue is especially relevant in the cases of hardware failures, versioning, upgrades, and configuration changes.
- **Free-up desktop resources**  
With existing architectures, Excel running on a desktop can leverage up to two CPUs, as supported by the desktop operating system (Windows Vista or XP). Although there is no limit to the number of cores each CPU can contain, the desktop computer can become CPU bound with

certain workloads. This problem results in longer computation times, and limits user's ability to run additional applications on the desktop, including the presentation functions of Excel itself.

- **Dynamic scalability**

Application loads can vary greatly throughout the day. Excel-based applications need the ability to scale up and down on the fly to meet varying loads without taking up unnecessary resources.

These topics are discussed in detail in a recent GigaSpaces white [paper](#). In summary, existing Excel-based applications are limited to a single compute and data node. There is a need for an end-to-end solution that allows these applications to run across a pool of machines – a grid.

## Addressing the Challenges

Microsoft and GigaSpaces have teamed up to provide an integrated solution that addresses the challenges described previously. Microsoft's technologies provide the ability to decouple computational logic from the presentation layer, while GigaSpaces technology completes the solution by coupling the logic with the relevant data, and executing them in the same process for minimum latency.

The solution combines GigaSpaces eXtreme Application Platform (XAP) with Microsoft's latest technologies for high performance computing and spreadsheet management:

- GigaSpaces **In Memory Data Grid (IMDG)** together with Microsoft's server-based spreadsheet calculation engine – **Excel Services and User-Defined Functions (UDF)** – provide applications with access to distributed, fast, reliable, and scalable data processing.
- GigaSpaces **IMDG** collocates the application logic with the data, and provides content-aware scheduling. Microsoft's Windows **Compute Cluster Server (CCS)** is used to initiate jobs and distribute Excel workload across a grid of compute nodes. Computational logic is pre-deployed to grid nodes and invoked from UDFs via CCS. When processing requires fast data access, XAP takes care of fetching large volumes of data with very low latency.

## Microsoft Solutions for Mainstream Financial Services Applications

This section provides a high-level overview of three of Microsoft's latest technologies that participate in the integrated solution. User Defined Functions (UDF) and Excel Services take part in server-based spreadsheet calculations, while Windows Compute Cluster Server (CCS) distributes workload to compute nodes.

### User-Defined Functions (UDFs)

User-defined functions (UDFs) are custom functions that extend the calculation and data-import capabilities of Excel. Developers create custom calculation packages to provide:

- Functions that are not built into Excel
- Custom implementations to built-in functions
- Custom data feeds for legacy or unsupported data sources and application-specific data flows

Users who create workbooks can call UDFs from a cell using formulas exactly like calling built-in functions.

Excel Services UDFs provide the ability to use formulas in cells to call custom functions written in managed code (e.g., C# or Visual Basic .NET) and deployed to Microsoft Office SharePoint Server 2007.

Users can create UDFs to:

- Call custom mathematical functions
- Get data from custom data sources inserted into worksheets
- Call Web services from UDFs

Managed UDFs (e.g., UDFs written in managed code) are required for Excel Services. UDFs written in native code (e.g., XLLs written using C API) cannot be used directly with Excel Services, and need to be wrapped to expose managed interfaces that Excel Services expects to see.

With Excel client, the situation is different. XLLs can be accessed directly from the client, whereas managed code UDFs need to be wrapped with an XLL add-in for direct access. Another way to automate Excel client spreadsheets is via managed add-ins created with Visual Studio Tools for Office 2005 (VSTO). With this approach, one can use managed code to segment all computationally intensive logic into an add-in (a .NET assembly) that is then deployed to the desktop, and made available to Excel spreadsheet by adjusting Code Access Security in .NET Framework.

Additional information is available from MSDN [articles](#) and [blogs](#).

