Java is a trademark of Sun Microsystems, Inc.

JavaOne℠

Alternative to Google Application Engine for Java™ Technology-Based Applications

Nati Shalom
CTO GigaSpaces
Natishalom.typepad.com
Twitter.com/natishalom

Francis de la Cruz
Manager,

Argyn Kuketayev
Senior Consultant,
Primatics Financial

About GigaSpaces

A middleware platform enabling applications to run a distributed cluster as if it was a single machine

2,000+ Deployments

Among Top 50 Cloud Vendors

100+ Direct Customers

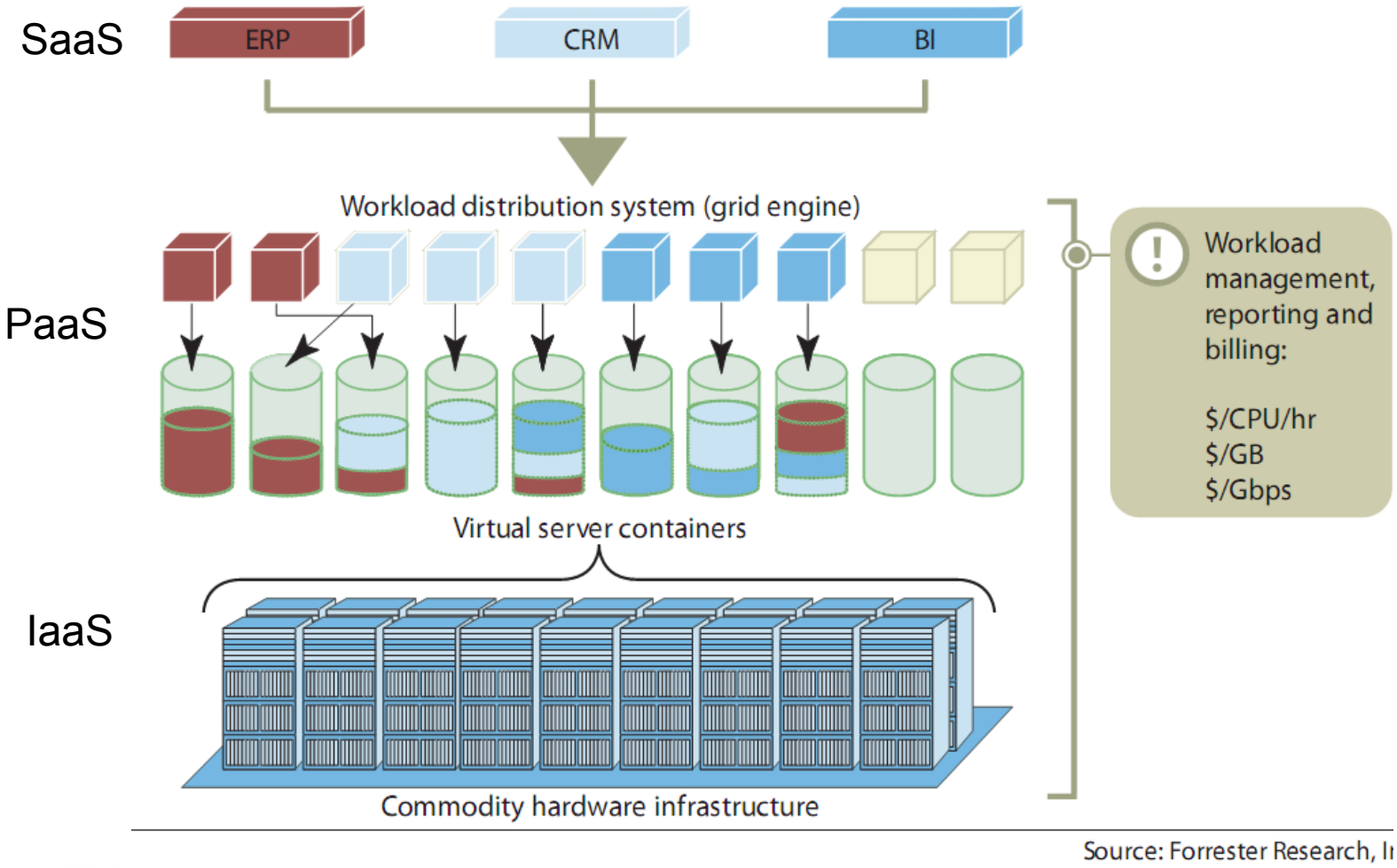
Agenda

- Introduction to Cloud Computing
- AWS vs GAE
- Challenges of Enterprise Applications
- PaaS on EC2
- Case study – Primatics Risk Management as a Service
- Summary & References

