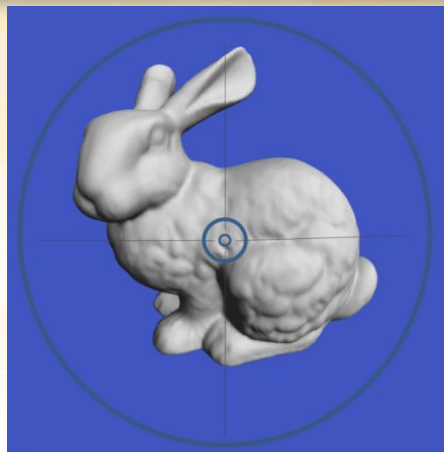


GIGASPACES

WRITE ONCE.
SCALE ANYWHERE.

High-speed SOA!

Managing High Performance Transactional Services on the Grid. —*without hops*

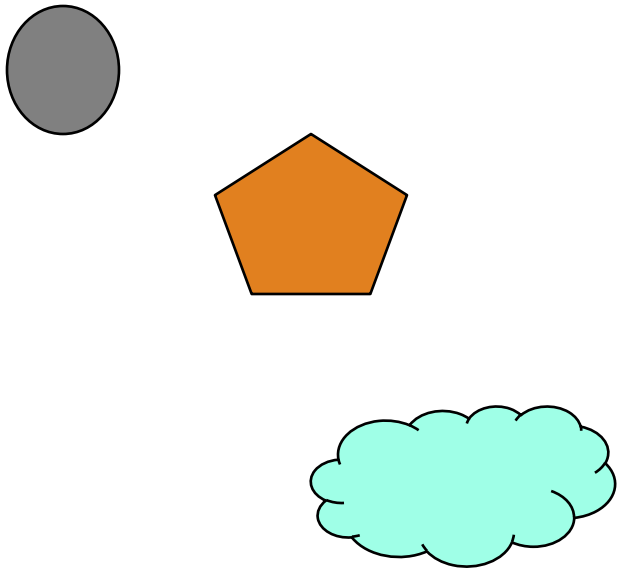
Owen Taylor

Blog:

http://www.jroller.com/owentaylor

About me. . .

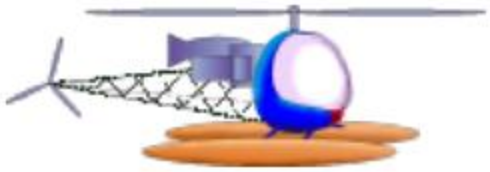
- ORBs
- Containers
- Frustration
- “The Final Frontier”

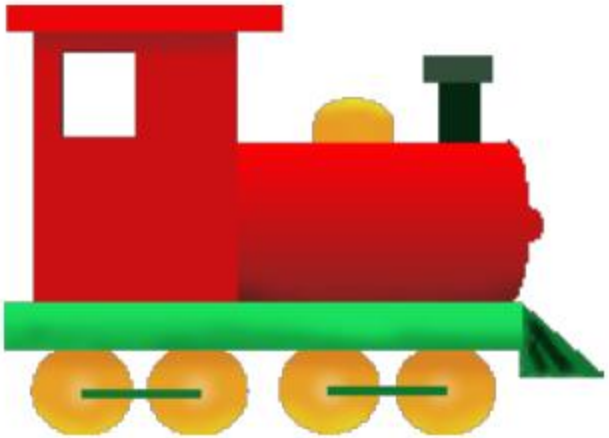


- http://jroller.com/owentaylor
- http://www.openspaces.com
- owen@gigaspaces.com

What? Me SOA?

- Microsoft landscape
- XML
- Indeterminate performance
- Galactic responsibility
- Trains or Helicopters?



