From Tiers to Services Without Web Services

Nati Shalom
CTO & Founder

Overall Presentation Goal

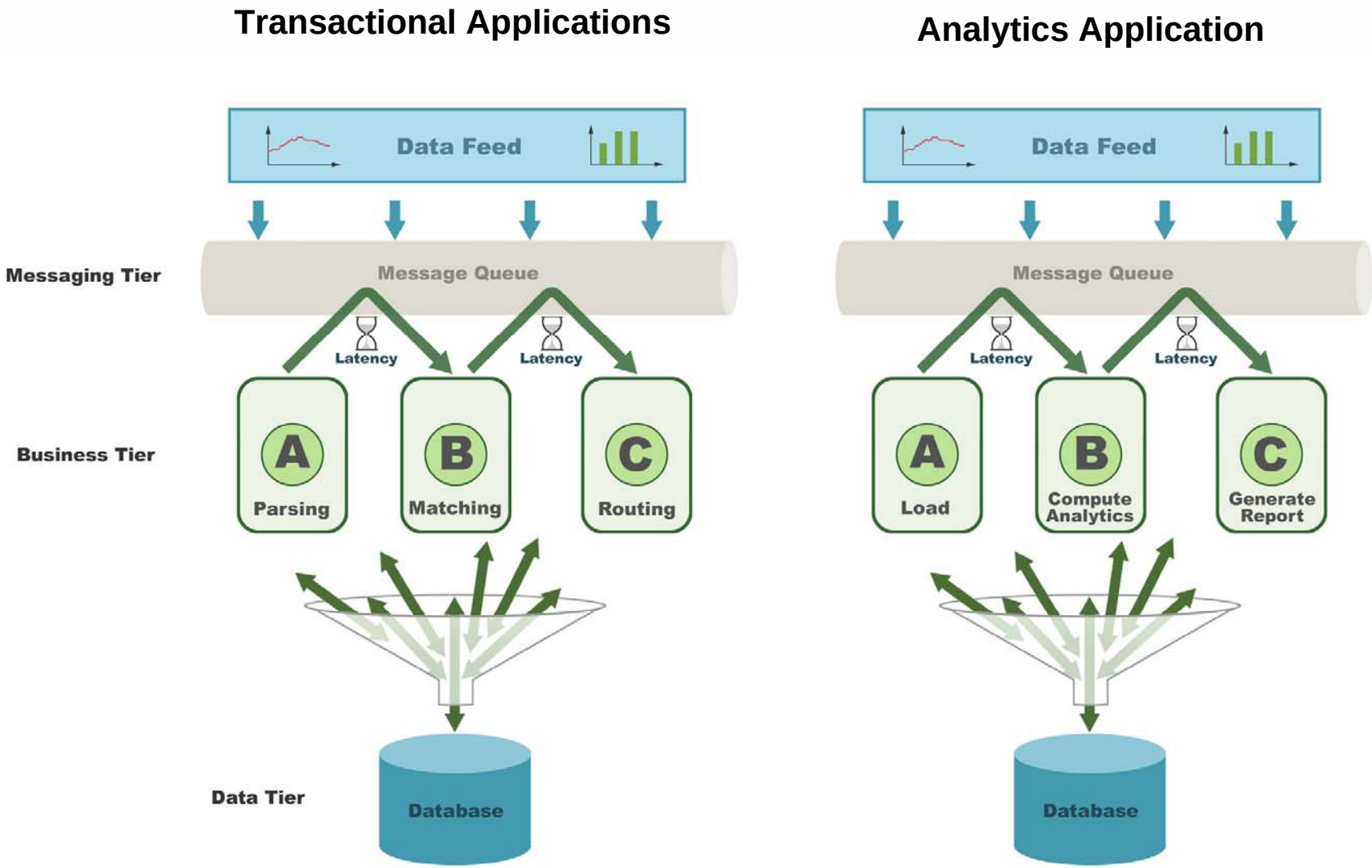
- **Understand how to design SOA solutions for applications that require high-performance and low-latency**

- **Understand how to use tools such as Spring and Space Based Architecture (SBA) to transform existing applications into scalable services in a very simple manner**

About GigaSpaces

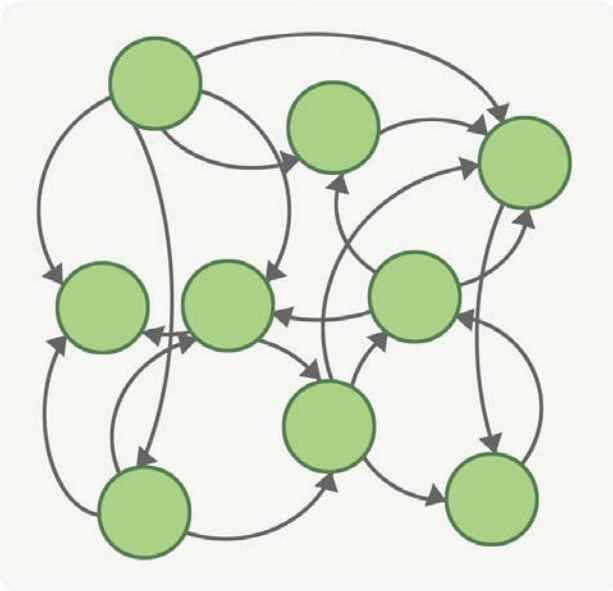
- **Founded in 2000**
- **Provides infrastructure software for applications characterized by**
  - High volume transaction processing and
  - Very Low latency requirements
- **Distributed (Grid) Application Server**
  - In Memory Data Grid (Caching)
  - Compute Grid
  - Messaging Grid
- **Customer base**
  - Financial Services
  - Telecom
  - Defense and Government

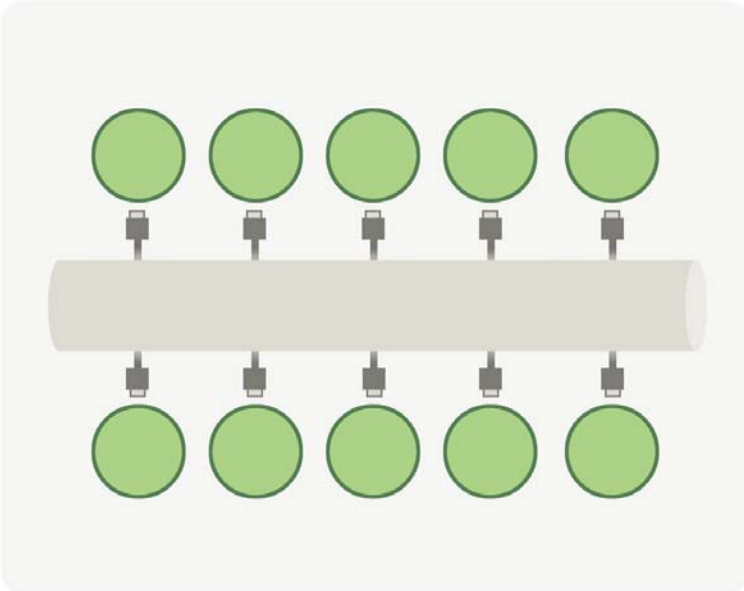
What's wrong with this picture?



From Tier Based to Services

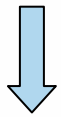
Application Centric → Service Centric





Initial Statements about SOA and Stateful Applications

- **Organizations move to SOA for easy integration, development flexibility and leverage, and business agility**
- **Common Interface used by SOA is Web Services**
  - Web Services are slow, complex and involve much chattiness
- **Stateful Applications require**
  - Low latency - Minimize data transformation and Network overhead
  - Linear and Dynamic Scalability



