App Servers - NG:

Characteristics of The Next Generation Application Servers

Guy Nirpaz,
VP R&D and Chief Architect
GigaSpaces Technologies
Who am I?

- 2 Years with GigaSpaces
  - VP of R&D
  - Chief Product Architect
- Veteran of several Startups
- Spent most of my life in design and architecture of complex systems
  - Financial Services, Command and Control, Teleco
  - Mercury, IBM, others
- Blog @ jroller.com/gnirpz
About GigaSpaces Technologies

Provides Application Platform product (XAP) for applications characterized by:

- High volume transaction processing and
- Very Low latency requirements
- Large Data Volumes

Scaled-Out Application Server - GigaSpaces XAP

- In-Memory Data Grid
- Service Grid
- Java, .NET and C++

Customer Base

- Financial Services
- Military and Defense
- Online Services: Retail, Banking, Gaming
Abstract

Why a new class of application servers is needed and how is it related to cloud computing and scaling out architectures
# Scalability - A Business Driver

- **Business Agility** - *Must be able to scale when needed*
- **Cost effectiveness** - *Must process an increasing volume of information faster at a lower cost.*

## Capital Markets:
- Algorithmic trading, Market Data, Risk Analysis, Portfolio Analysis, Surveillance/Compliance

## Telecom:
- Real-time billing, Order Management, VOIP, Location-based services, Mobile Device Content

## Online:
- Banking, Gaming, Travel, Advertising/Marketing, Commerce, Search Engines

## Defense:
- Real-time Intelligence, Pattern Analysis
Facing Unpredictable Growth

Seasonal Peaks

Online Services
- Slashdot, Digg, TechCrunch Effect
- Rapid, unexpected customer demand/growth
animate them so they come one after the other

A.B.S17

Allt Bar Sadeh; 11-03-2008
Inability to Rapidly Scale Leads to Disaster

April 20th, 2007

BlackBerry: database problem caused the crash

Posted by Russell Shaw @ 5:49 am

Categories: News, Outages

Tags:

Colleague Marguerite Reardon reports that BlackBerry-maker Research In Motion has finally issued a statement identifying the cause of the 14-hour service disruption earlier this week.

In a statement, RIM has "determined that the incident was triggered by the introduction of a new, noncritical system routine that was designed to provide better optimization of the system’s cache."

Hmm...can it be done right?

We’ll get to it in a few slides...

The Dow's technical glitch, dissected

Publisher expects normal operations on back-up system, but still looking for root cause of Tuesday’s glitch.

The publisher of the Dow industrials said that a system problem starting at 1:50 p.m. ET on Tuesday, amid unusually heavy trading volume, caused a 70-minute lag during which the value of the market measure lagged the declines in the underlying stocks.
Barriers of Scaling:

- Architecture

- Middleware Implementation
  - Application Server
  - Database Server
  - Messaging Server

- Operations
Jeff Bar, Amazon:

“If you build your application for dynamic scalability, you will be able to benefit from the compute power available on EC2...”
Traditional Tier-Based Architecture

Silo Approach
- Independent hardware and software
- Multiple skill sets
- Separate models to design, deploy, test, monitor and manage
- Integration required
A Transaction Flow Example - Order Management

- Submit Order
- Validate
- Check/match
- Execute order
- Notify Completion
- Register
- Perform Query
- Data Access Logic
- Messaging Infrastructure

-Too many network hops
- Unacceptable latency
Maintaining Resiliency in a Traditional Tiered Application

- Separate failover strategy and implementation for each tier
- Integration points are not addressed
- Redundancy increase network traffic
- Latency is increased

- Validate
- Check/match
- Execute order
Scaling and Managing a Traditional Tiered Application

- Scalability is not linear
- Scalability management nightmare

Business tier

Data feed

Back-up

Data Tier

Back-up

Messaging Infrastructure

Back-up

Messaging Infrastructure

Back-up

Messaging Infrastructure

Data Tier

Back-up
Space Based Architecture

Process Unit
- Self sufficient unit of scale
- Combination of Data, Processing and Messaging
- Model for development, testing and deployment (OSGi)

Principles of Partitioning
- Data Partitioning
- Processing Partitioning

Content Based Routing
- Affinity between data/messaging and processing location

Interaction Model Abstractions

See Wikipedia for further details:
Scaling .... made simple!