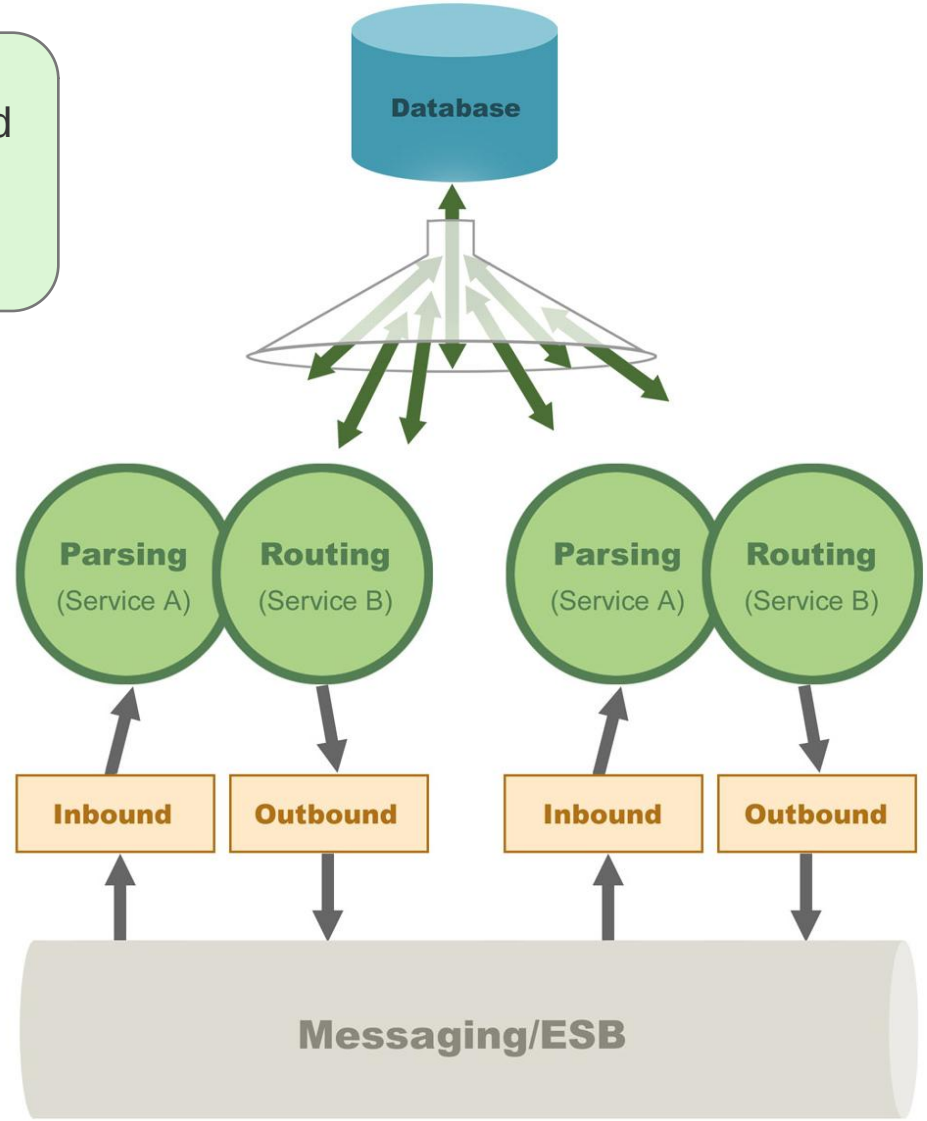


Typical SOA model based on ESB

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

Services can be Java, C++, .Net

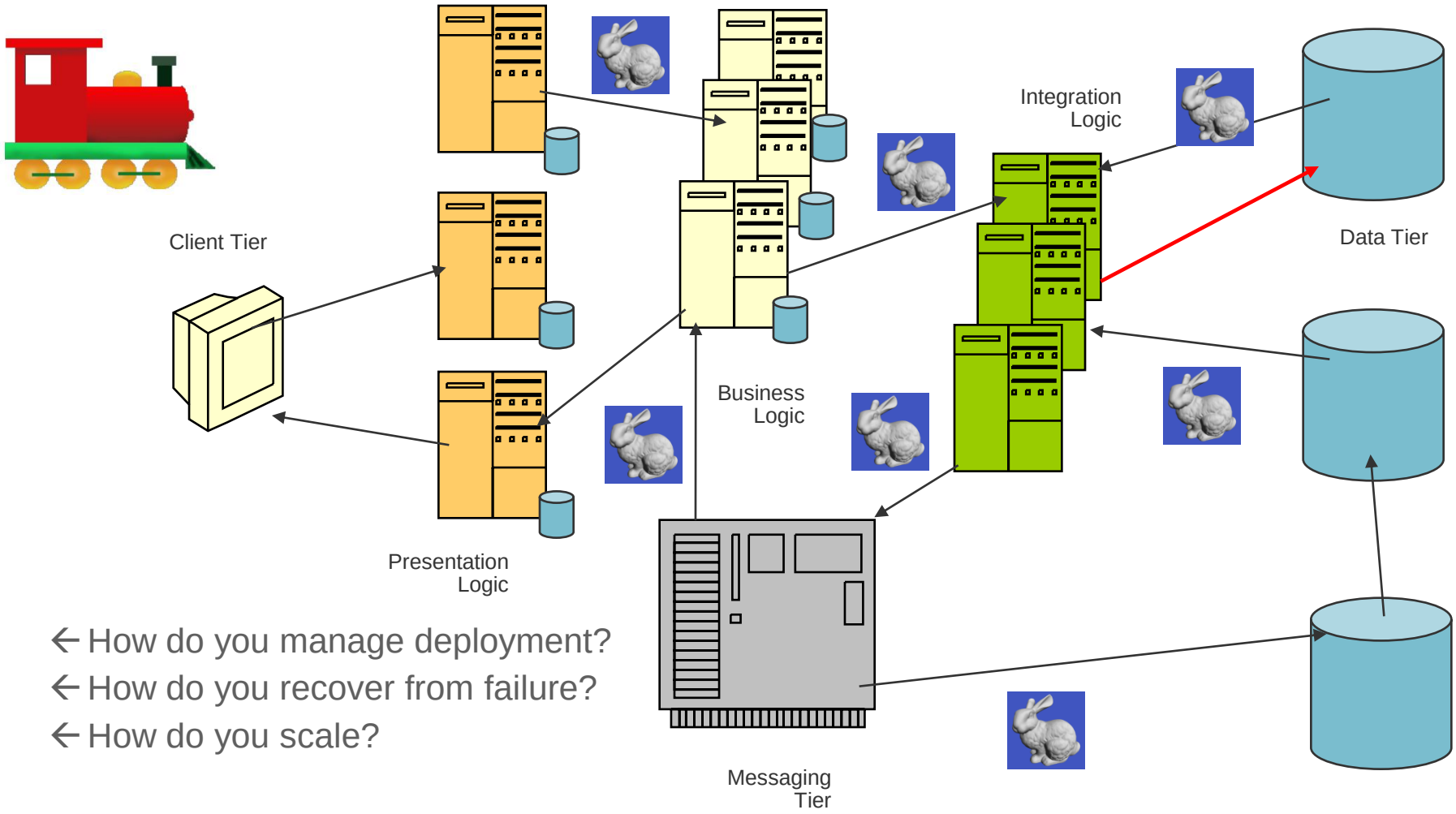
Content-Based Routing



The Teir_ful Reality: Train-wreck

← The latency path traverses multiple machines... →

← The CPU(s) often sits idle waiting on I/O [waste OF $$$$] →



← How do you manage deployment?

← How do you recover from failure?

← How do you scale?

The Journey *Step 1*

- Caching :: read-mostly
  - Small impact on architecture and code
  - Collocates frequently accessed information and business logic
  - **When information is non-volatile:**
    - Greatly increases throughput
    - Greatly reduces latency
  - Still requires
    - Remote interaction with events
    - Remote writes /checkpoints

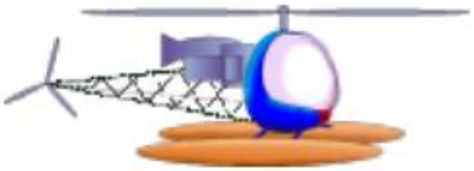
The Journey…*Step 2*

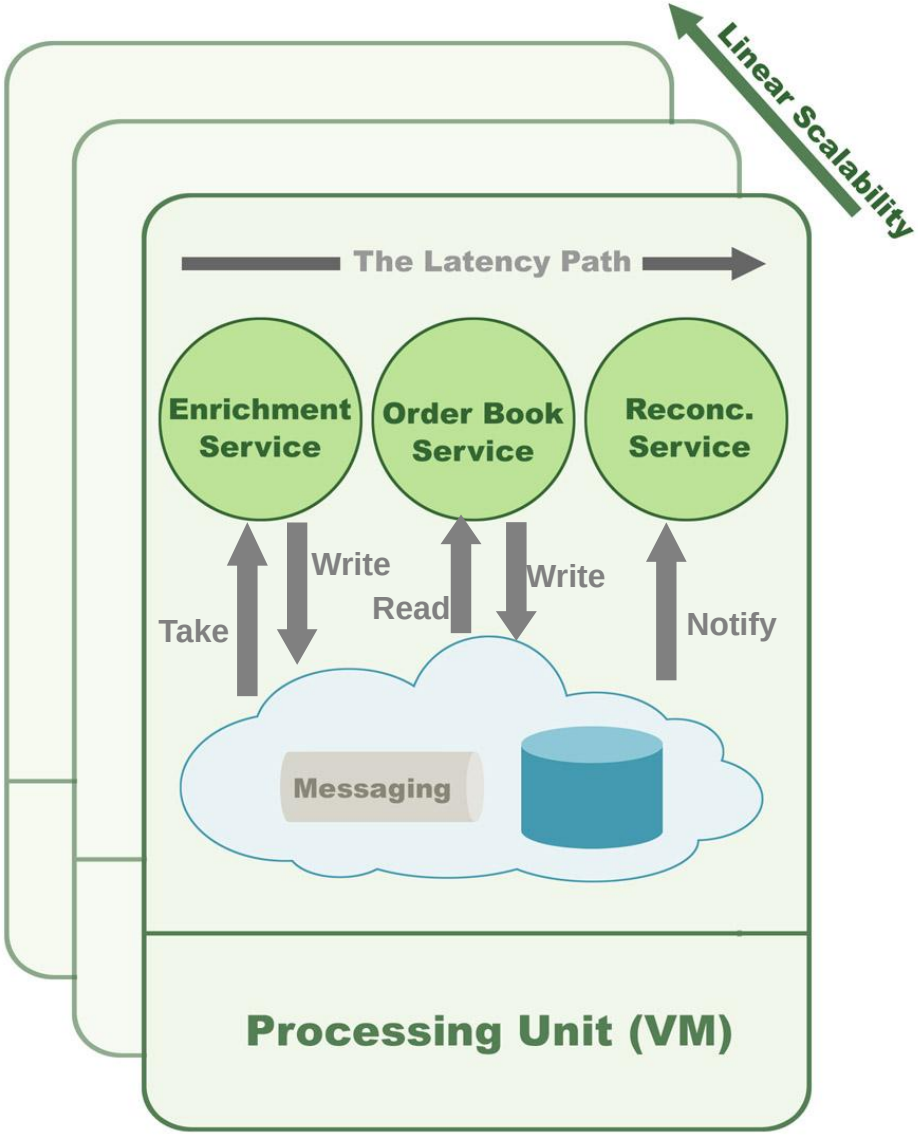
- DataGrid :: read-write/process in replicated memory
  - Proportional impact on architecture and code to reap the benefits:
    - Must develop code with awareness of scatter-gather/map-reduce
  - Collocates most information and business logic
  - Partitions/Spreads the workload across multiple servers
  - **Works wonders:**
    - Greatly increases throughput
    - Greatly reduces latency
  - Still requires remote event service :: ***other tiers tend to remain***
  - Requires particular data-grid-aware programming