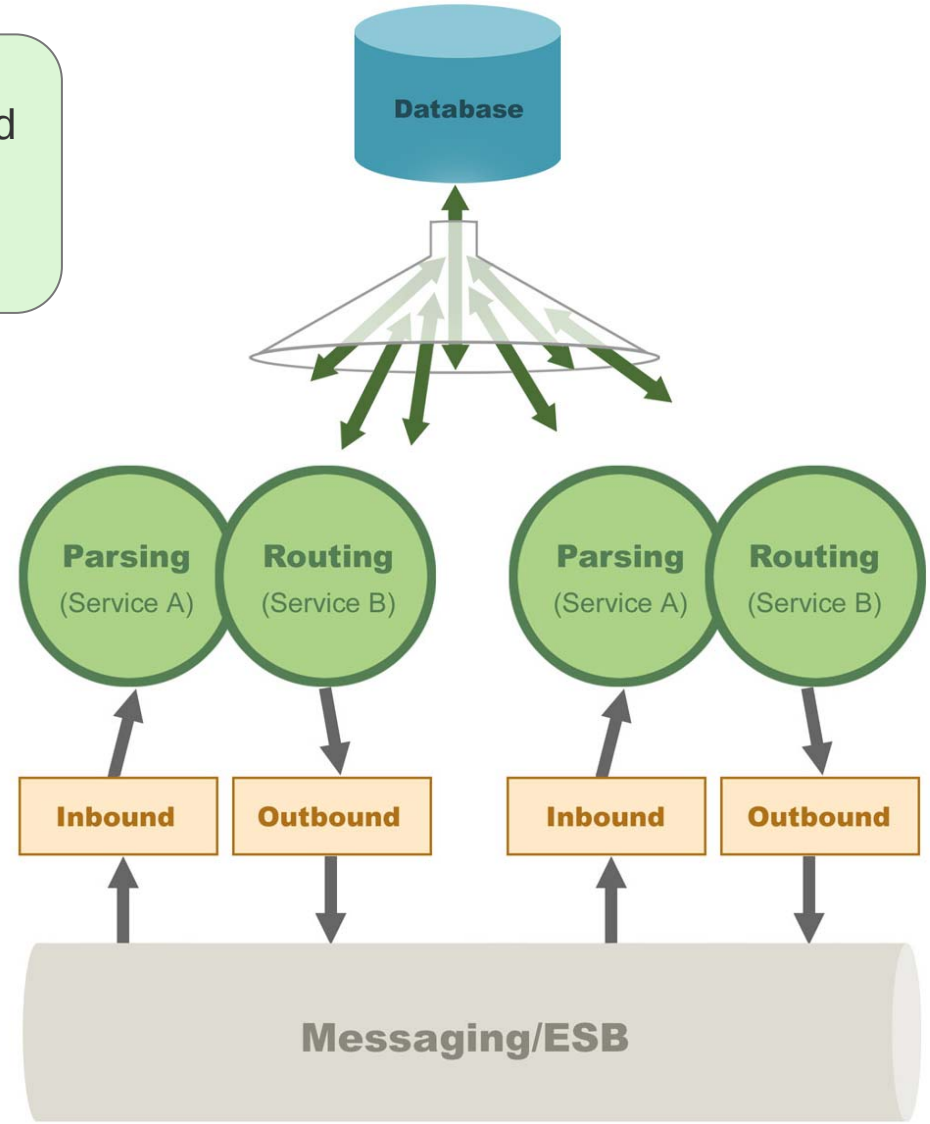
- **Requirements of stateful applications are not met by standard SOA platforms**

Typical SOA model based on ESB

Most ESB solution do not address stateful services and often use a centralized database to share state between services

Services can be Java, C++, .Net

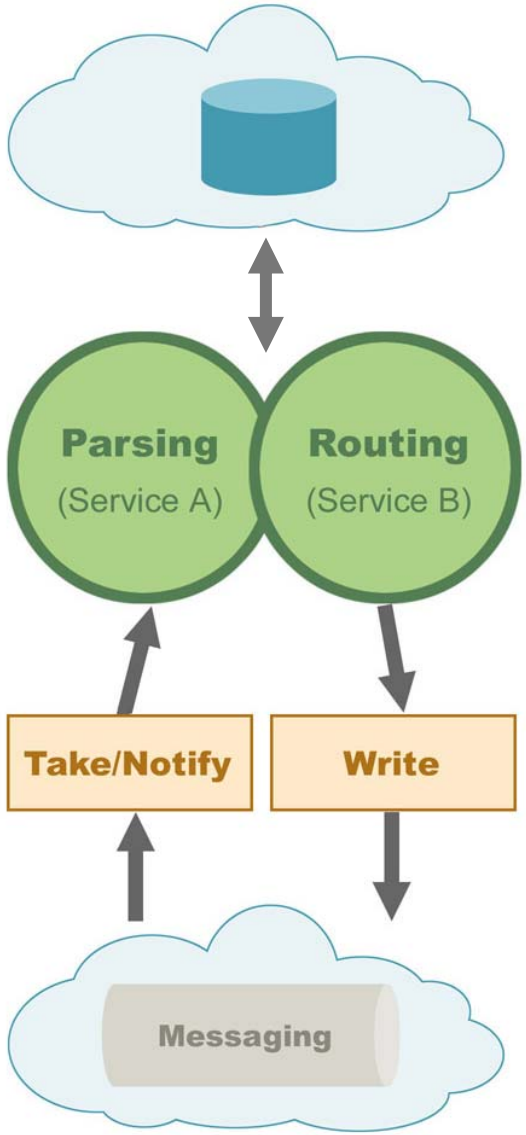
Content-Based Routing



SOA on Space - Handling Stateful Services

- **Space provides Messaging and Data using the same underlying technology**

- **Simple! Only 4 Verbs API**
  - Read, Write, Take, Notify
  - Details in the appendix of this presentation