PetClinic in the Clouds: Scaling a Classic Enterprise Application
Wednesday 1:35 PM - 3:15 PM LAB-5564BYOL Hall E 132

Free drinks Webtide & GigaSpaces party – Tuesday 8 PM
SOMA 201 3rd St (gigaspaces.com/javaone)

Introduction to cloud computing



Source: Forrester Research, I

Agenda

- Introduction to Cloud Computing
- AWS vs GAE
- Challenges of Enterprise Applications
- PaaS on EC2
- Case study – Primatics Risk Management as a Service
- Summary & References

AWS vs GAE

Source:Zdnet

AWS vs GAE (1/2)

- AWS
  - Very flexible, lower-level offering (closer to hardware) = more possibilities, higher performing
  - Runs platform you provide (machine images)
  - Supports all major web languages
  - Industry-standard services (move off AWS easily)
  - Require much more work, longer time-to-market
    - Deployment scripts, configuring images, etc.
  - Various libraries and GUI plug-ins make AWS do help

AWS vs GAE (2/2)

- GAE
  - Easy to use, free (but limited)
    - fees coming soon
  - Very tightly controlled – no installation/config of open source software
  - Proprietary environment = hard to move away from
  - Further support may be limited (ex: Ruby on Rails tightly coupled with relational DBs)