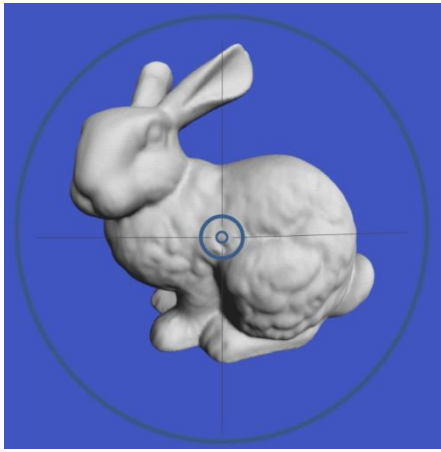
The Journey…*Step 3*

- Virtual AppServer :: read-write/process in managed service environment
  - Significant impact on architecture
    - Must design system with awareness of scatter-gather/map-reduce
  - Support for Spring Framework, JDBC, JMS, MAP API lessens impact on code
  - Collocates most information + business logic **+ events**
  - Partitions/Spreads the workload across multiple servers
  - **Works Wonders:**
    - Greatly increases throughput
    - **Produces smallest possible latency**
    - **Provides intelligent self-healing Service Virtualization Framework**

Voila: Helicopters



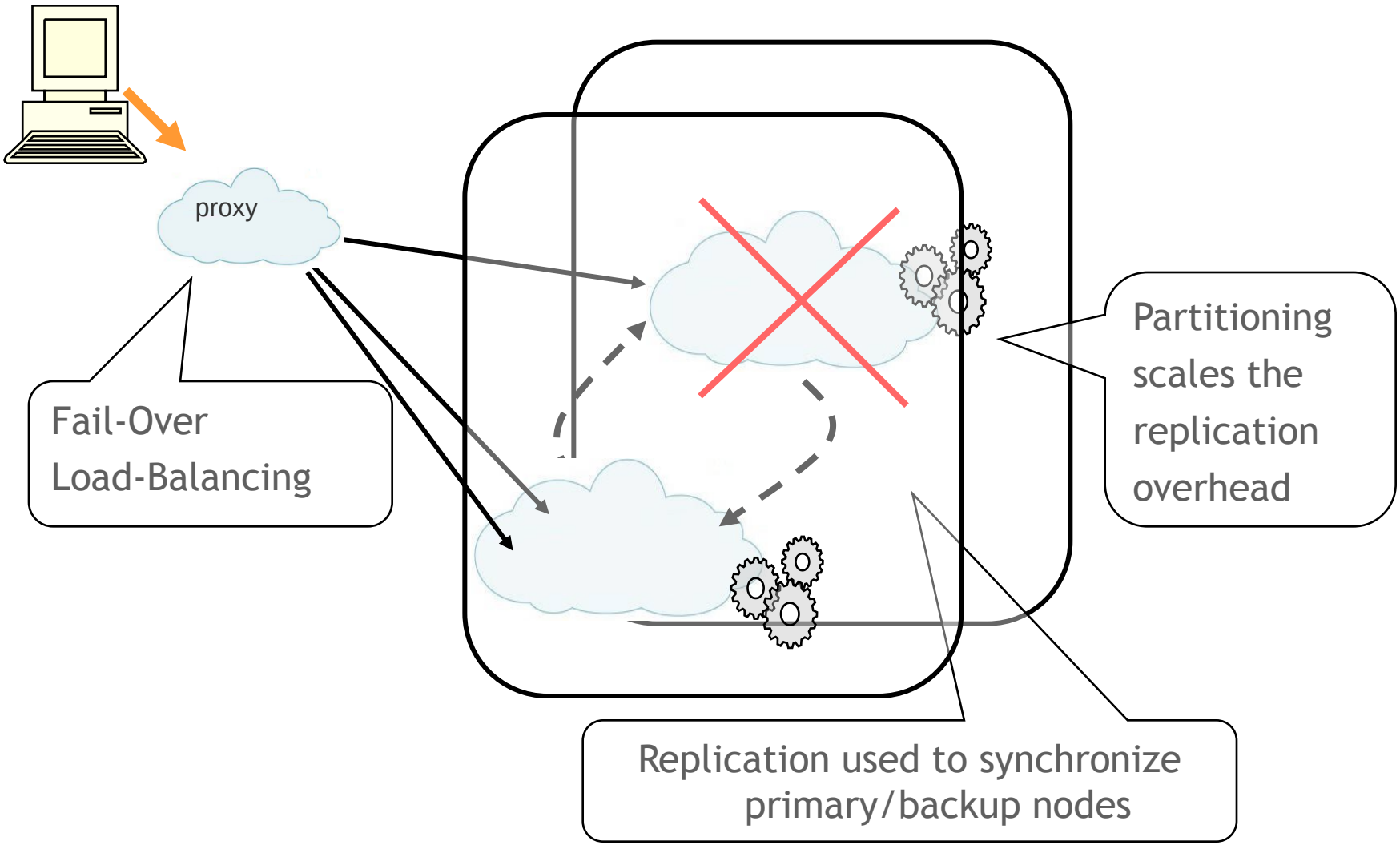




Where the SB-SOA Rocks:

- Internal service implementation – exposed by arbitrary ‘Portal’
- No XML to communicate – only for configuration
- SLA- driven behaviors [scale-out/up on demand]
- Fault-tolerance/self-healing
- Highly decoupled [autonomous]
- Amazon EC2-ready

The GigaSpacesXAP Service Framework...

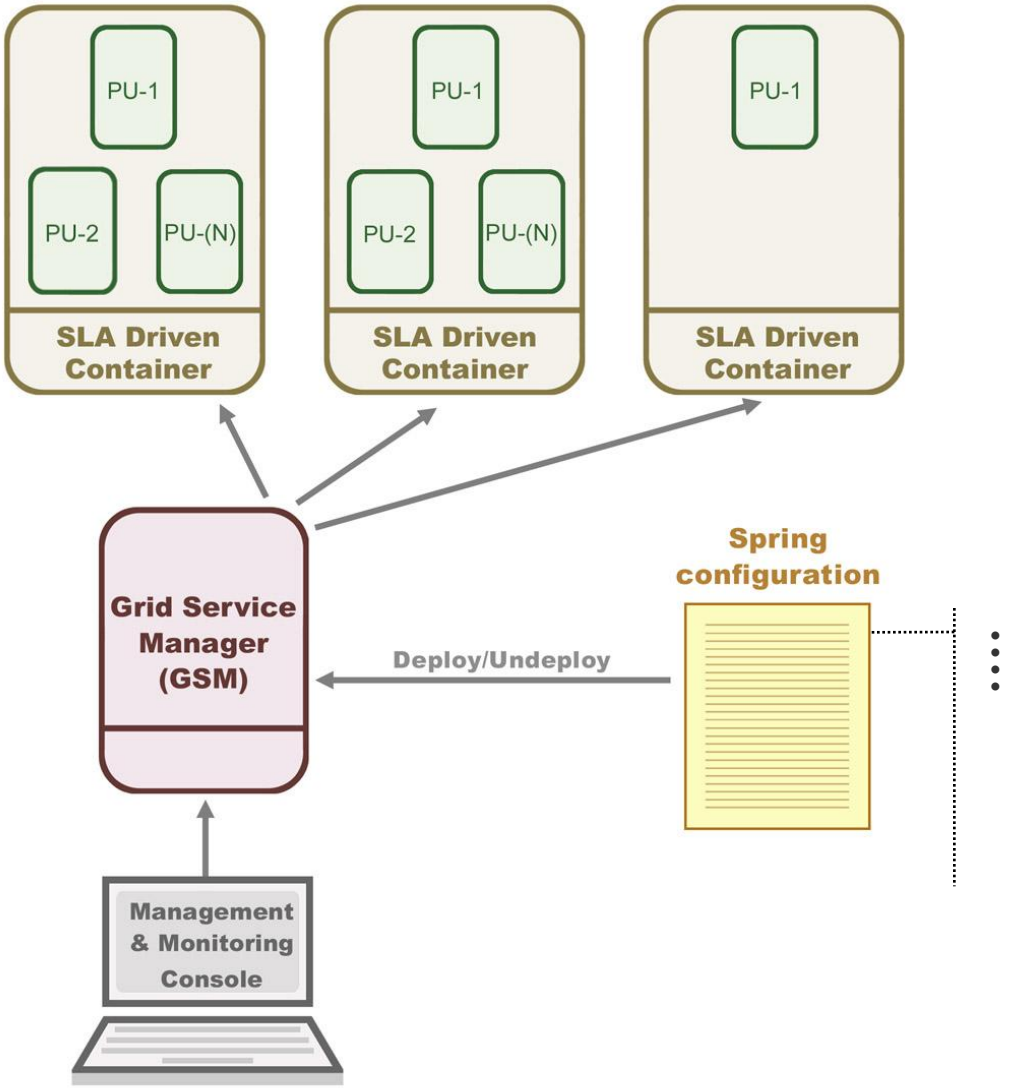


GigaSpacesXAP Business Services:

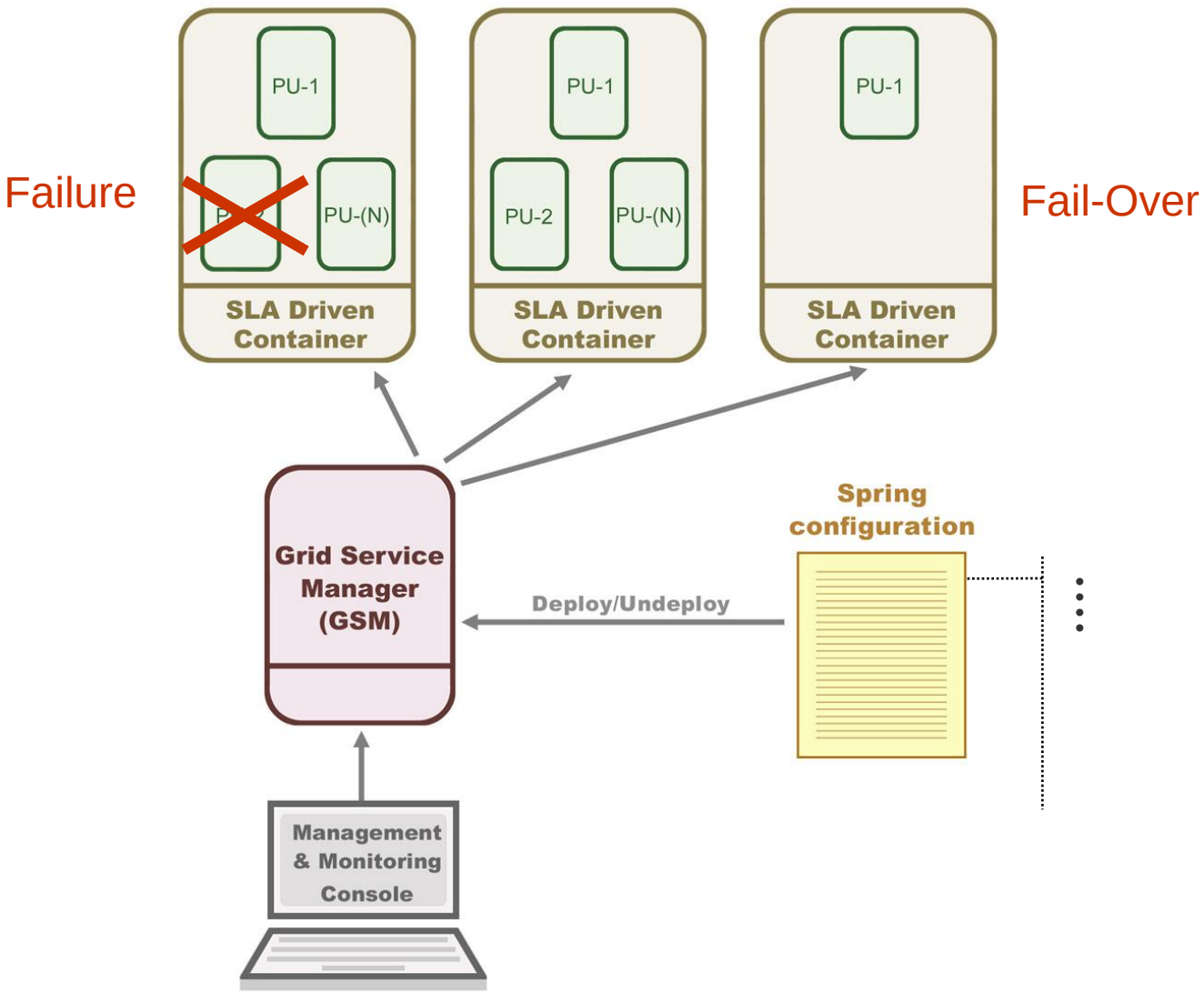
- Interact with each other through the space
- Are co-located with data/events for faster results
- Are deployed and managed in an adaptive and fail-safe environment
- Are implemented as Spring Beans

```
public class MessageProcessor {
   @SpaceDataEvent
   public Message messageProcessed(Message obj){
     obj.setContent(obj.getContent()+" processed by: "+
     this.getClass().getSimpleName()+"  at: "+
     new Timestamp(System.currentTimeMillis()));
     obj.setProcessed(true);
     System.out.println(this.getClass().getSimpleName()+
     " processed message #"+obj.getId());
     return obj;
   }
 }
```