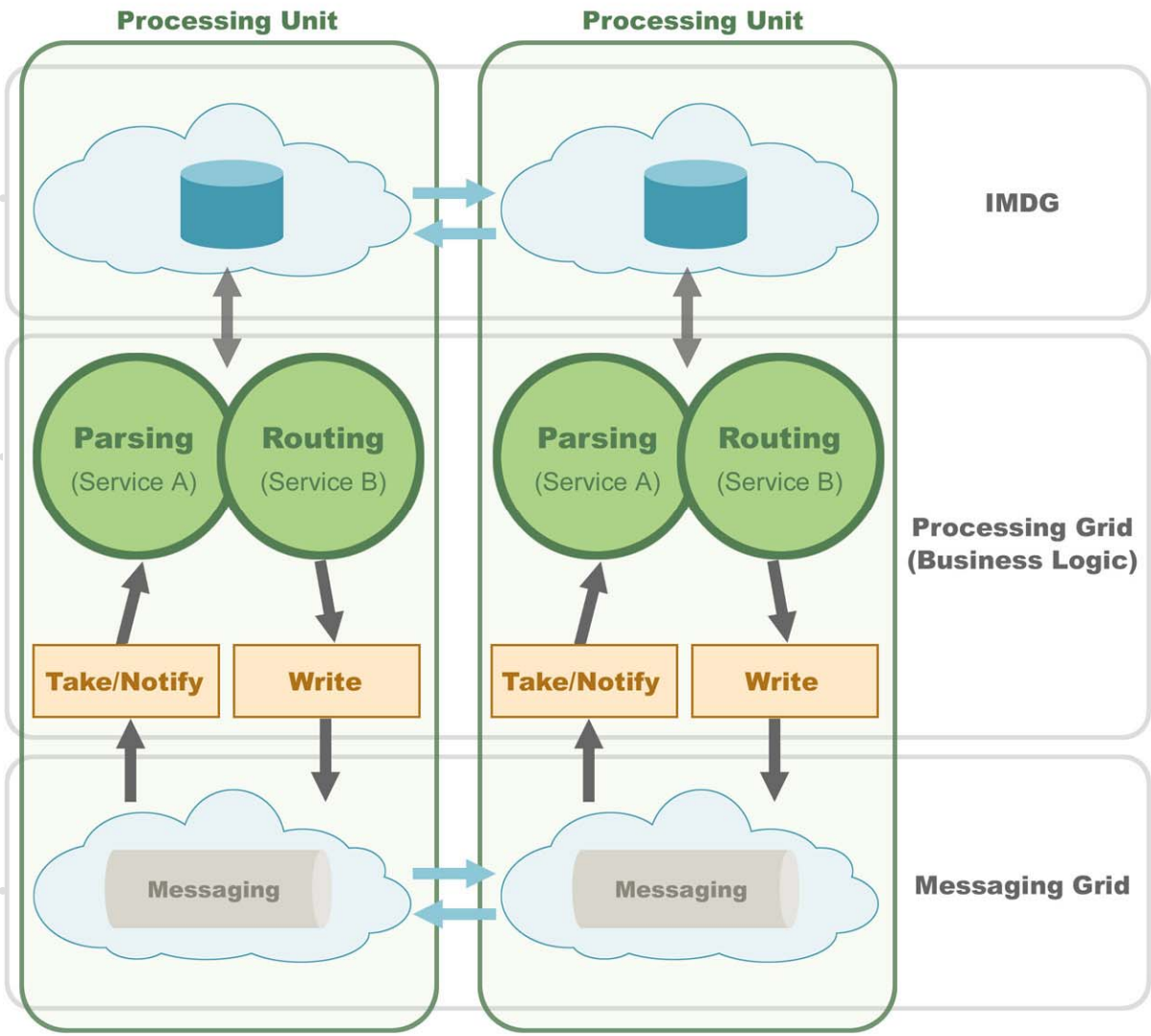


SBA – Real-time SOA for Stateful Services

Shared state to enable stateful services

Services can be Java, C++, .Net

Content-Based Routing



SBA and SOA

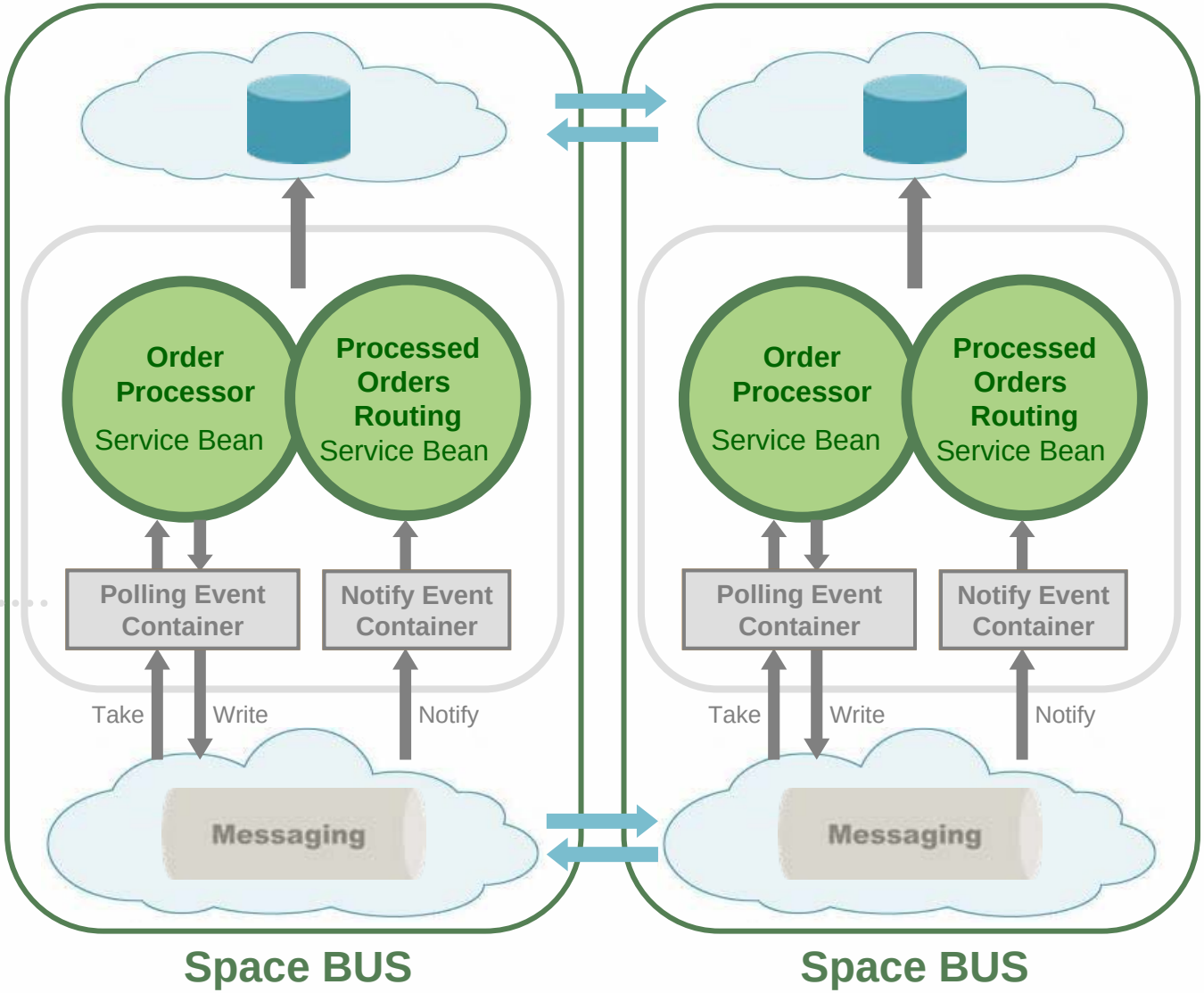
- **SBA is a technology to implement loosely coupled services and achieve:**
  - Performance -
  - True Linear Scalability using the “share-nothing” approach
  - Reliability
- **SBA maintains at least two of the key principles of SOA:**
  - The ability to compose applications from services
  - Services are loosely coupled

Can we make it even simpler???

Introducing Declarative SBA Using Spring

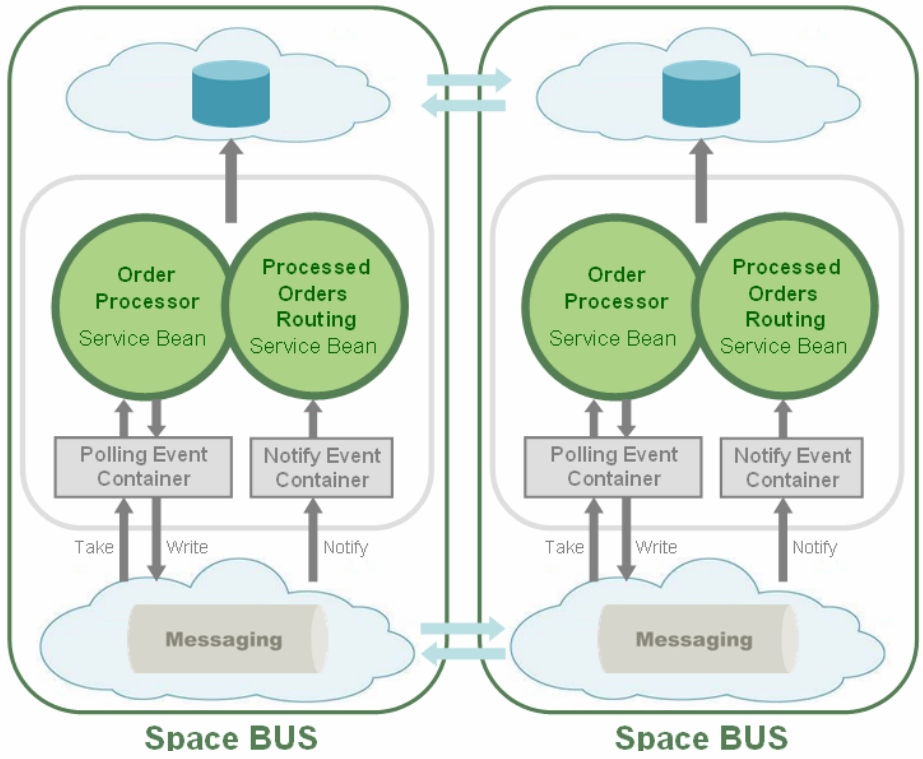
Declarative Spring-SBA - Making Scalability Even Simpler

Declarative Abstraction



SBA – Making Scalability Simple!

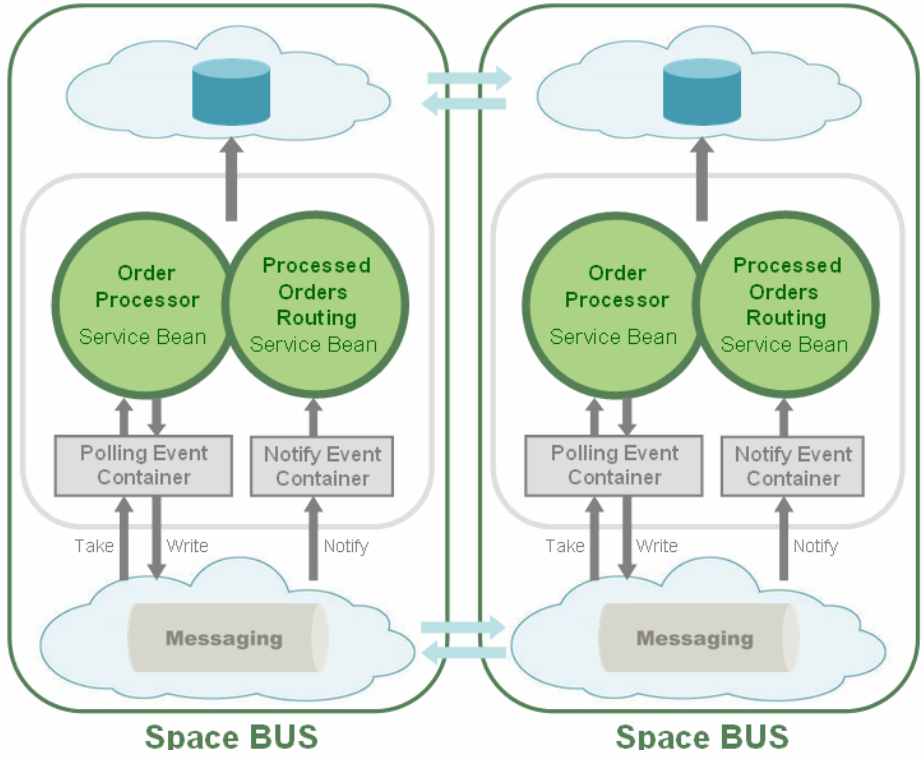
- **Write Once Scale Anywhere**
  - Testability – 90% of the functionality can be tested on a single VM
- **Simple as Spring!**
  - Fit natively with spring environment
  - Everything done through spring
- **Look as one**
  - View the entire cluster as if it was a single server when executing queries, registering for events or writing data