## Excel Services

Excel Services is a feature of Microsoft Office SharePoint Server 2007. It can be used to deploy Excel 2007 spreadsheets to a server, where they get processed by a server-based calculation engine. Excel Services takes advantage of Windows SharePoint Services, including enterprise content management features such as document checking in and out, versioning, auditing, alerting, and other capabilities. It addresses enterprise needs for “the single version of the truth” in spreadsheet management.

Excel Services delivers spreadsheet results as HTML code rendered with remarkable fidelity in a Web browser for unprecedented reach while maintaining user interactivity. Workbook authors can create fully interactive, data-bound workbooks that include charts, tables, and PivotTable reports as part of a portal application. Courtesy of some specialized Web parts, Excel Services can be used to build powerful business intelligence dashboards or scorecards. In addition, Excel Services can be invoked and made to deliver results programmatically to other enterprise applications via Web services technology, thereby re-using business logic that has so far been locked inside spreadsheets. Spreadsheets remain stored and

versioned securely in a central datacenter location, without ever compromising or even disclosing sometimes IP sensitive business logic.

The Excel Services architecture consists of a Web front-end and an application-server tier. The Excel Calculation Services loads requested workbooks and performs any required calculations. Excel Web Access displays the results in HTML, and Excel Web Services provides a Web services interface to enable applications to access the workbooks.

Excel Services is built on ASP.NET and Microsoft Windows SharePoint Services 3.0 technologies. Additional [information](#) is available on MSDN.

## **Windows Compute Cluster Server (CCS)**

Windows Compute Cluster Server (CCS) brings together the power of commodity x64 (64-bit x86) computers, the ease of use and security of Active Directory service, and the Windows operating system. This combination provides a security-enhanced and affordable high-performance computing solution that is seamlessly integrated with mainstream business functions.

CCS can be easily and quickly deployed using standard Windows deployment technologies. Additional compute nodes can be added to the compute cluster by simply plugging in the nodes and activating them. Microsoft Message Passing Interface (MPI) implementation is fully compatible with the reference MPICH2 implementation. Integration with Active Directory enables role-based security for administration and users. The use of Microsoft Management Console provides a familiar administrative and scheduling interface.

When integrating CCS with spreadsheet based workloads, it is recommended that all computationally intensive logic be contained in standalone .NET executables that are pre-deployed to compute nodes. Using CCS API from UDFs and managed add-ins, one can offload spreadsheet workloads to CCS managed compute cluster or grid.

Additional technical [information](#) is available from TechNet.

## **The GigaSpaces Solution**

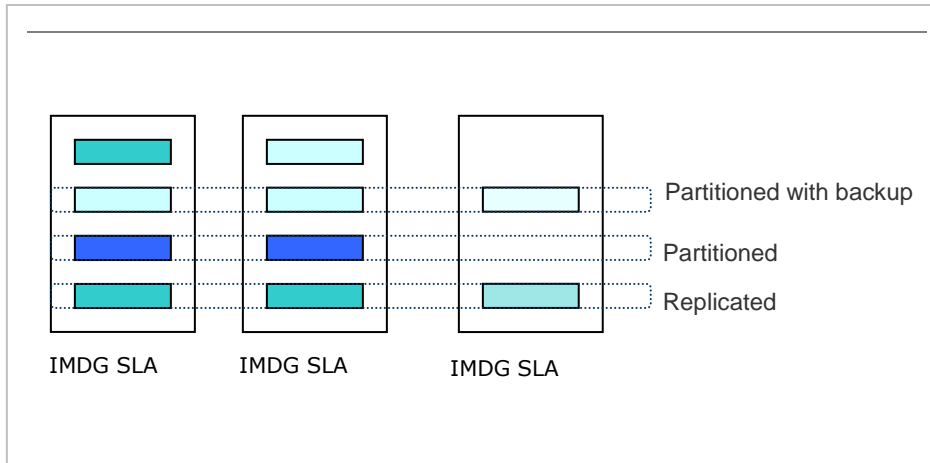
GigaSpaces eXtreme Application Platform (XAP) is middleware for building applications with low latency and limitless scalability. It simply and easily transforms new and existing applications into scalable services with increased performance.