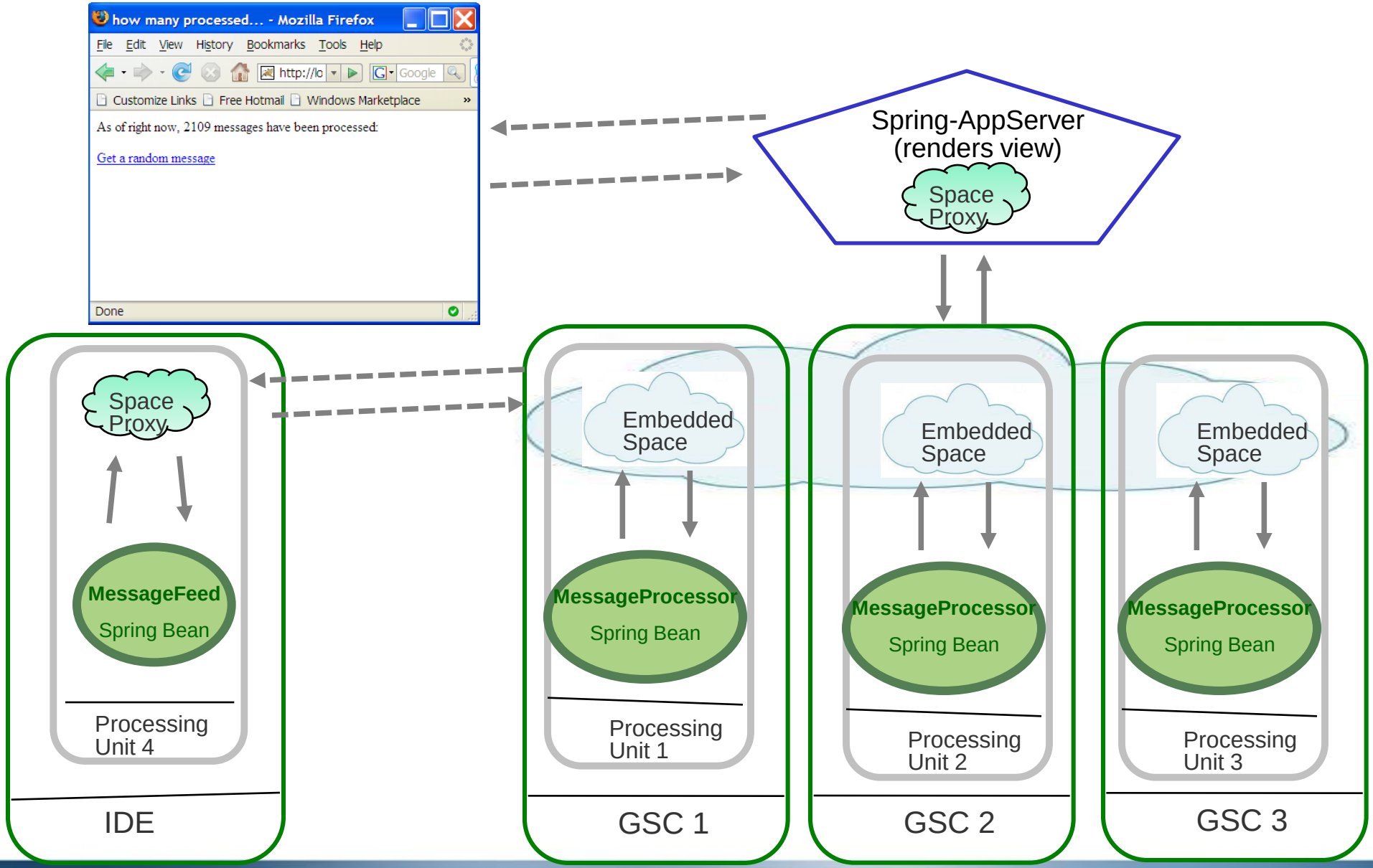
SLA Driven deployment



Continuous High Availability

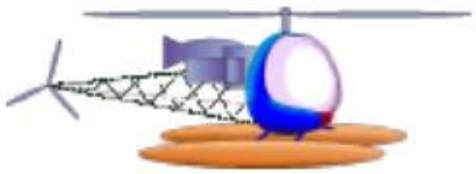


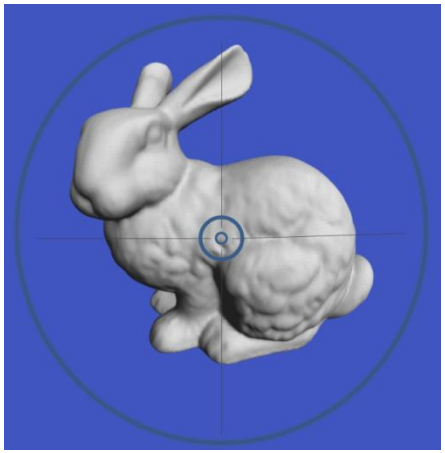
Service Reliability Self-Healing/Integration Demo

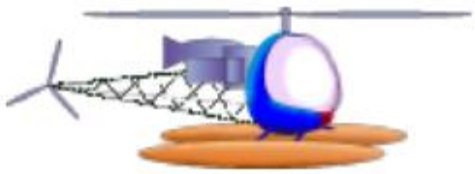


TPC: Transparent Partitioning & Colocation

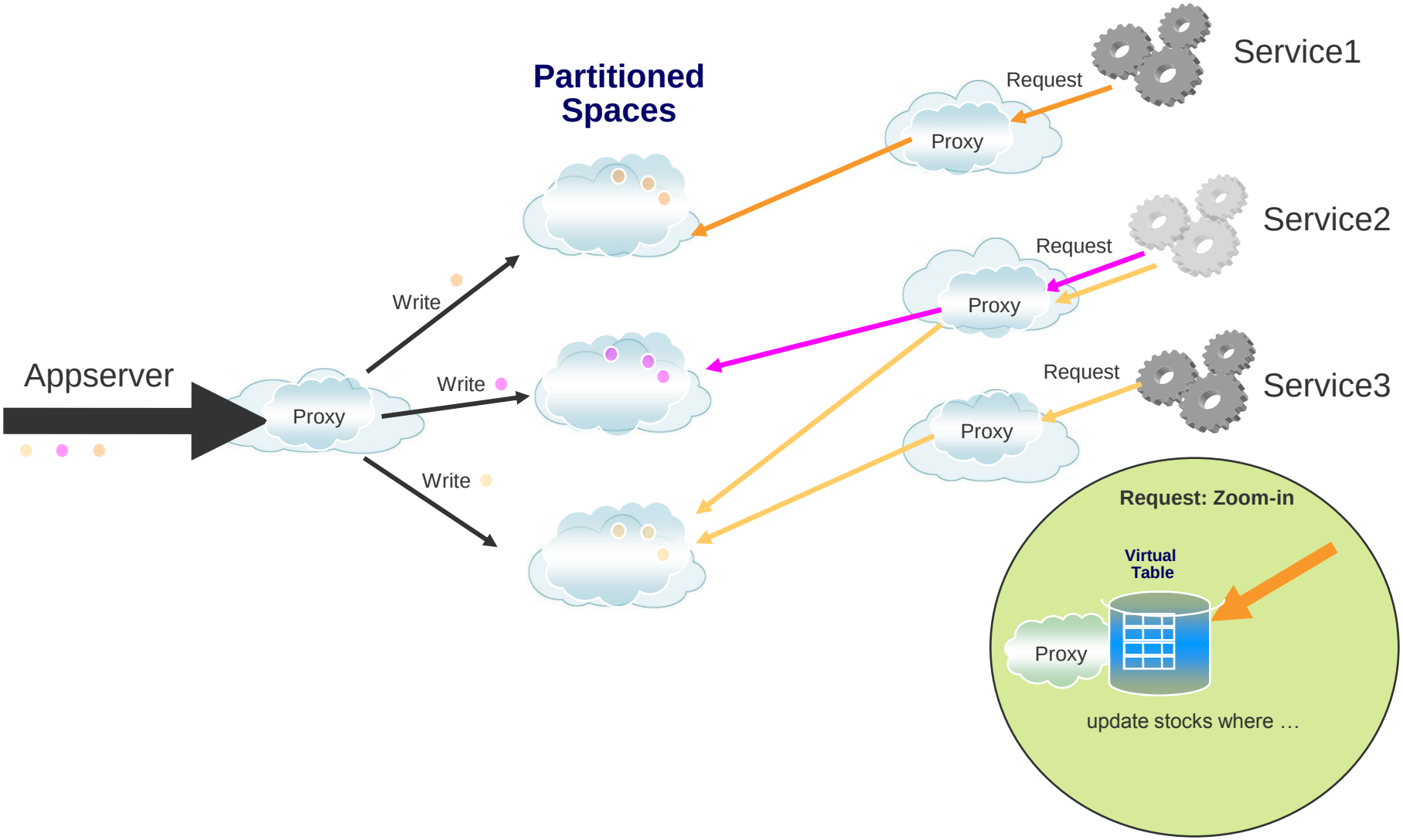
- The next wave of applications will scale in this manner
  - Because Gartner says so ☺
- Many already scale one or two tiers in this way
  - Partitioned Databases with Triggers
  - Content-Based Routing though Messaging with coupled consumers
  - All-in one web applications using caching (IMDG)
- Space-Based Architecture offers an implementation of TPC using Spaces
  - GigaSpaces enables this architecture for .NET, C++, and Java