A closer look at Declarative Spring SBA

Declarative Spring-SBA – How it works..

- **Step 1:**
  - Implement POJO domain model
- **Step 2:**
  - Implement the POJO Services
- **Step 3:**
  - Wire the services through spring
- **Step 4: (Optional)**
  - The client side
  - Adding space based remoting



The POJO Based Data Domain Model

```
@SpaceClass
public class Data implements Serializable {

    public static long[] TYPES = {1, 2, 3, 4};

    private String id;

    private Long type;

   @SpaceId(autoGenerate = true)
    public String getId() {
        return id;
    }

   @SpaceRouting
    public Long getType() {
        return type;
    }

   public void setProcessed(boolean processed) {
        this.processed = processed;
    }

}
```

SpaceClass indicate that this is a SpaceEntry – SpaceClass includes classlevel attributes such as FIFO,Persistent…

SpaceId used to define the key for that entry.

SpaceRouting used to set the data affinity i.e. define the partition where this entry will be routed to.

Order Processor Service Bean

```
public class DataProcessor implements IDataProcessor {

    @SpaceEvent
    public Data processData(Data data) {
        data.setProcessed(true);
        data.setData("PROCESSED : " + data.getRawData());
        // reset the id as we use auto generate true
        data.setId(null);
        System.out.println(" ------ PROCESSED : " + data);
        return data;
    }
}
```

SpaceEvent
annotation marks the processData method as the one that need to be called when an event is triggered

Wiring Order Processor Service Bean through Spring

```
<bean id="dataProcessor“ class="com.gigaspaces.pu.example1.processor.DataProcessor" />

<bean id="dataPollingContainer" class="com.gigaspaces.pu.event.PollingEventContainer">
    <property name="gigaSpaces" ref="gs" />
    <property name="concurrentConsumers" value="1" />
    <property name="template">
        <bean class="com.gigaspaces.pu.example1.Data">
            <property name="processed" value="false" />
        </bean>
    </property>
    <property name="eventListener">
        <bean class="com.gigaspaces.pu.event.adapter.annotation.SpaceEventListenerAdapter">
            <property name="delegate" ref="dataProcessor" />
        </bean>
    </property>
</bean>
```

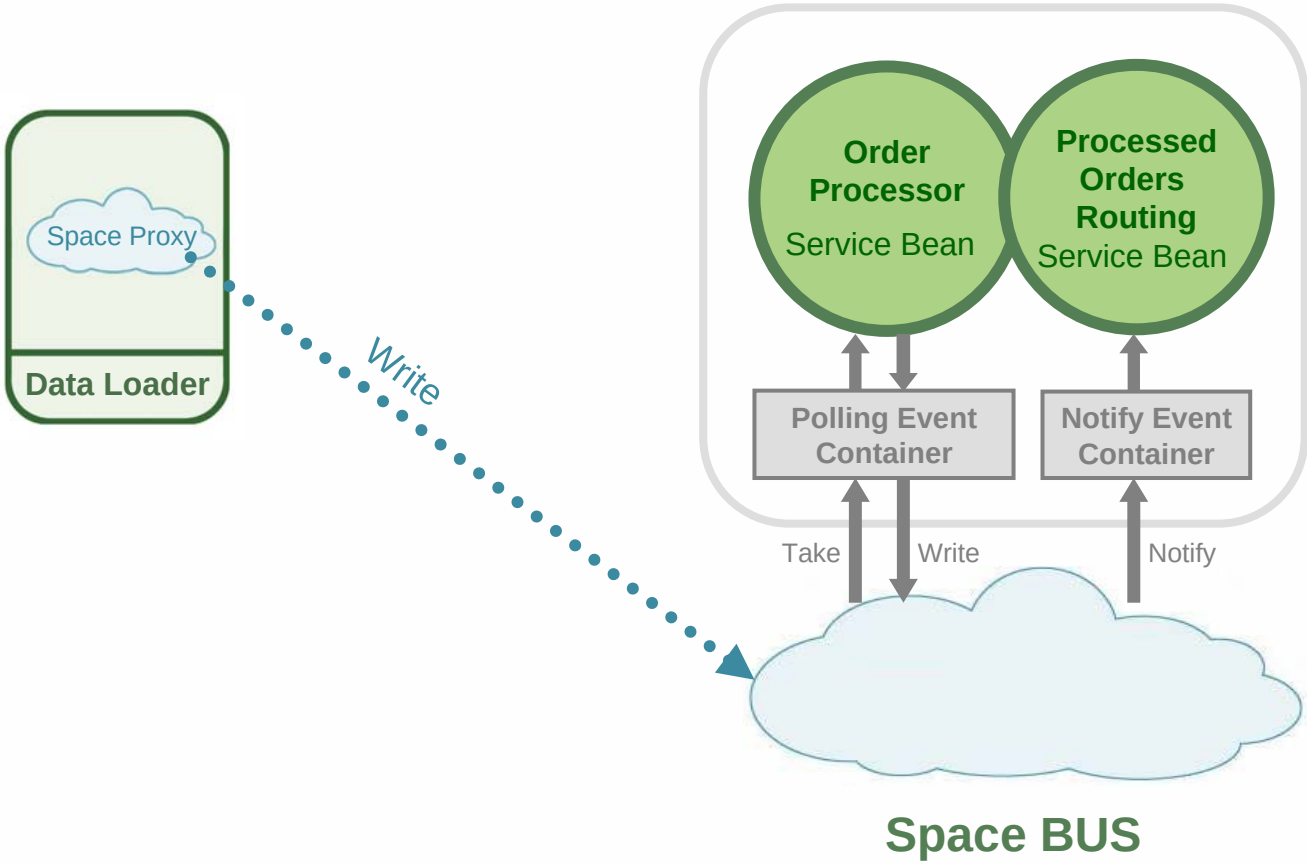
The PollingEventContainer will implicitly call take with template defined in the tempate property and invoke the method marked with @SpaceEevent on dataProcessor bean.