XAP provides a complete middleware platform for managing data, messages, and business logic for applications that require high performance and the ability to scale horizontally across hundreds of machines. XAP combines a Service Level Agreement (SLA)-driven Data Grid together with a unique

design referred to as Space-Based Architecture (SBA). Additional [information](#) about XAP, as well as product [download](#) is available from GigaSpaces Web site.

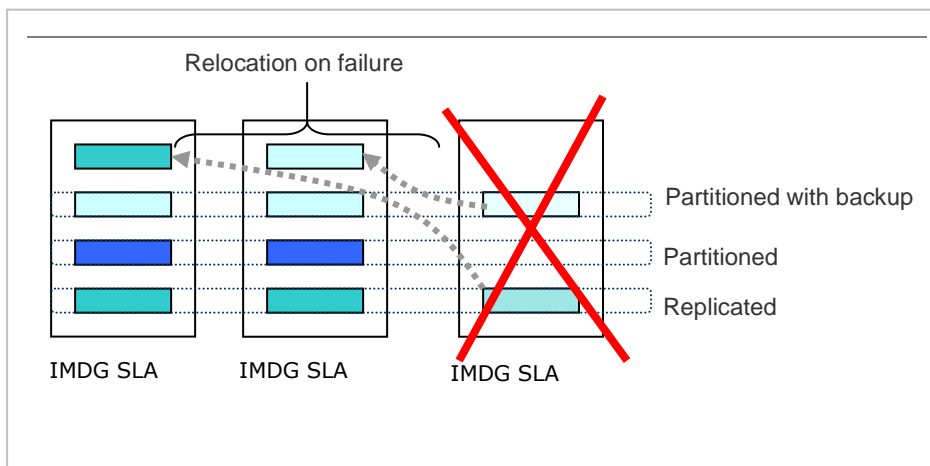
## An SLA-Driven Data Grid

The GigaSpaces In-Memory Data Grid (IMDG) introduces the notion of an *SLA-driven container*, a generic hosting environment for data grid instances. GigaSpaces data grid instances run inside containers driven by Service Level Agreements (SLAs), which provide them with unique capabilities.



**Figure 1. Data Grid instances running in SLA-driven containers**

Figure 1 illustrates how several IMDG instances, deployed in SLA-driven containers, can be clustered into different data grid topologies — replicated, partitioned, and partitioned with backup — across a pool of shared hardware resources.



**Figure 2. Data high availability through SLA-driven containers**

A key advantage of this approach is that the SLA-driven containers maintain continuous high availability. As Figure 2 illustrates, if one of the containers fails, it is automatically relocated to an available container. The state of the instance is recovered automatically before the relocated instance becomes available, ensuring that the application accessing the data continues to work without interruption.

## Meeting the Requirements of Data Awareness

GigaSpaces can enable data awareness in two ways, each relevant to a different operational scenario:

Scenario	Method of Providing Data Awareness
<b>IMDG instances deployed directly by CCS</b> (without SLA-driven containers)	<b>Integration using affinity keys</b> —CCS and users submitting tasks share special keys that identify the relevant data to each task. This way, CCS can execute tasks on the same machine as the relevant data (see <a href="#">Section 4.3</a> ).
<b>SLA-driven containers launched by CCS</b> (each container launches the relevant IMDG instances)	<b>Integration using implicit data awareness</b> —data-intensive procedures can run in the SLA-driven container, together with the IMDG instances. As the container itself is data aware, data affinity can be guaranteed, without making the CCS itself data aware (see <a href="#">Section 4.4</a> )

## In-memory Data Grid Unique Features

Aside from data awareness, GigaSpaces IMDG offers all the standard features of a high-end IMDG product, such as built-in distributed topologies including replication, partitioning, and high availability. Additionally, GigaSpaces offers the following unique features:

Features	Benefits
<b>Extended and Standard Query</b> based on SQL, and ability to connect to IMDG using standard JDBC connector	Makes the IMDG accessible to standard reporting tools.  Makes accessing the IMDG just like accessing a JDBC-compatible database, reducing the learning curve.
<b>SQL-based continuous query support</b>	Brings relevant data close to the local memory of the relevant application instance

<b>Central management, monitoring, and control</b>	Allows the entire Data Grid to be controlled and viewed from an administrator’s console.
<b>Mirror Service</b> —transparent persistence of data from the entire IMDG to a database or other data source	Allows seamless integration with existing reporting and back-end systems.
<b>Real-time event notification</b> —application instances can selectively subscribe to specific events	Provides capabilities found in messaging systems, including slow-consumer support, FIFO, batching, pub/sub, content-based routing.
<b>Local and distributed transactional support</b>	GigaSpaces IMDG contains powerful built-in transactional semantics both for in-process and distributed operations.

### Space-Based Architecture in a Nutshell

In Space-Based Architecture (SBA), one develops a software component that can accept a task (such as a service or user request), **perform all steps of the transaction on its own**, and provide a result. This component is a **processing unit**—a mini-application that can perform the entire business process for a relatively small number of requests.

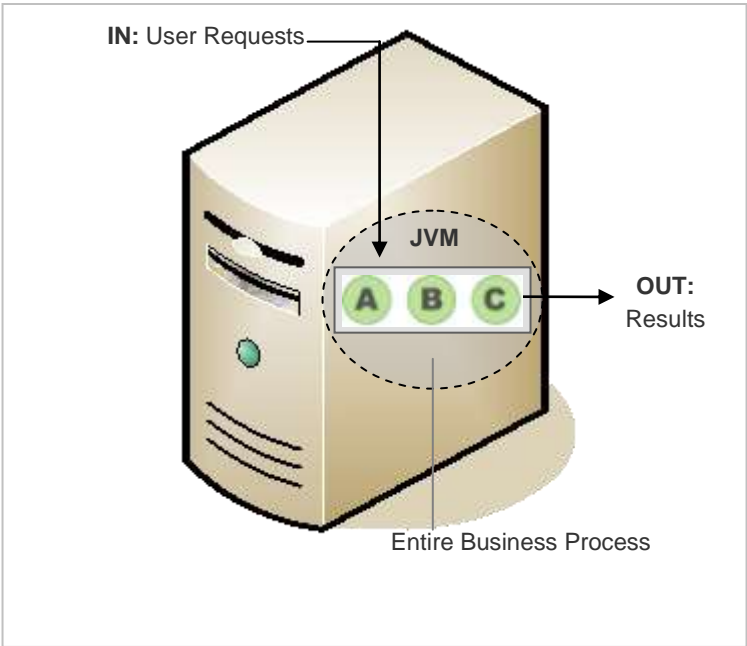


Figure 3. Processing unit

The idea of a processing unit removes the need for sharing of state information and partial results among different components of the application running on different physical machines. Information only needs to be shared **within the processing unit**—in other words, within the confines of the local machine and the same process. Therefore, each processing unit can manage its own workflow, as illustrated in Figure 3.

As processing units are self-sufficient, each processing unit provides data storage facilities to its own service instances, i.e., each processing unit contains a built-in IMDG instance. Given that all services are collocated on the same machine, the information they share among them can be stored in local memory. This means that, within the processing unit, communication and coordination can take place with the lowest possible latency—that of in-process, in-memory access. For more information, refer to white [paper](#) available from GigaSpaces Web site.

## Combined Solution

This section presents the proposed Microsoft-GigaSpaces solution for scaling Excel-based applications, and discusses key benefits to the end user.

## Solution Overview

Figure 4 illustrates how the combination of GigaSpaces and Microsoft technologies provides an end-to-end scalable, high-performance solution (from market feed to trader's desktop).