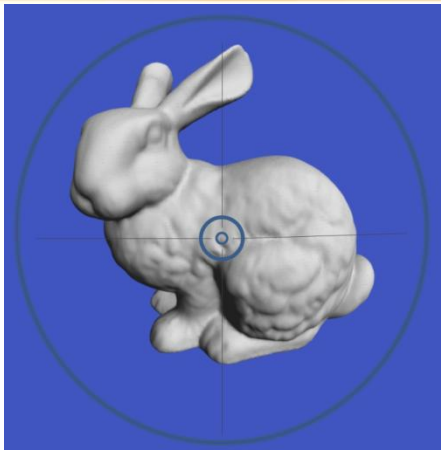


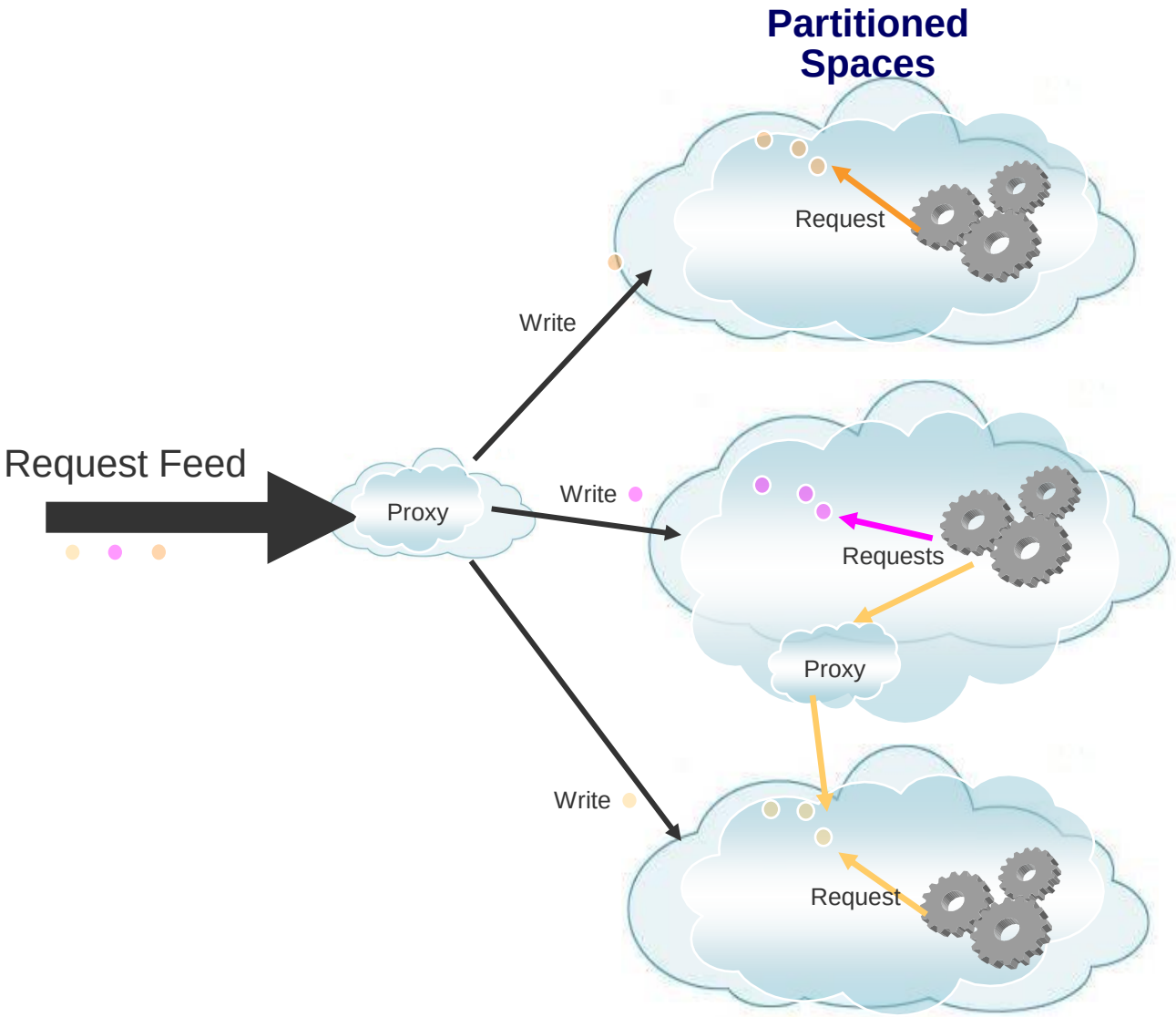


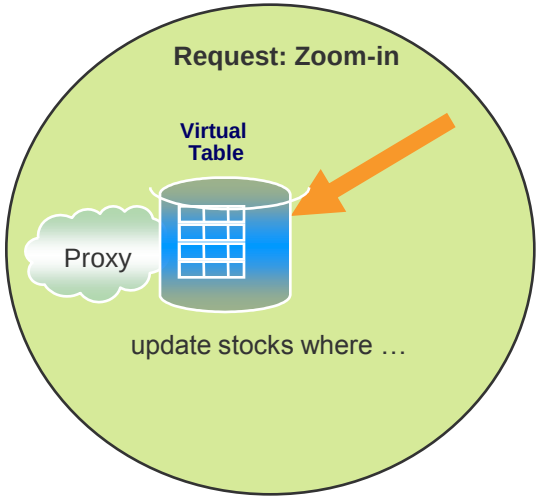
Parallel Processing: Divide and Scale out



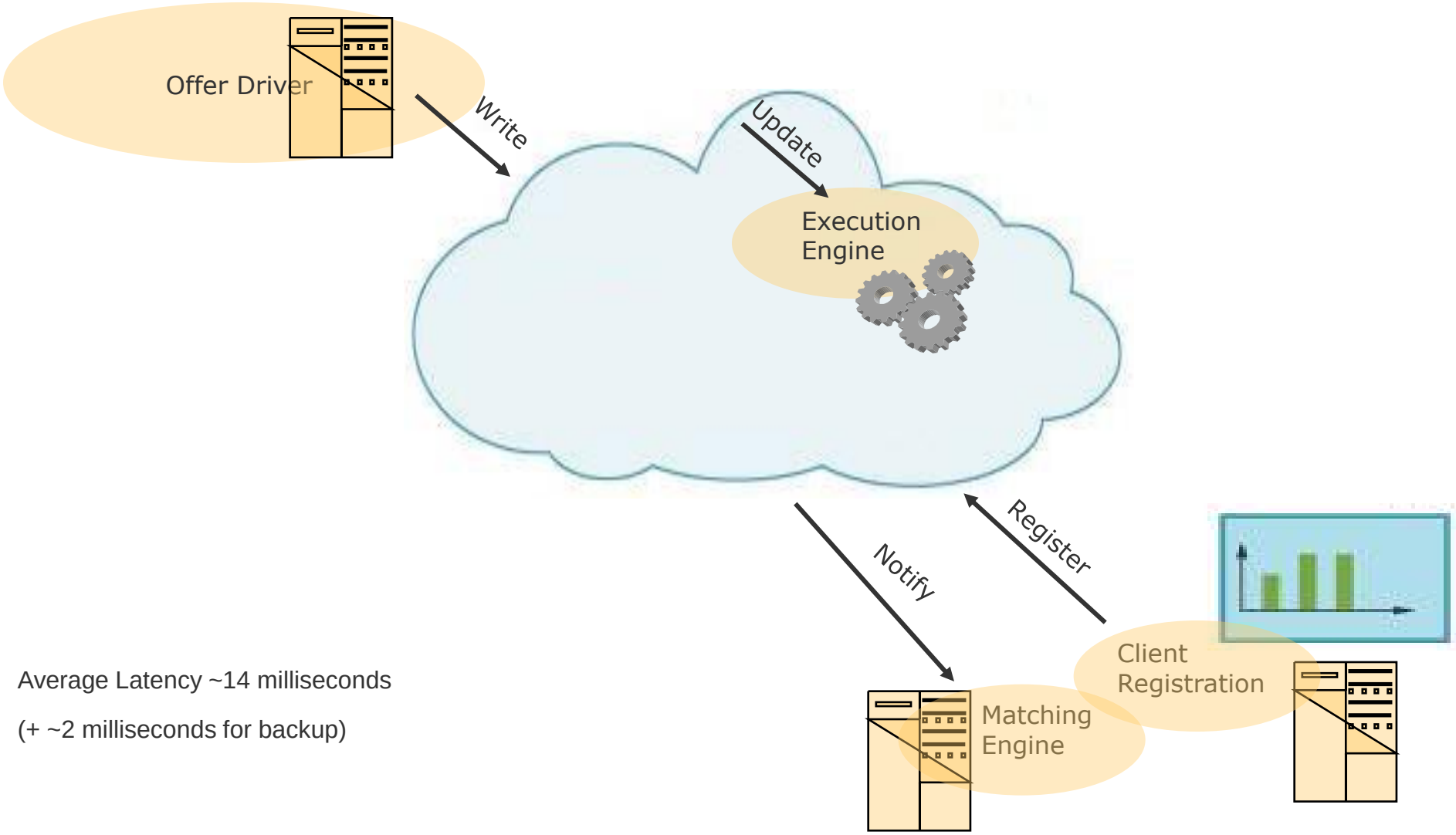
Colocation: Dramatically Reduce Latency







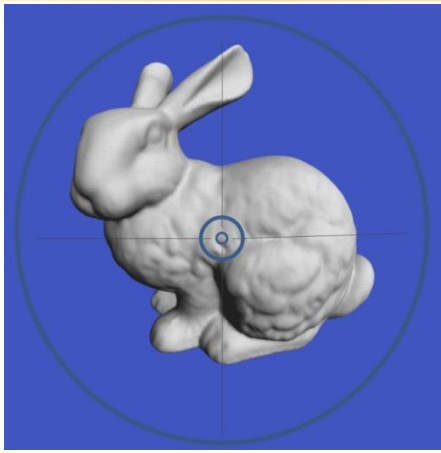
Example: Algorithmic Trading



Average Latency ~14 milliseconds

(+ ~2 milliseconds for backup)