Processed Orders Routing Service Bean

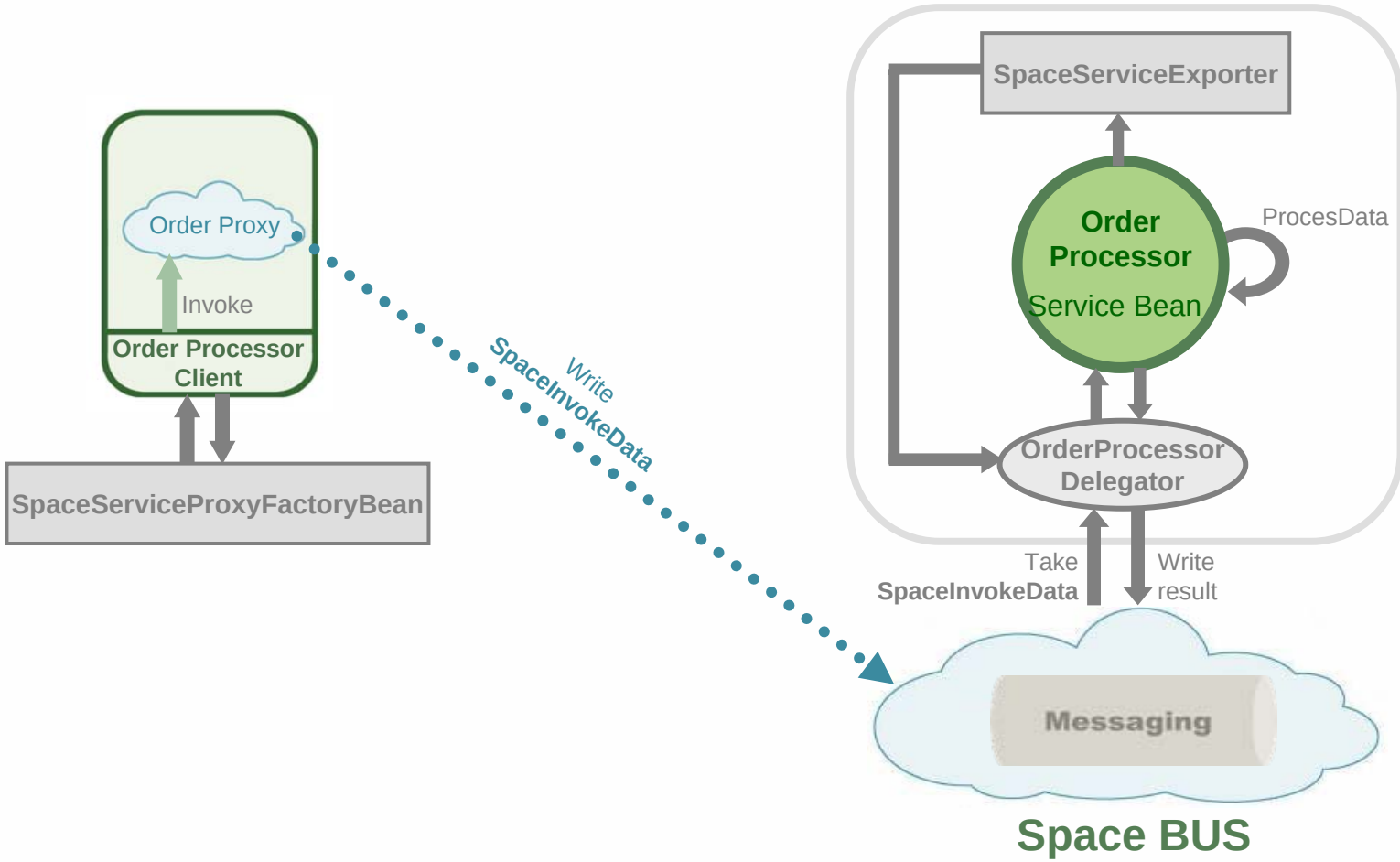
```
public class DataProcessedCounter {

    private long count;

    @SpaceEvent
    public void onData(Data data) {
        System.out.println("---- DATA PROCESSED COUNT [" + count++ + "]");
    }
}
```

SpaceEvent annotation marks the processData method as the one that need to be called when an event is triggered

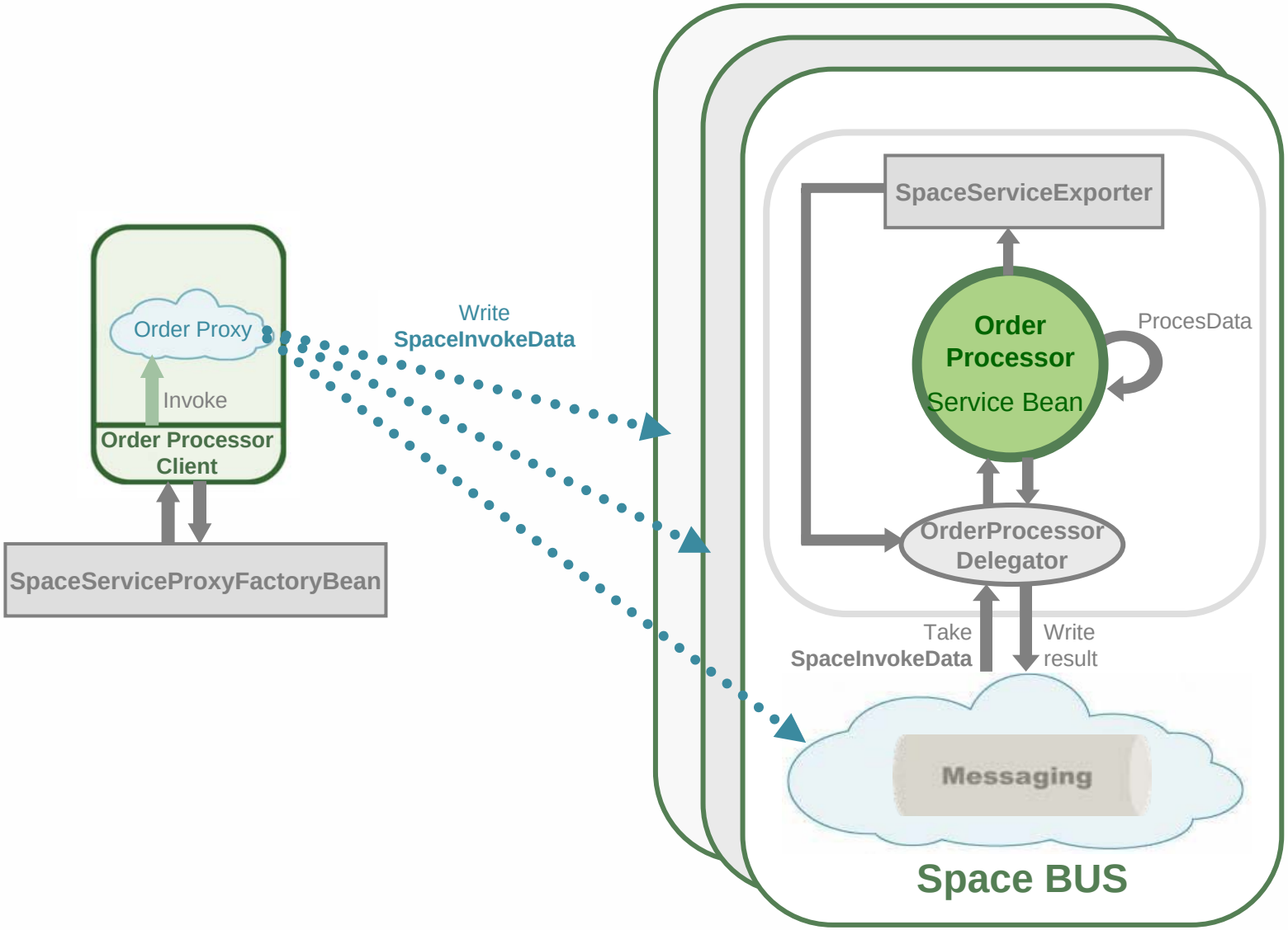
Direct Data Loader Client



Space Based Remoting



Space Based Remoting – Inherent Scalability/Reliability



So Why use a Declarative SBA?