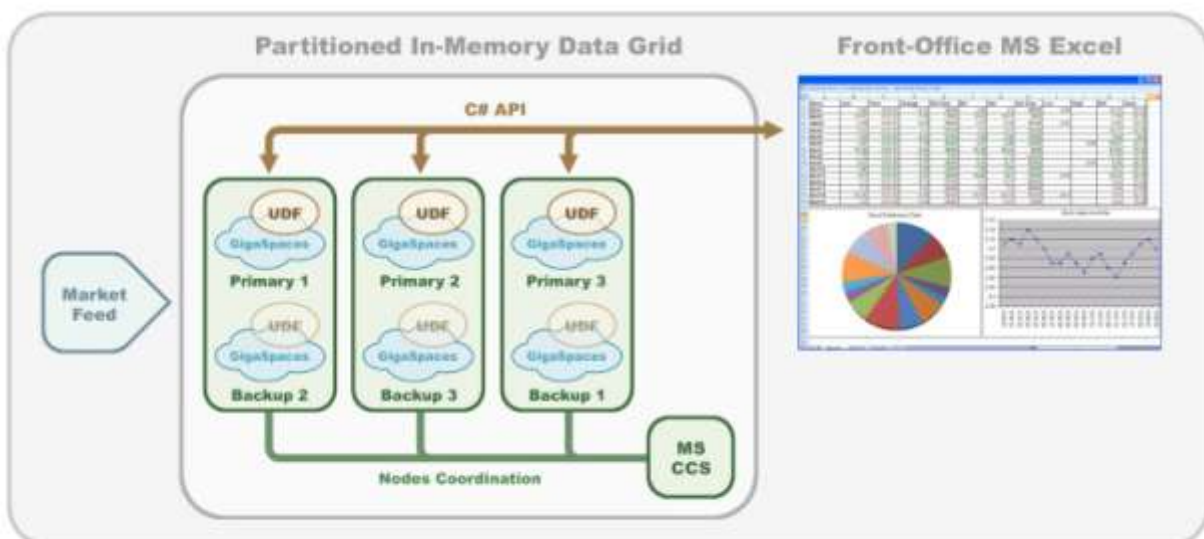


Figure 4. Solution Architecture

GigaSpaces partitioned IMDG hosts computational logic contained in .NET executables and deployed to compute nodes. Together, GigaSpaces data-aware scheduling and CCS node coordination mechanism route the incoming data to the appropriate compute node, i.e., the node in the GigaSpaces cloud where the computational logic that is meant to process the data resides. This approach guarantees collocation of data and computational logic for lowest latency and highest performance.

Taking a typical market data use case for example, high-volume market feeds arrive at the data grid, are routed to the correct node for complex computation, and the results are then conveyed to the trader's desktop for simple calculations and presentation in the Excel workbook. GigaSpaces partitioned IMDG ensures that this "processed data repository" is reliable and easy to scale.

We present three approaches for moving the processed data between the GigaSpaces IMDG and Excel-based application:

1. In the first approach, we use Excel Services/UDF as a calculation engine and **Excel Web Access** to present the workbook as a Web page.
2. In the second approach, Excel Services is used as the central calculation engine, and **Excel workbook** is used for local calculations and as the presentation layer.
3. In the third approach, we use **Excel as a standalone client**.

In the following section we provide a detailed overview of each approach.

## **Approach #1: Excel Services and Excel Web Access**

### **Data access via UDFs**

UDFs are deployed on the Microsoft Office SharePoint Server where they are hosted by Excel Services. UDFs are written using managed code, and they use CCS API to invoke job scheduler, which in turn distributes tasks to .NET executables pre-deployed to compute nodes. .NET executables retrieve data from the partitioned in-memory data grid (based on supplied parameters such as a Stock Symbol). As part of the .NET executable, a static member is initiated for the required cached data.