Reduced Latency Due To Colocation

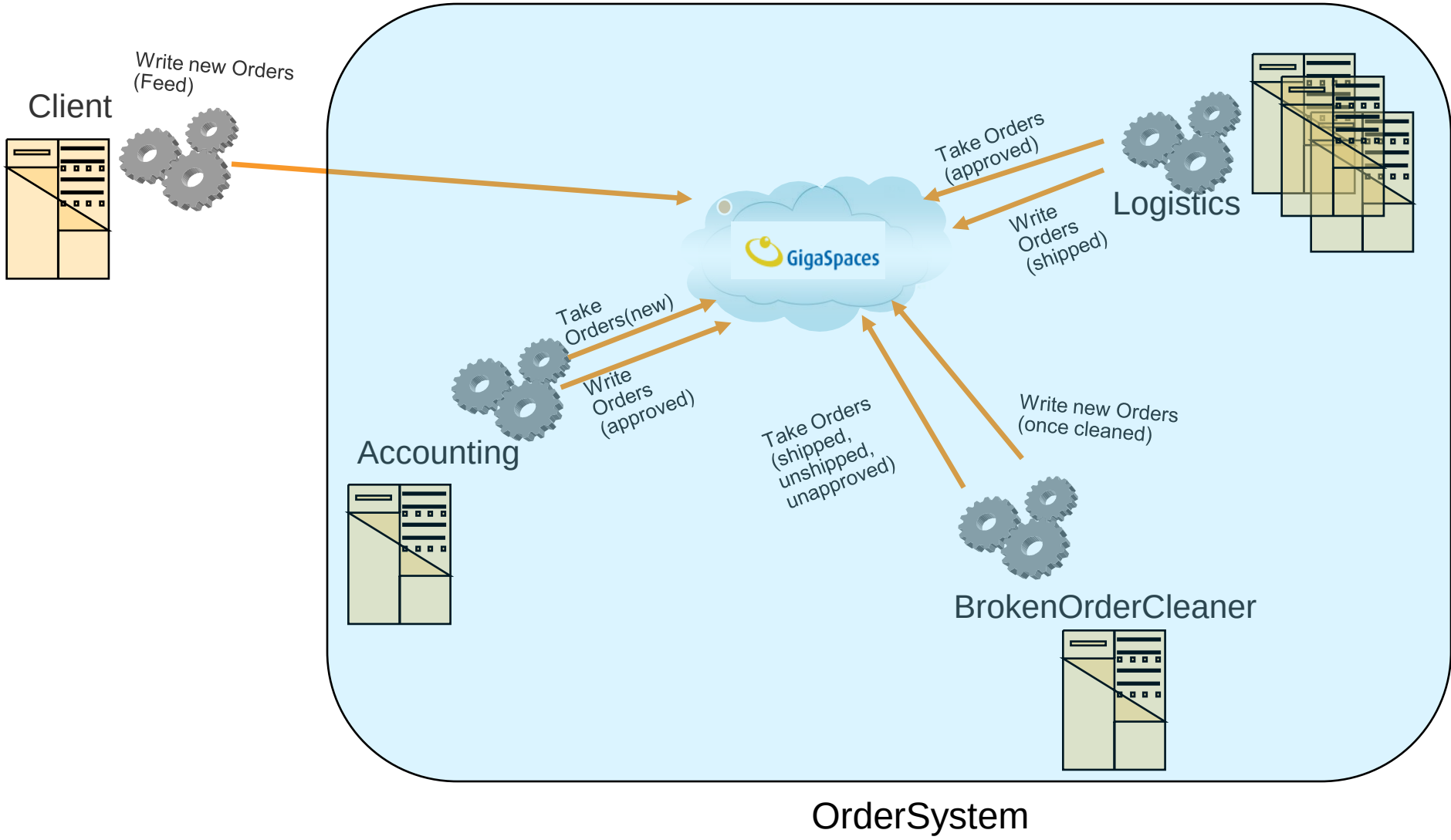


Average Latency 0.600 milliseconds

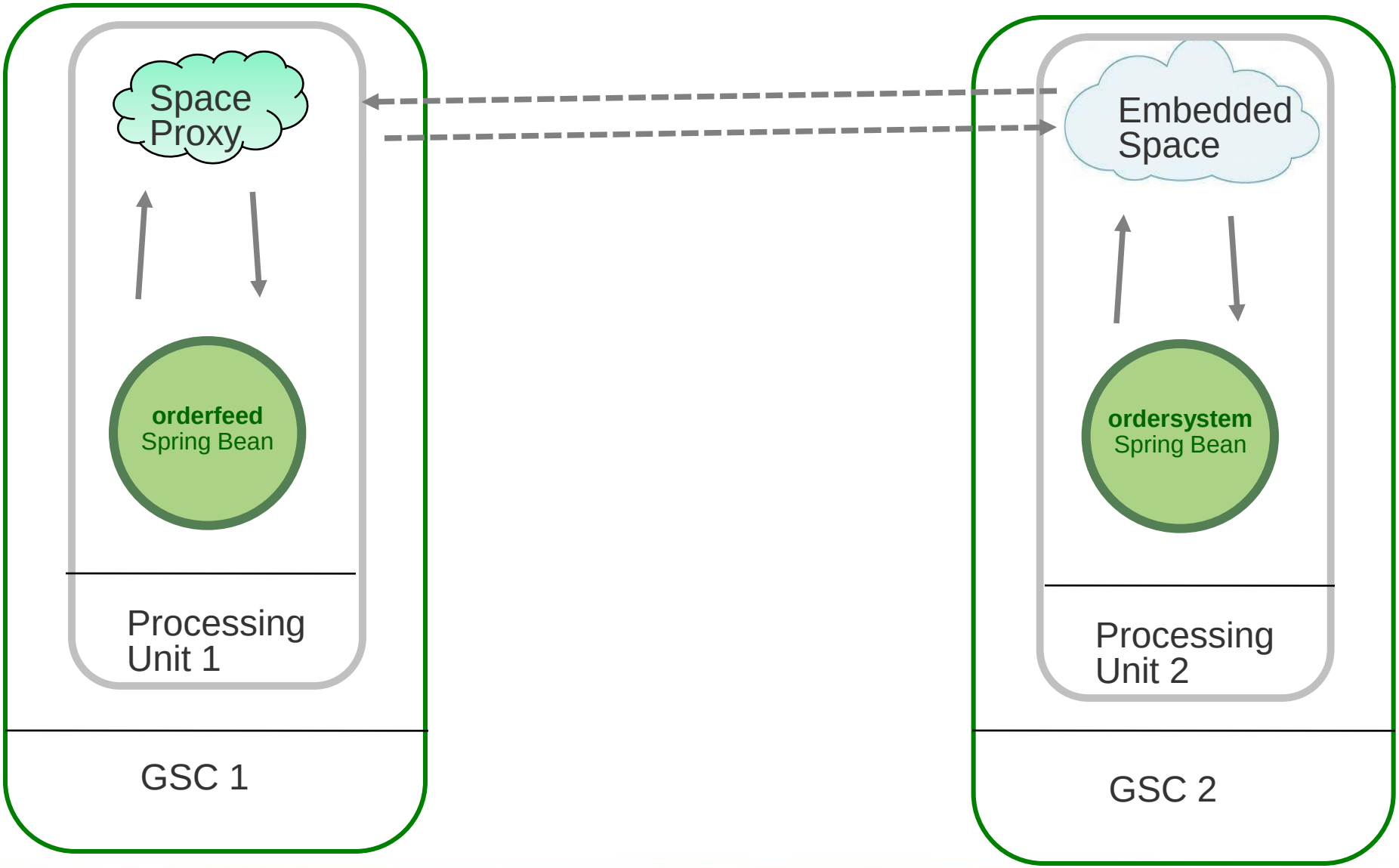
[600 microseconds]

(+ ~2 milliseconds for backup)