✓ Single, efficient process to scale your application
✓ Linear scalability
✓ Write Once Scale Anywhere
✓ Automated, SLA-driven engine for deployment, monitoring and management engine
Scalability Challenges in Existing Systems

Scaling is Difficult and Expensive:
- Large Up-Front Investment
- Invest Ahead of Demand
- Load is Unpredictable

70% of Web Development Effort is “Muck”:
- Data Centers
- Bandwidth / Power / Cooling
- Operations
- Staffing

Source: Jeff Bar, Amazon 2008
The Dream Solution: Scale-Out On Demand

Scale-Up

High End Server

Scale-Out On-Demand

The Cloud (commodity servers)
The Missing Piece

Existing Middleware Implementation

Development Framework
Spring

Messaging  Services  Data

Silo  Silo  Silo

Existing JEE middleware was not designed to run on a cloud!
Scaling on the Cloud - what does it mean?

- Everything virtualized (OS, Middleware, Deployment,..)
- Self-healing
- SLA-driven
- Multi-tenancy
- Service-oriented
- Data, Data, Data
The Solution - Virtualization & SOA

“SOA separates functions into distinct units (services), which can be distributed over a network and can be combined and reused to create business applications. These services communicate with each other by passing data from one service to another, or by coordinating an activity between two or more services” (Wikipedia)

“Virtualization is a technique for hiding the physical characteristics of computing resources from the way in which other systems, applications, or end users interact with those resources. This includes making a single physical resource (such as a server, an operating system, an application, or storage device) appear to function as multiple logical resources; or it can include making multiple physical resources (such as storage devices or servers) appear as a single logical resource.” (Wikipedia)
Combining SOA and Middleware Virtualization

Development Framework

**Spring**

Middleware Virtualization

- Messaging
- Services
- Data

Virtualization and SOA used for Next Gen middleware

Deployment Virtualization

OS Virtualization

The Cloud

Our Spring-based App remains untouched!
Middleware Virtualization - Core Principles

Development Framework

Spring

Middleware Virtualization

API Façade

- Messaging
- Services
- Data

Common Virtualization / Clustering layer

Deployment Virtualization

OS Virtualization

The Cloud

Abstracting The API from the physical resources
Common clustering

- Provide common clustering services to enable high availability, synchronization, and matching-- for all APIs.
- Core supported topologies
  - Replicated
  - Partitioned
  - Partitioned with Backup
  - Master/Local
- Transactional support
- Data affinity
- Location transparency
Data and Messaging Virtualization
Data Virtualization

Keep your existing database in-sync

Application

GigaSpaces Proxy

Local Cache

JVM

GigaSpaces

Master

Database

Application

GigaSpaces Proxy

Local Cache

JVM

Common Virtualization / Clustering layer

Middleware Virtualization
Service Virtualization How to?

**RPC – Client -> Server**

**SOA – Client -> Virtual Service**
Services Virtualization Attributes

- Location Transparency
- Dynamic Service Discovery
- Parallel Execution - Batch Mode
- Parallel Execution - Map/Reduce with Data Affinity

Let’s explore each...
Location transparency - Scenarios

Collocated Proxy:
Common cluster and implementation on the same server.

Remote Client:
Client is remote to service implementation. Sync or Async

Remote & Collocated:
Collocated services can be accessed remotely and locally, simultaneously.
Dynamic Service Discovery

1. **Bind**: Service Implementation
   Connects to Common Cluster

2. **Find**: Client discovers Common Cluster

3. **Invoke**: Content based routing

- Client is not bound to a specific service.
- Services can be found using multicast or unicast.
Parallel Execution - Batch Mode

- Client executes multiple asynchronous operations in a loop
- No need to wait for results
- The client blocks only till the requests are registered with the space.

- Services take those requests (through the polling container) and execute them in parallel.
- Results can be retrieved by the client who submitted the request through future handle.
The text on this slide is not clear;
- you talk about a polling container but it's not mentioned in the diagram
- not sure what you mean by "client blocks requests,
- etc.

please explain
Alit Bar Sadeh; 12-03-2008
Parallel execution - Map/Reduce (with data affinity)

- The space act as a data bus containing both messaging and data.
  - Enables routing request messages as data, without the need for external coordinator.

- Consistency is simpler to handle
  - Since external transaction coordinator is redundant, a simple local transaction is sufficient.
Remember Our Stack?

Development Framework

Middleware Virtualization

API Façade

Common Virtualization / Clustering layer

Deployment Virtualization

OS Virtualization

The Cloud

A.B.S22
Our Spring-based App remains untouched!