The locally cached data is a subset of the master IMDG data, enabling the client to read relevant data without any remote operations. The data is streamed into the client view in an implicit manner. The local view is continuously updated by the master IMDG in asynchronous mode (using notifications). If needed, the .NET executable manipulates the retrieved data, e.g. by creating an average calculation for a data range.

It is very important to note that .NET executables are developed in C# code, and can be connected to IMDG using GigaSpaces high-performance .NET API.

### **New Excel workbook template creation**

The organization's Excel experts create their own Excel workbooks based on the predefined UDFs, add their own calculations and graphs, and save the Excel workbook in the SharePoint repository. Using

Web Parts, Excel spreadsheets can be included in enterprise portals. Moreover, individual end users can request that spreadsheets be rendered (via Excel Web Access) inside their Web browsers.

### **Client-side presentation refresh**

The end-user presentation layer is refreshed by sending a recalculate request from the end-user's browser to Excel Services. The refresh is done via JavaScript that locates the HTML IFRAME element, and sends a recalculate request to Excel Services.

Optimization is achieved by first checking if a data change event occurred on the server and only then performing a recalculate and refresh operation.

### **Approach #2: Excel Services and Excel Client**

Data retrieval is performed by UDFs in the same manner as presented in Approach #1. Excel Services process the data combined with values based on the Excel workbook's parameters. The processed data is then used directly inside the Excel workbook. Excel Services functions as a "processing unit" while the Excel workbook is used for client-side calculations and graphical rendering.

The mechanisms of getting the data, issuing the calculation requests, and notifying the client software on any data change events all make use of the locally cached data for fast performance. The user can then use Excel's built-in functionality, such as functions and charts to customize the view to user's specific needs, and perform additional local calculations.

The result is a robust and efficient architecture:

- Since the models are all based on Excel workbooks, it is possible for business analysts to update changes as needed and propagate to all users with minimum effort.
- Having Excel as the landing pad for the information, enables trading floor analysts (who typically are proficient Excel users) to work in a familiar environment, and design workbooks that assist them in their day-to-day work.
- Processing is facilitated by the local IMDG instance (local cache), which takes care of connecting various components and ensuring that the data is only updated as required, reducing unnecessary network and machine load.

### **Approach #3: Excel as a Standalone Client**

In this approach, the Excel workbook communicates directly with the IMDG via a UDF implementation. Users develop a client-side UDF and deploy it on their local machine. The UDF contains managed C# code. It retrieves and manipulates data from the IMDG (via the local view) using GigaSpaces high-performance .NET API.

## **Server-Side Components**

### **Partitioned In-Memory Data Grid**

IMDG stores the data, e.g. trade data, in a shared pool of machines. The partitioning of data across instances is based on one of the data fields, e.g. a stock symbol. Using a pool of machines removes memory limitations. Common data calculations and data analysis (requiring a large quantity of data as input) run on the pool as well, or, more specifically, on the actual machine holding the data.

### **Windows Compute Cluster Server**

CCS can be used for coordinating and scheduling the deployment of GigaSpaces XAP nodes. The client resident UDF uses CCS API to schedule jobs across compute nodes. XAP collocates processing logic contained in pre-deployed .NET executables with data retrieved from IMDG. Because XAP handles the processing logic as managed services, it is fully aware of the data and business logic semantics. GigaSpaces and CCS are, therefore, complementary to one another: GigaSpaces XAP manages business process execution, while CCS manages resources, including node provisioning and job scheduling.

## **Integration with Excel 2003 and 2007 Clients**

This section describes two ways to leverage GigaSpaces XAP IMDG with Excel 2003 and 2007 workbooks, without the use of UDF and Excel Services.

### **Excel Bridge**

A small “bridge” application is deployed on the client machine where the Excel workbook runs. The bridge application registers as a listener to the IMDG. On any data change event, the bridge application receives a notification (which carries the new or updated data) from the GigaSpaces IMDG. The data is sent to the Excel spreadsheet using the Excel bridge application.