Linear Scaling Demo: Logical Workflow

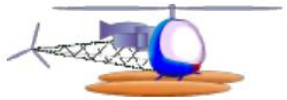


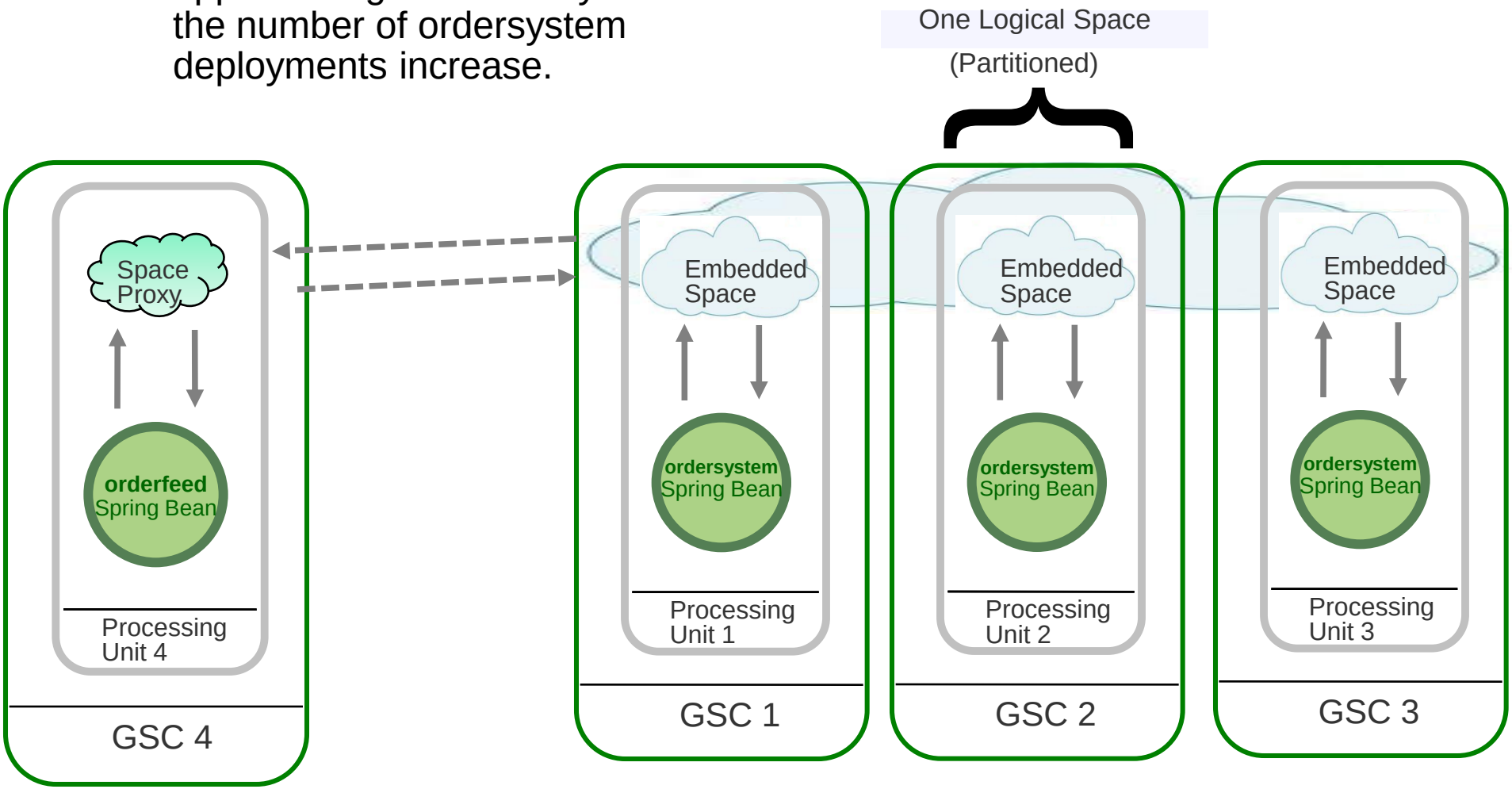
Linearly Scalable OrderProcessing: Deploy Once:



Linearly Scalable OrderProcessing: Deploy Again:

The throughput of the application grows linearly as the number of ordersystem deployments increase.



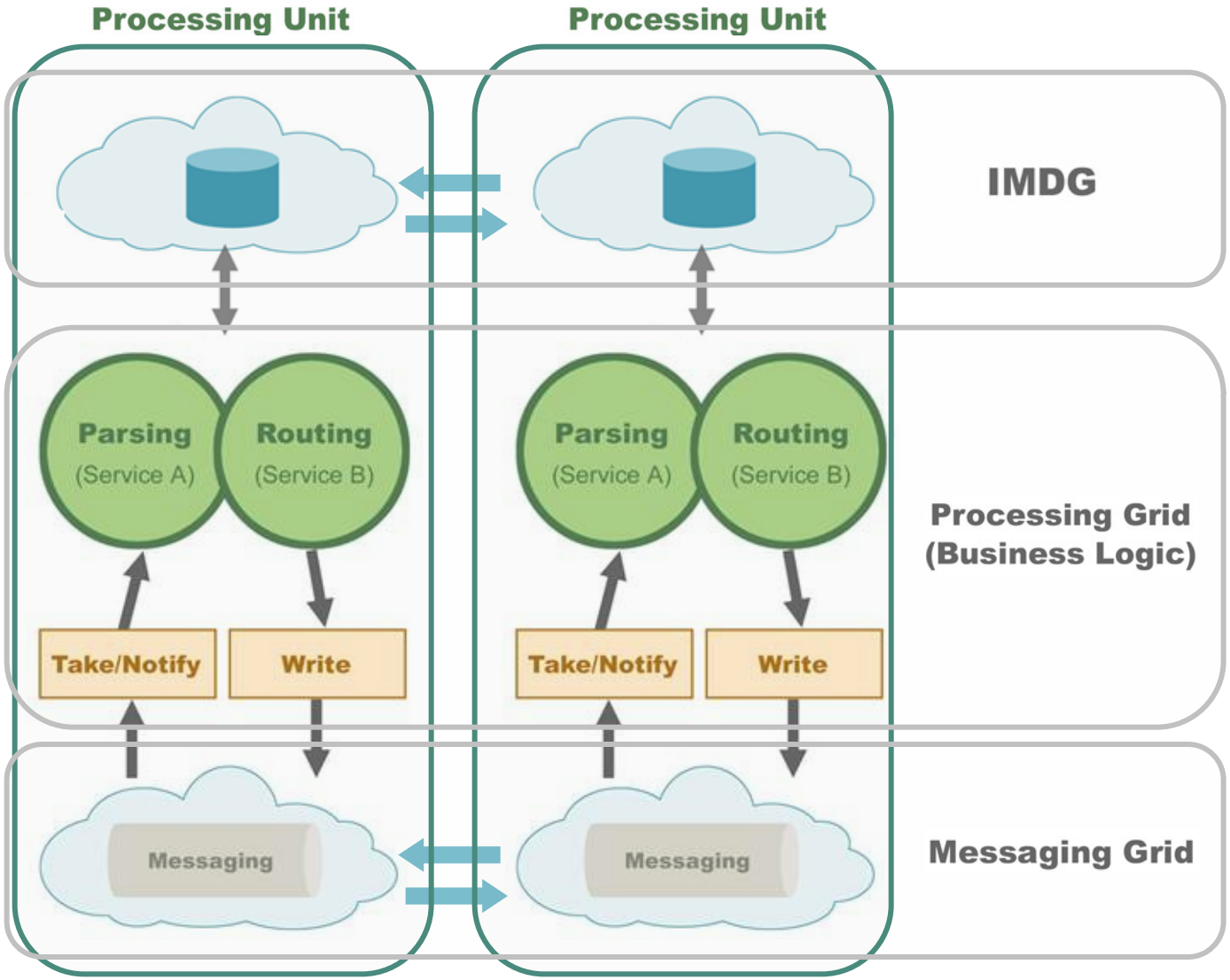


SB-SOA = Real-time SOA for Stateful Applications

Shared state to enable stateful services

Services can be Java, C++, .Net

Content-Based Routing



Questions Please?

Thank You!

http://www.jroller.com/owentaylor

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