Virtualization and SOA used for Next Gen middleware

Let’s review this layer...
A.B.S22  add balloons
Alit Bar Sadeh; 11-03-2008
Deployment Virtualization (SLA Driven Container)

**Deployment Virtualization**

- Grid Service Manager (GSM)
  - Deploy/Undeploy
- Application Server (AS)
  - AS-1
  - AS-2
  - AS-N
  - SLA Driven Container
- SLA:
  - Failover policy
  - Scaling policy
  - System requirements
  - Space cluster topology
- PU Services beans definition

**OS Virtualization**

**The Cloud**

**SLA Driven Container**

**A.B.S23**
I think that the slide is a bit too detailed, let's see if we can make it "thinner".

Also, I just realised that we're calling the PUs Application Servers (as you requested), but we didn't actually mentioned that before so it will be confusing.

Alit Bar Sadeh; 12-03-2008
The OpenSpaces Framework

- An extension of The Spring Framework
  - Deployment units for scaling and fail-over - Processing unit
  - SLA definition (primary/backup, system requirements,..)
  - Middleware virtualization
- Seamless integration of Spring with the GigaSpaces clustering
- Integration with other development frameworks:
  - Mule/ServiceMix - ESB
  - Lucene - Indexing service
A.B.S24  are you sure you want to mention "processing units"? we never discussed this in this ppt
A.B.S24  Alit Bar Sadeh; 12-03-2008

A.B.S25  Not sure this slide is clear, lets discuss in the office
A.B.S25  Alit Bar Sadeh; 12-03-2008
Adoption Options for Existing Applications

- **Option 1: Deploy existing Spring application on SLA driven containers**
  - The only change is the packaging
  - (no EAR files, use an OSGi-like deployment package)

- **Option 2: Services Virtualization Framework**
  - POJO driven Session Beans can seamlessly be turned into scalable services by plugging new remoting implementation

- **Option 3: Migrating the messaging tier to a “virtual messaging bus”**
  - Seamless - assuming the use of JMS
  - Mule can be used to abstract the message flow

- **Option 4: Migrating the data-tier to an In-Memory Data Grid**
  - Relatively seamless - assuming DAO abstraction and declarative transactions are used
  - API choices (JCache/Map, GigaSpaces)
are you talking here about steps or options??

The title is not clear and is not in synch with the bullets

Aliit Bar Sadeh; 12-03-2008
## Comparison with existing App Servers

<table>
<thead>
<tr>
<th>J2EE Application Server</th>
<th>Scale-out App Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tier-based</td>
<td>Tier-less, Scale-out</td>
</tr>
<tr>
<td>XA transaction</td>
<td>XTP (XA is not required)</td>
</tr>
<tr>
<td>EJB -&gt; EJB3.0</td>
<td>POJO data model and POJO services</td>
</tr>
<tr>
<td>RDBMS centric</td>
<td>In-memory data grid</td>
</tr>
<tr>
<td>Messaging/Data are separated</td>
<td>Integrated In-memory data-bus</td>
</tr>
<tr>
<td>Bloated implementation - (not flexible, tightly coupled, etc., unless using Spring,..)</td>
<td>Focus on runtime and provide integration with existing dev framework</td>
</tr>
<tr>
<td>Static deployment</td>
<td>SLA driven deployment</td>
</tr>
<tr>
<td>N/A</td>
<td>Service Virtualization framework</td>
</tr>
<tr>
<td>N/A</td>
<td>Built-in optimization for SaaS deployment (Ec2)</td>
</tr>
<tr>
<td>N/A</td>
<td>Built-in integration with existing web containers</td>
</tr>
<tr>
<td>Web container</td>
<td>Support of parallel computing patterns</td>
</tr>
<tr>
<td></td>
<td>Map reduce - parallel processing</td>
</tr>
<tr>
<td></td>
<td>Mater/Worker</td>
</tr>
</tbody>
</table>
not sure about the title, let's discuss
Ait Bar Sadeh; 12-03-2008
Solution For Scaling:

- Architecture - Scale Out!
  - Logical Layers - No Physical Tiers
  - Partitioning into Processing Unit
  - Modularity (OSGi)
- Middleware Implementation - Next Generation Application Server
  - Virtualized
  - Single Scaling Model
  - SLA Based
- Operations - Amazon EC2, Google, Sun, Your own IT, Others...
Demo on EC2
Questions?
Questions?