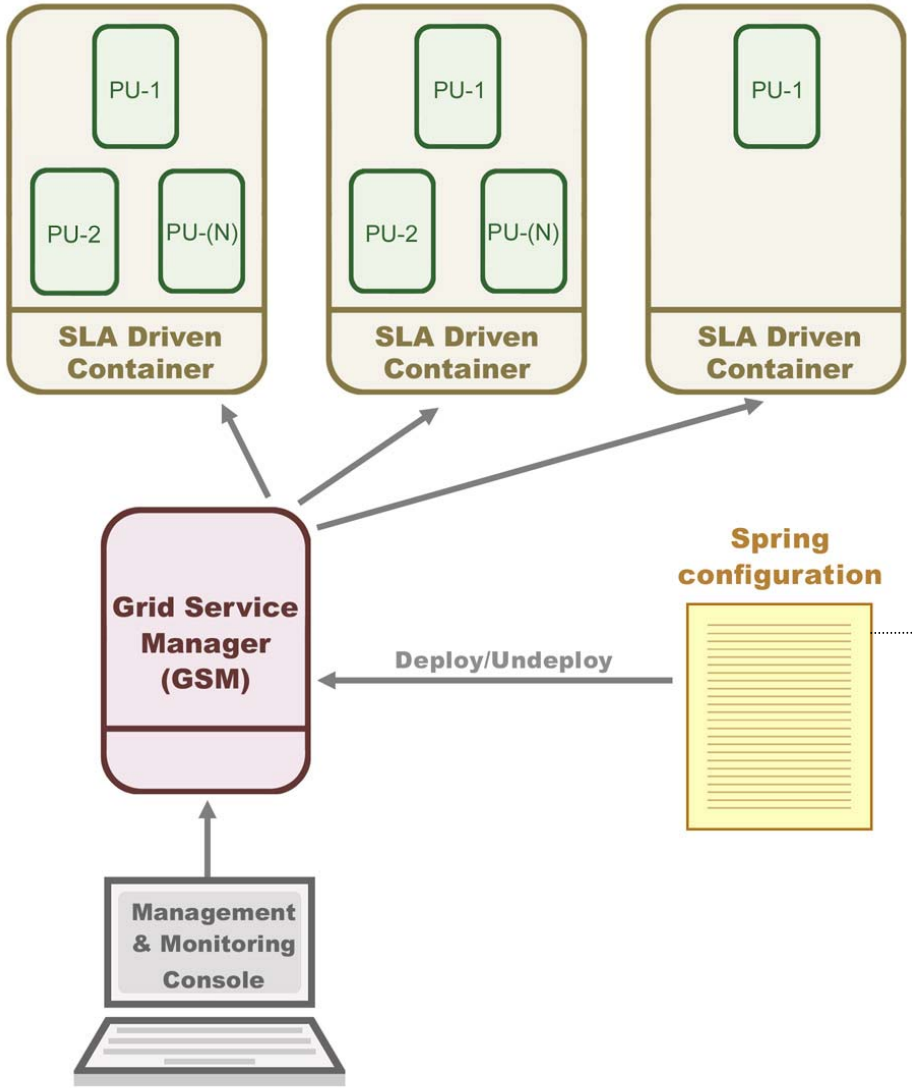
- **Simple!**
  - Building a fully fledged distributed application becomes just a matter of wiring simple local beans
  - Its all declarative through annotation or XML
- **Non intrusiveness through abstraction**
  - High level abstraction provides higher degree of flexibility to plug-in different solutions from different topologies, technologies and standards without compromising on the least common denominator
- **Extremely Extensible**
  - Very easy to extend the model and the underlying implementation without changing the application
- **Reduced learning curve – Its just Spring!**
  - Fits into common development framework and infrastructure
  - Fits natively to the existing Spring model
  - Utilizes common practices such as annotation, IoC etc.

Ensuring SLAs are met…

SLA Driven Application Service Container

- **Provides built-in support for deployment of Spring based applications**
- **Virtualize the network and physical resources from the application**
- **Handles Fail Over, Scaling and Relocation policies using SLA based definitions.**
- **Provides distributed dependency injection to handle partial failure and deployment dependency.**
- **Provides single point of access for monitoring and management**

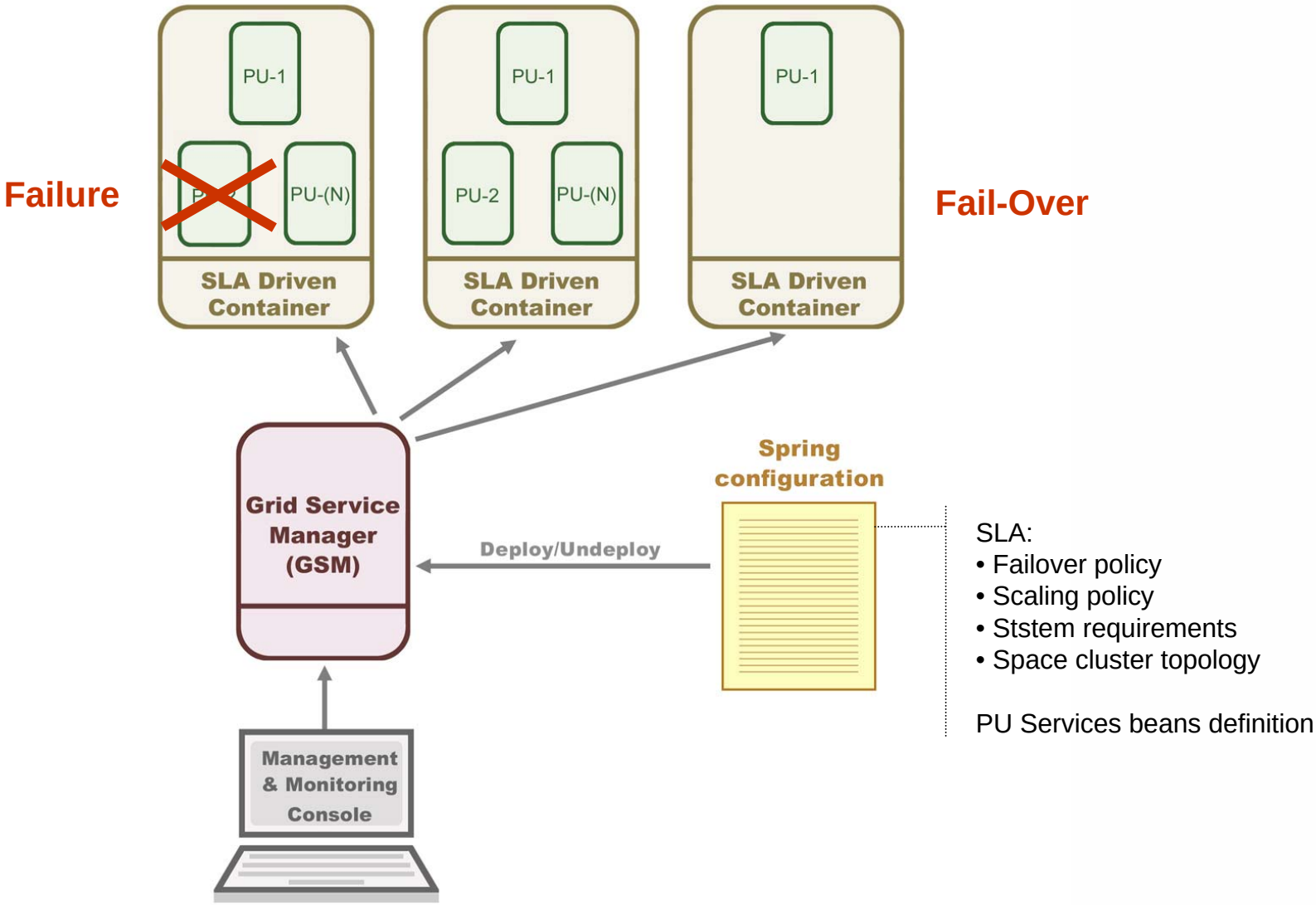
SLA Driven deployment



SLA:
- Failover policy
- Scaling policy
- Ststem requirements
- Space cluster topology

PU Services beans definition

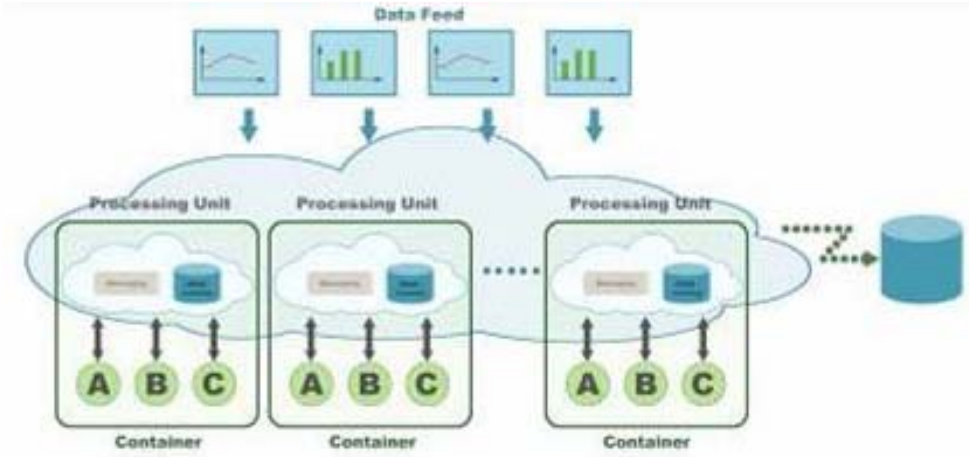
Continues High Availability



SBA Use Cases:
Analytics and Transactional Applications

FX Trading

- **Business Challenge – DB is the FX trading WW leader for the past 5 years, and has no plans to become #2…**
  - Ever increasing trade volumes, and introduction of new currencies (and commodities)
  - Business Requirement: 10ms end-to-end order processing latency
  - Comply with orders' scheduling and auditing regulations