GAE Java Limitations

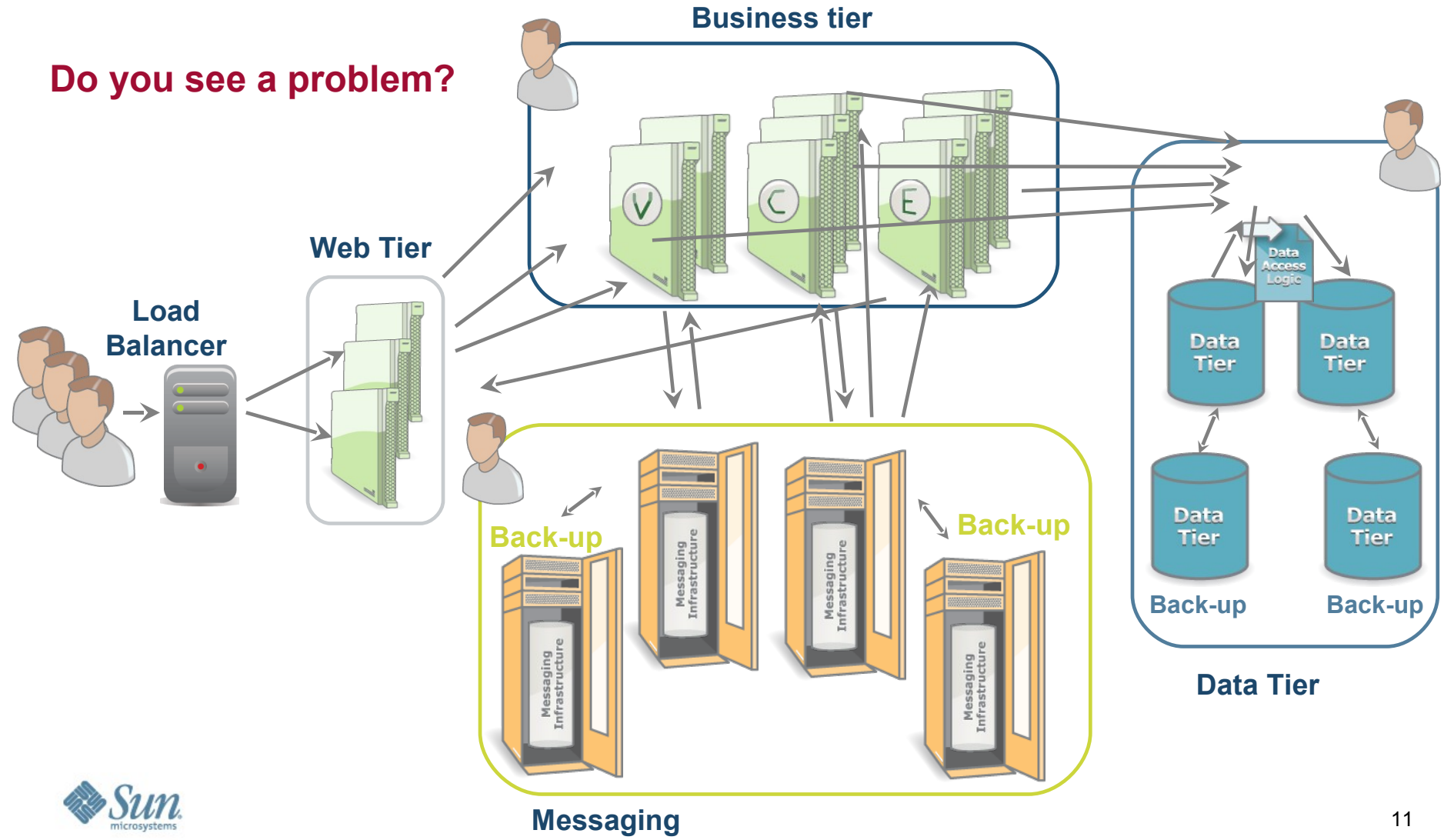
- Once a request is sent to the client no further processing can be done.
- A request will be terminated if it has taken around 30 seconds
- Sandbox restrictions:
  - Applications can not write to the file system and must use the App Engine datastore instead.
  - Applications may not open sockets
  - Applications can not create their own threads or use related utilities such as timer.
  - java.lang.System has been restricted as follows:
  - exit(), gc(), runFinalization(), and runFinalizersOnExit() do nothing.
  - JNI access is not allowed.
- Limited JPA support (many relationships, Aggregation queries, Join" queries,..)

Source: InfoQ

Agenda

- Introduction to Cloud Computing
- AWS vs GAE
- Challenges of Enterprise Applications
- PaaS on EC2
- Case study – Primatics Risk Management as a Service
- Summary & References

Typical Enterprise Application



Enterprise application challenges

- Adding additional resources dynamically
- No out-of-the-box infrastructure for J2EE
- Dealing with a lack of persistence
- Dealing with distributed programming models
- Having to think about the whole stack. Not just the code.
- Using Memory Data Grids and Caching considerations
- Messaging
- Understanding configuration management tools
- Pricing and licensing

Source: Cloud Mailing List

The solution

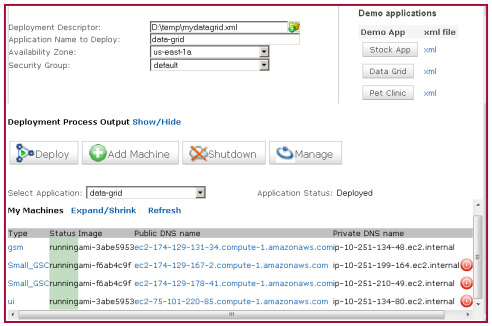
- Build a Generic PaaS ontop of AWS
  - Get the best out of the two worlds
    - Flexibility and performance of AWS
    - Simplicity and ease of use of PaaS
- JEE as first class citizen
  - Minimize lock-in
- Use middleware virtualization
  - Abstract the application code from the infrastructure
  - Enable end to end elasticity

Agenda

- Introduction to Cloud Computing
- AWS vs GAE
- Challenges of Enterprise Applications
- PaaS on EC2
- Case study – Primatics Risk Management as a Service
- Summary & References