### **Real-Time Data (RTD)**

Real-time data processed by IMDG can be output to Excel client via Excel real-time data (RTD) components, and displayed using the familiar spreadsheet look and feel. RTD servers are COM DLLs that implement a specific interface provided by Excel. Displaying real-time data works via push-pull mechanism, whereby RTD server notifies Excel of data changes, and Excel in turn requests changed data from RTD server.

By using the RTD function in the Excel client workbook, data can be retrieved from the local instance of the IMDG. The local instance is continuously updated by the master IMDG in an asynchronous mode, i.e., using notifications (a “push” approach).

## **Solution Benefits**

The following section describes numerous benefits of the combined solution.

### **Handling Large Volumes of Data in Memory**

The GigaSpaces XAP IMDG enables data distribution across an essentially unlimited number of nodes, making it possible to handle extremely large data volumes in memory. This is achieved by partitioning data across different instances and having only a portion of the data in each instance. The Excel-based application views the clustered spaces as one large memory cloud, where .NET executables containing computational logic serve as the client, routing each operation call to the data partition to fetch or store data objects.

Queries on the data can treat the cluster as either a “Black Box” – one large cloud, without drilling down to specific partitions – or as a “White Box”, in which data is fetched from specific partitions.

### **Low-Latency Data Access from Compute Nodes**

Because all the data is held in random access memory, there is no need to access external databases and/or external memory devices, eliminating significant latencies normally introduced by disk I/O and network traffic. Processing is executed on the same machine where the data resides, ensuring minimum latency.

A GigaSpaces SLA-driven container is capable of running one or more processing units, together with one or more IMDG instances that can load relevant data, save partial results, and participate in topologies that change dynamically to meet service-level agreements. The middleware infrastructure underlying GigaSpaces IMDG instances, called a “space”, can also be used to enable messaging between different sub-components of the processing units. Therefore, the container becomes a self-sufficient unit that can be scaled out by adding additional instances.

As a result, the logic driven from UDF/Excel Services and the corresponding data are collocated in the same process, and the data is accessed at in-process speeds without the usual network delay caused by accessing data in a remote data source.

### **Dynamic Scalability**

The solution described in this paper enables dynamic scaling both up and down the cluster size while maintaining high performance. Combining GigaSpaces XAP with Windows Compute Cluster Server job scheduling functionality, the cluster topology can change dynamically to meet SLA requirements. For example, if one application increases its data throughput, relevant instances are automatically spawned for that application, shutting down instances that serve other applications, and distributing data evenly between the remaining instances.

Excel users can now rely on a dynamic on-demand cluster to serve an infinite number of Excel-based applications.

## Continuous High Availability

With this solution, GigaSpaces IMDG instances are replicated in such a manner that every partition always has one or more backups, and these backups never reside on the same physical box as the primary partition. If the machine fails, the .NET executables containing processing logic can be transparently routed to an identical instance on the backup container without experiencing any downtime or service discontinuity.

In addition, the GigaSpaces SLA-driven container provides a self-healing environment. When the primary partition fails, the backup becomes the primary, and another identical partition instance is immediately spawned automatically, therefore ensuring that a backup is always instantly available.

## Free-Up Desktop Resources

With the proposed solution, all computations and data processing are performed on the server side, not on the desktop computer that runs the Excel workbook. The desktop machine is therefore freed-up to handle other tasks, such as fast rendering of the presentation layer in Excel, as well as other programs that user wishes to run.

## Interoperability and Integration

GigaSpaces XAP runs in a Java Virtual Machine, and can therefore be deployed on any hardware platform and operating system — Windows and non-Windows. This flexibility enables Excel-based applications to work with back-end non-Windows server clusters.

Moreover, because XAP is [compatible](#) with the J2EE standard, front-end and middle-tier .NET/Windows components can interact with a J2EE application server without compromising performance or ease of integration. GigaSpaces support for both the Spring Framework and the .NET Framework can make the integration happen with minimal code and architectural changes.