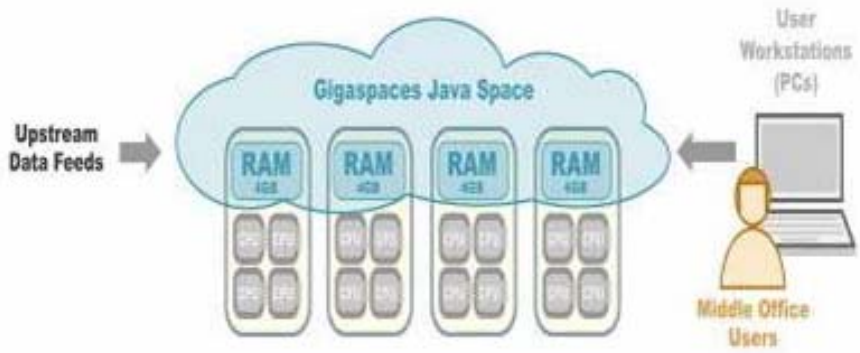


- **Technical Challenge**
  - Envisioned solution was complex. It was requiring 3 different technologies to build up the new system (Application Server, Messaging and Caching)
  - Seamless scalability, and HA
  - Zero impact on existing clients (internal and external)
  - Persist the workflow to meet regulations w/o affecting throughput
  - Vendor agnostic APIs for new clients and applications

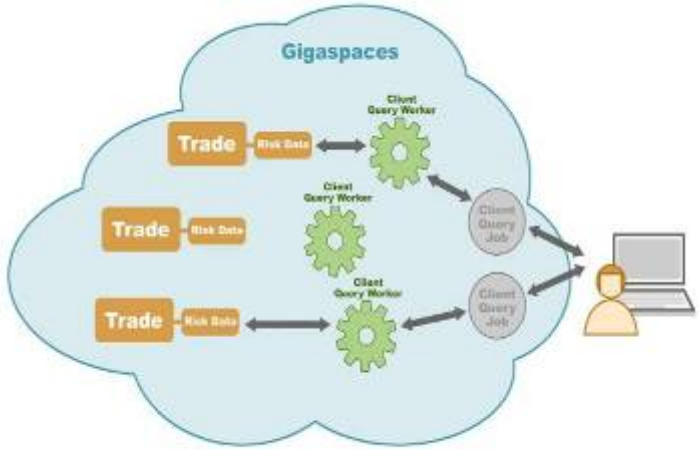
- **Results**
  - Classical SBA that results in a linearly scalable application (partitioned by currency-pair)
  - POC performance: 2.5ms order processing latency vs. 10 ms requirement
  - Zero changes to clients, writing JMS messages
  - Heavy use of Spring templates
  - Central management and provisioning

A Dynamically Scalable Architecture for Data Intensive Trading Analysis Applications

- Most financial organizations today use Excel™ or Reporting Databases as the main trading analysis tools. These are very difficult to scale.
- The solution is to create a shared In-Memory Data Grid (IMDG) which stores the trading data in a shared pool of machines. Common data calculation and analysis run on that pool as well, leveraging the available memory and CPU resources.
- JavaSpaces is a powerful model for distributed persistence. GigaSpaces is a JavaSpaces vendor providing Enterprise features.
- Spring hides the details of the JavaSpaces model, allows effort to be focused on requirements rather than frameworks.

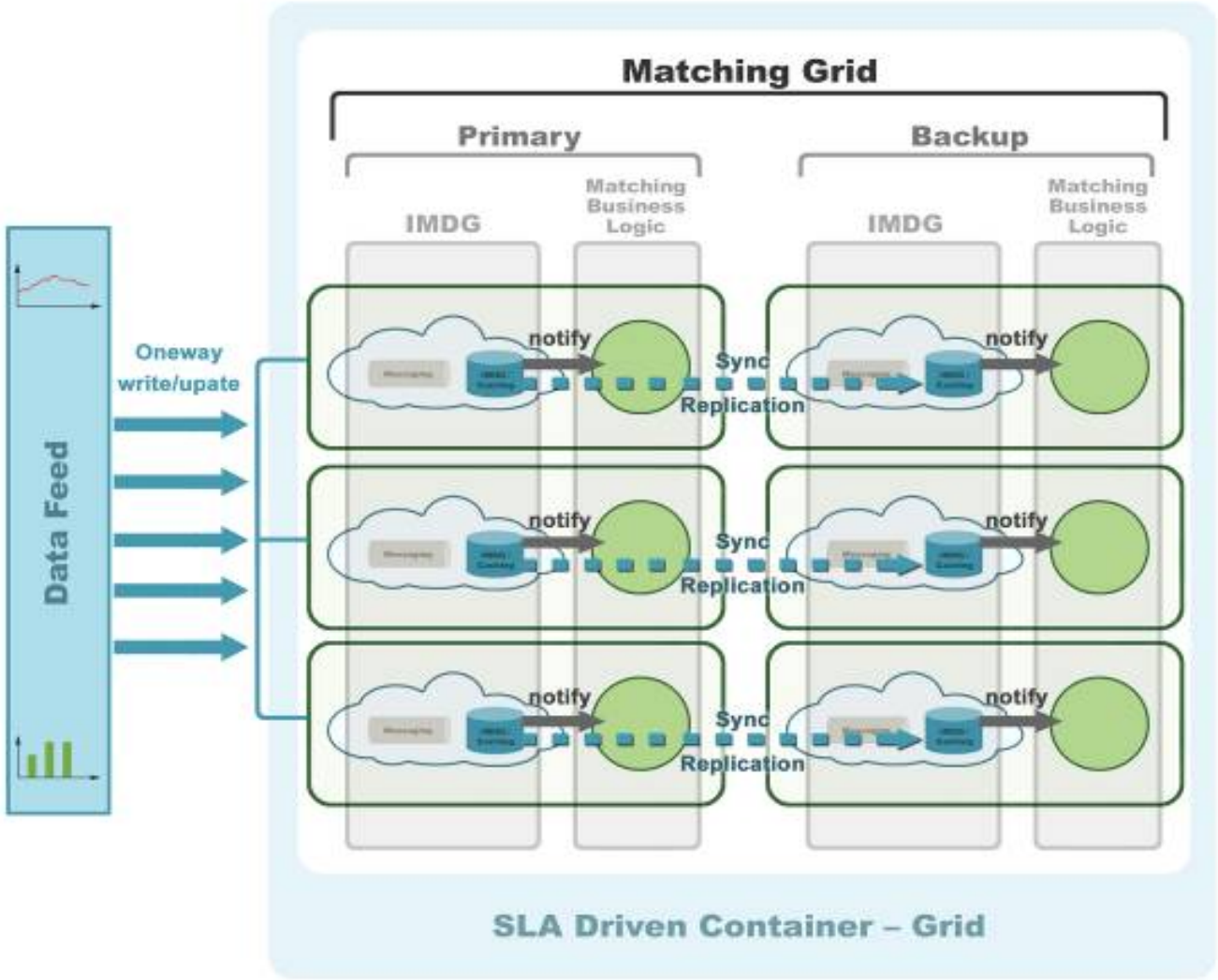


Using shared data grid for all users



Running analytics close to the data to improve performance and leverage the available resources