PaaS on top of AWS – Architecture view

MT Application Provisioning



2)Deploy

Application
Deployment
Configuration

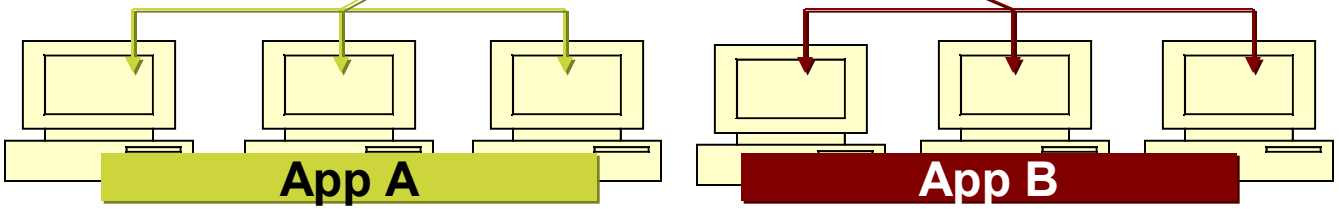
3)Manage

1)Install

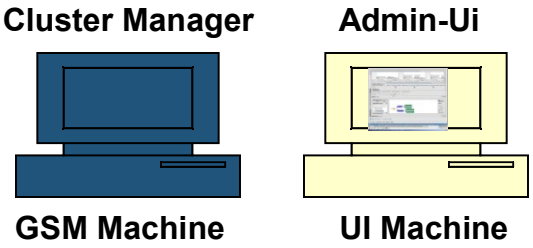
Provision

IaaS Provider (EC2, GoGrid, Sun, VMWhere, Citrix,..)

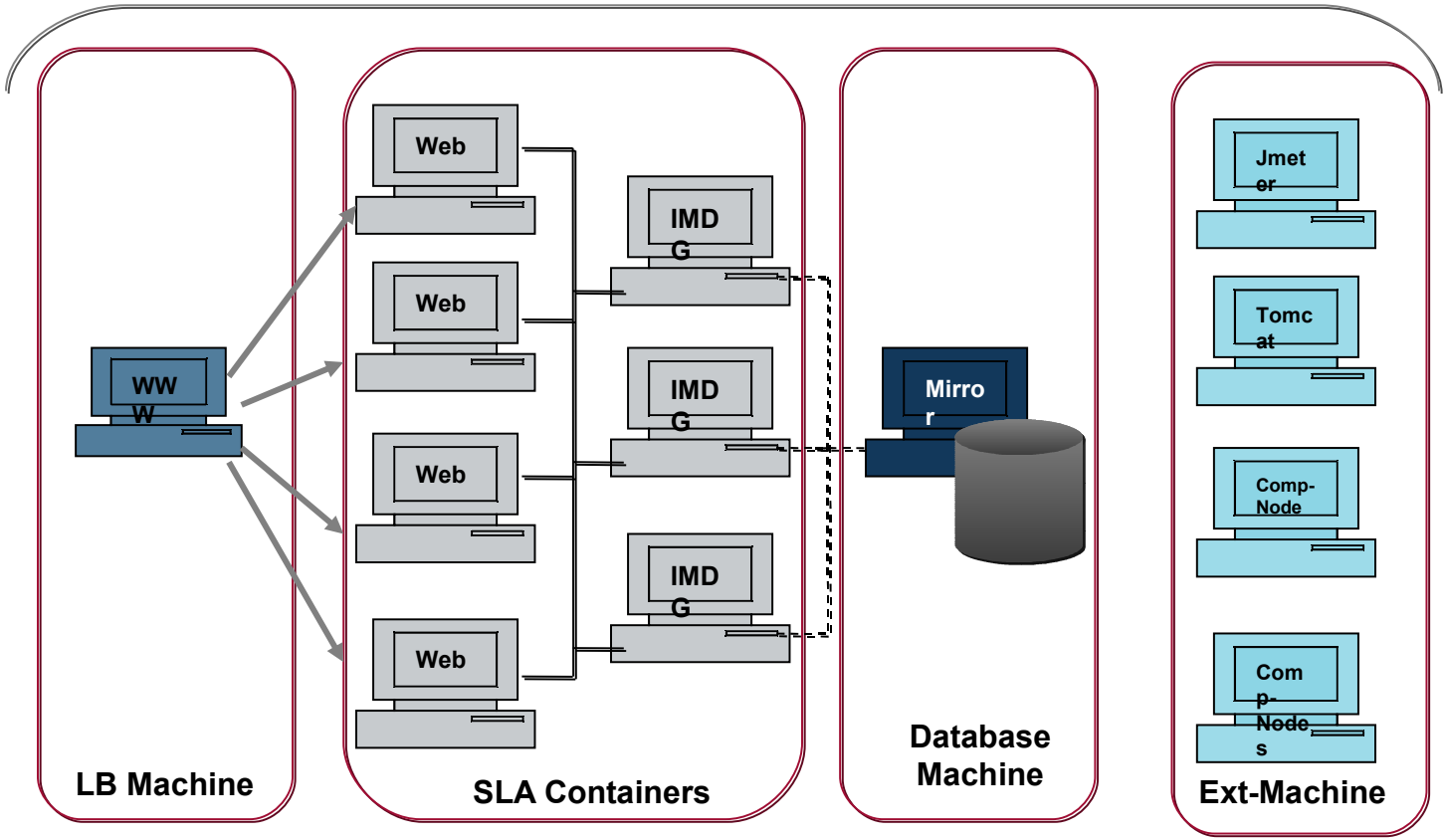
Application
Repository
(S3)