Because GigaSpaces provides a high-performance C# API and computations are collocated with data, interoperability can be obtained while maintaining or increasing the existing application performance. Traditional performance constraints accompanying interoperability architectures are eliminated.

## Reuse of Existing Code

GigaSpaces provides a full featured C# API based on a declarative approach. This means that developers who have already built UDF/Excel Services applications can continue to use their original C# code without performing fundamental changes or learning a new API. In fact, all that is required to integrate a C# code with the Space is to add code annotations. Please refer to additional [information](#) on GigaSpaces .NET API.

## Multi-Tenancy

Running the GigaSpaces IMDG instances in SLA-driven containers enables several topologies to co-exist, and it also enables dynamic changes to the IMDG topology to meet SLA requirements, without any downtime.

The front-end Excel applications, as well as the middle-tier components (UDF/Excel Services), can now utilize the same cluster nodes for several applications simultaneously, contributing to improved resource utilization and lower total cost of ownership.

## **Excel as an SOA Front-End**

One of the benefits of the solution is that Excel-based applications are now a natural fit in a service-oriented architecture. The business logic components driven from UDFs, as well as the data (in the IMDG) are treated as loosely-coupled services. This means that changes in one UDF/Excel Services have minimal effect on others, and therefore mainstream Excel-based business applications can be an integral part of SOA implementations.

## **Running Mainstream Business Functions on the Grid**

The end-result of all of these benefits is that mainstream business applications based on Excel can run on the grid – a distributed, high-performance, service-oriented environment.

## **Customer Use Case Example: Trading Analysis**

In this section we describe a customer application (trading analysis) that leverages solution benefits described previously.

The customer, one of the world's largest global investment banks, required real-time insights into risk and P&L information of its trading business. They experienced high latency compounded by constantly growing data volumes. The customer was, therefore, looking for a near real-time analytics solution that could scale.

### **The Challenge**

The customer's main trading analysis tool is Office Excel. While users loved the ease-of-use and flexibility of the tool, it faced several challenges:

- Difficult to scale, and, therefore, limited in its ability to handle additional users and growing trading data volumes
- Computations are done mainly on the user's desktop computer, which created latencies and reduced user productivity
- Long processing times, which means that reports were not produced in a timely fashion
- High maintenance and lack of consistency among traders (many versions of "the truth")

## The Solution

GigaSpaces XAP was utilized to collocate processing logic and data on the server side, while retaining Excel client on the front end. The next step in the solution is to combine this architecture with Excel Services.

A shared IMDG is created to distribute trading data across a pool of machines. Common data analyses can also run on this pool of servers, thereby leveraging the available memory and CPU resources, and freeing up end user's desktops. UDFs are defined to retrieve required data from the IMDG.

A new Excel workbook is created to call the predefined UDFs, and the workbook is then published to Excel Services, making it consistently available to all users.

## The Results

- Highly scalable architecture
- Linear performance improvement made possible by adding further processors/memory
- Improved performance by a factor of six
- Fail-over and high availability
- Very low incremental cost in scaling as trade volumes grow
- The ability to use cost-effective commodity hardware
- Significant code re-use between different versions of the application, across different products
- Easy to maintain

## Conclusions

Microsoft and GigaSpaces teamed up to provide a comprehensive solution for mainstream business Excel-based applications in securities and capital markets industry. The solution addresses the needs of customers who demand high performance and low latency from applications typically found in front and middle offices. By combining Microsoft Office Excel, Office SharePoint Server, and Windows Compute Cluster Server with GigaSpaces eXtreme Application Platform, the solution enables Excel-based applications to process very large volumes of data with very low latency. Business logic is offloaded to compute nodes where it is collocated in-process with relevant data fetched by in-memory data grid. The solution offers dynamic scalability, high availability, and self-healing fault tolerance.

## References

[GigaSpaces Web site](#) – Provides information about GigaSpaces, its products, and available services including product [downloads](#)

[MSDN Financial Services Industry Center](#)