Reconciliation calculation

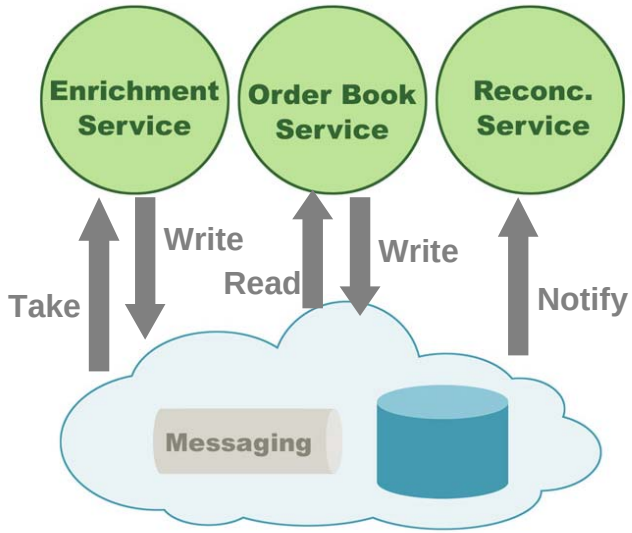


Appendix

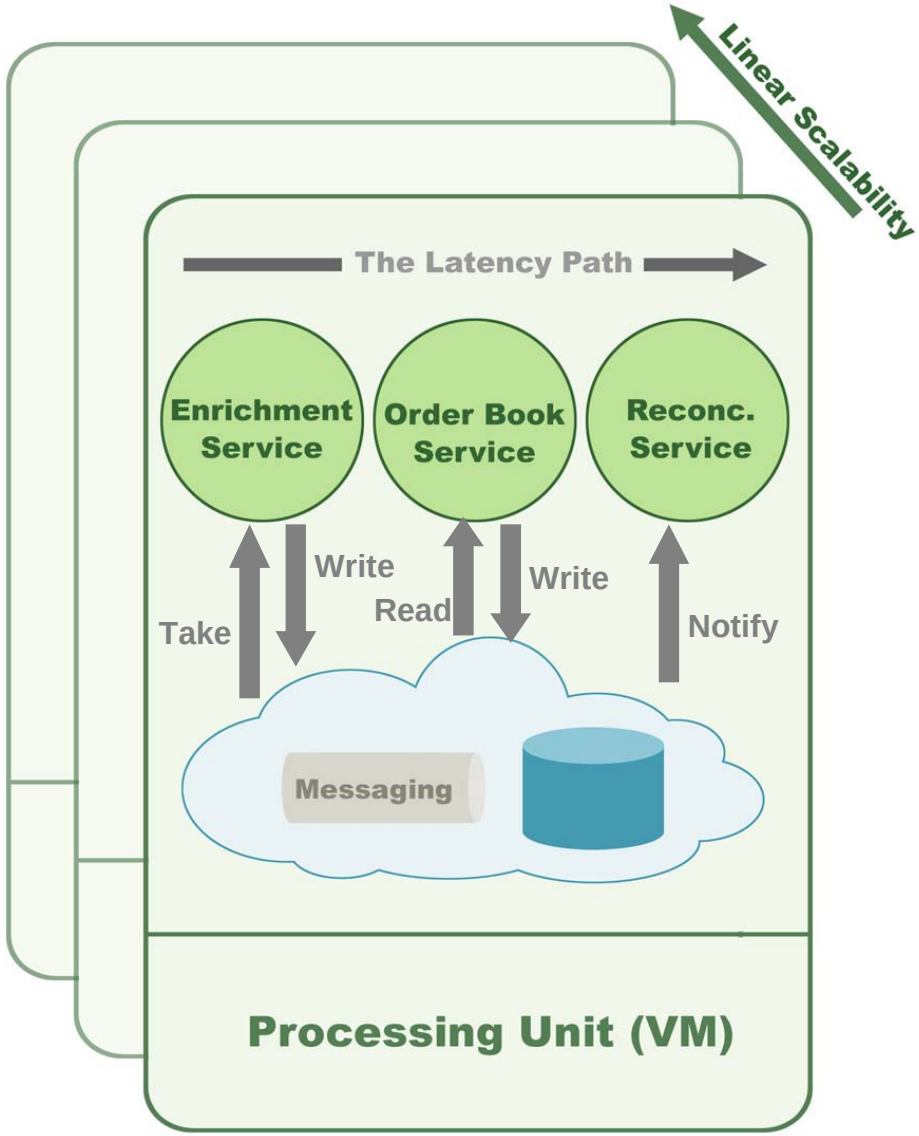
Space Based SOA using 4 Simple Verbs

Space Based SOA using 4 Simple Verbs

An Order Book Example

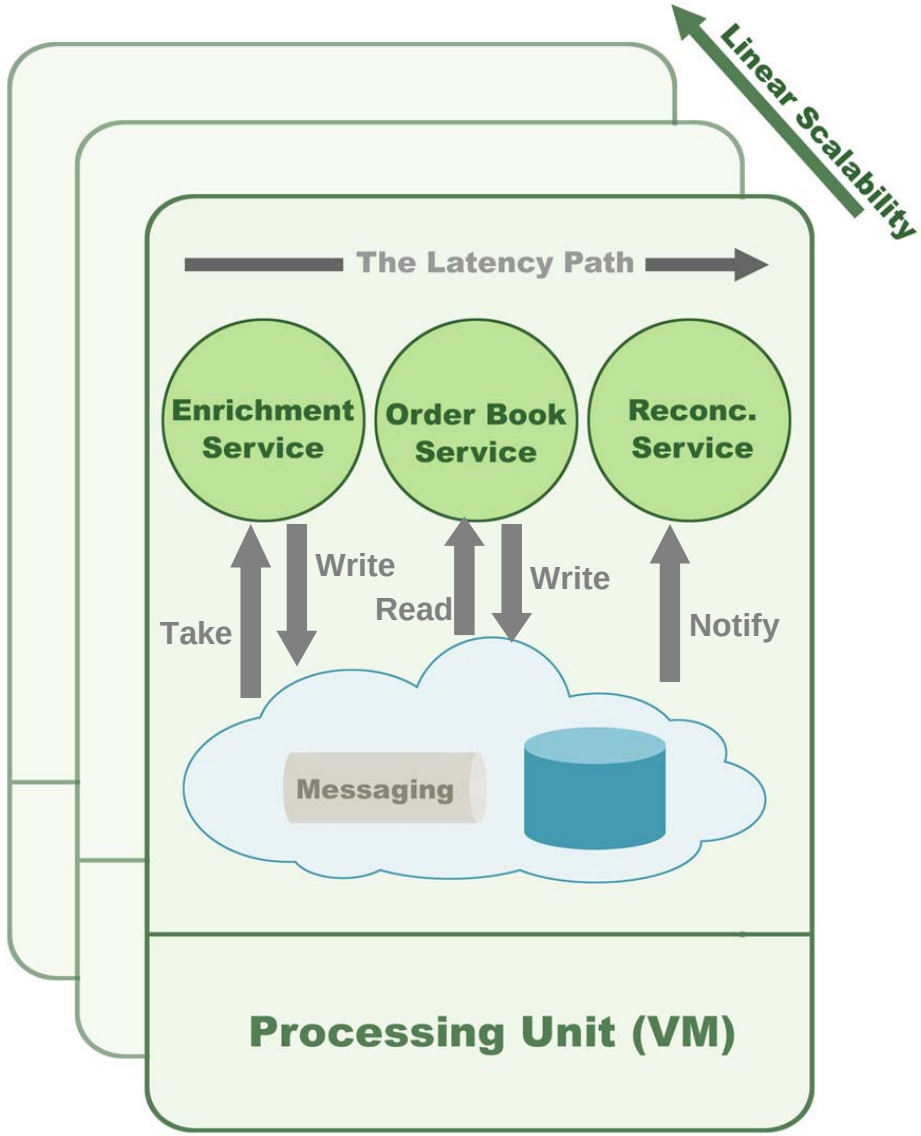


Reaching Scalability with Spaced Based SOA



Perspectives on SBA

- **The architect view**
  - Scaling = adding more processing unit
- **The developers view**
  - Testability – 90% of the functionality ca be tested on a single VM
  - Short learning curve – code it as writin to a single server
- **External applications view**
  - View the entire cluster as if it was a single server when executing queries, registering for events or writing data
- **Operation view**
  - Deployment is done through a single command
  - Single point of access for monitoring and management



SBA - Summary of Benefits

- **Simplicity**
  - Simplify the entire architecture not just a single component
- **Guaranteed linear scalability**
- **Latency**
  - Kept low due to reduced network calls and I/O
- **Throughput**
  - Maximized on a per processing unit through parallelization
  - Total throughput can be easily increased by adding more processing units
- **Reliability**
- **Single cluster used for all tiers => less moving parts**

Thank You

Download free version from our website

www.gigaspaces.com
www.gigaspacesblog.com

Email: sales@gigaspaces.com

U.S. Headquarters
GigaSpaces Technologies Inc.
1250 Broadway, Suite 2301
New York, NY 10001
Tel: 646-421-2830
Fax: 646-421-2859

U.S. West Coast Office
GigaSpaces Technologies Inc.
555 California St., 3rd Floor
San Francisco, CA 94104
Tel: 415-568-2125
Fax: 415-651-8801

Europe Office
GigaSpaces Technologies Ltd.
30 Borough High St.
London SE1 1XX, UK
Tel: +44-709-286-3096
Fax: +44-709-286-3097

International Office
GigaSpaces Technologies Ltd.
4 Maskit St., P.O. Box 4063
Herzliya, 46140, Israel
Tel: +972-9-952-6751
Fax: +972-9-956-4410