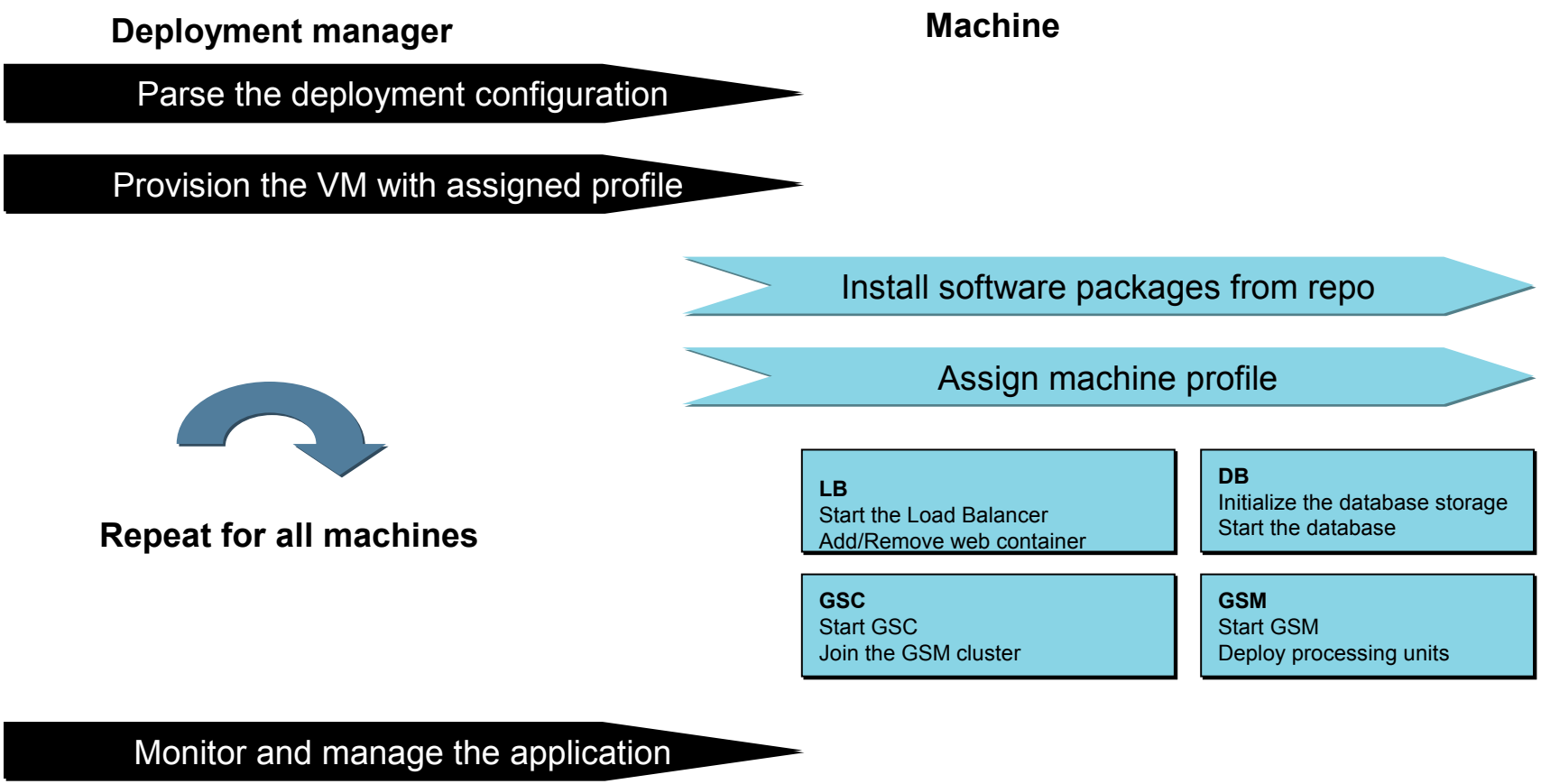
Dynamic Images



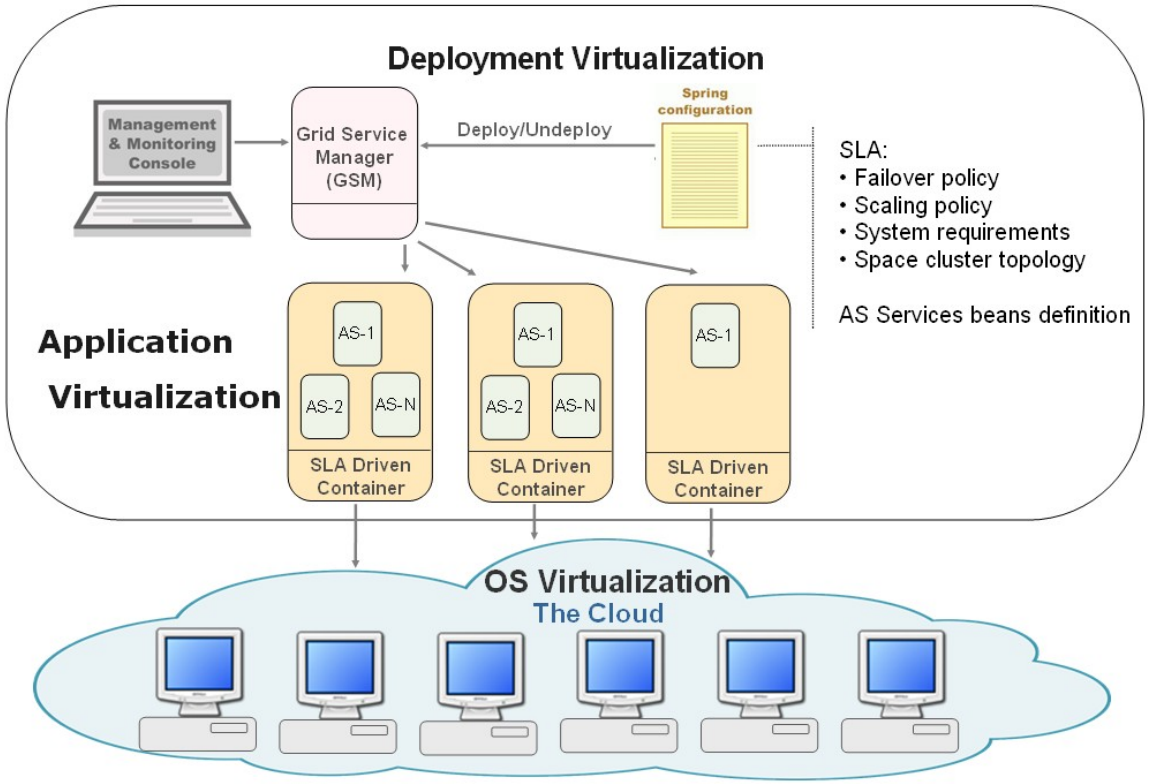
- Each machine can be assigned with
  - 64/32 S,M,L,XL Images individually
- Each machine dynamically install software packages
- GSM responsible for deploying application packages to GSC machines



Understanding the provisioning process



SLA Driven Containers of Containers



- Break physical machines into sub machines
- Automation of manual deployment
- Auto scaling
- Self healing
- Container of Containers
  - Jetty
  - Glassfish
  - Tomcat (Not supported yet)

Develop Custom SLA

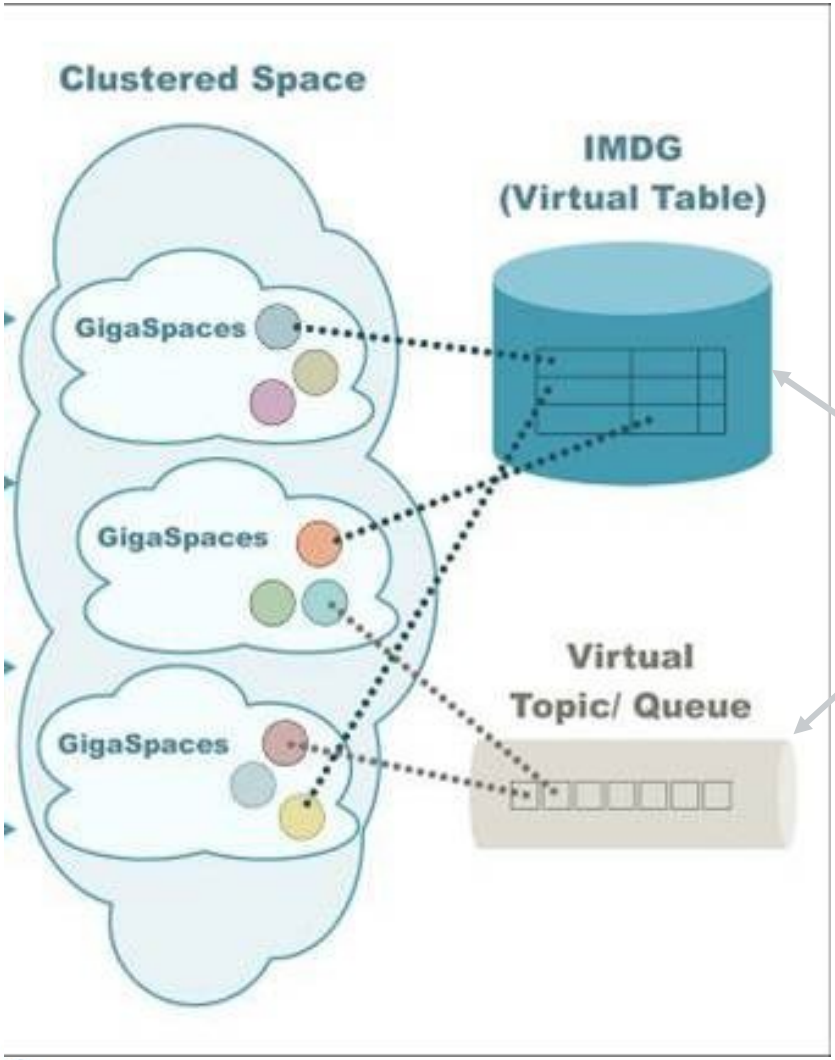
```
ProcessingUnit pu = admin.getProcessingUnits().waitFor("SimpleWebApp", 30000L, TimeUnit.SECONDS);

...

totalRequests = 0
previousTotalRequests = 0
//iterate over all processing unit intances
pu.each {ProcessingUnitInstance instance ->
    totalRequests += instance.statistics.webRequests.total
    previousTotalRequests += instance.statistics.previous.webRequests.total
}
//calculate average requests per container
averageRequests = ((totalRequests - previousTotalRequests) / pu.totalNumberOfInstances)
```

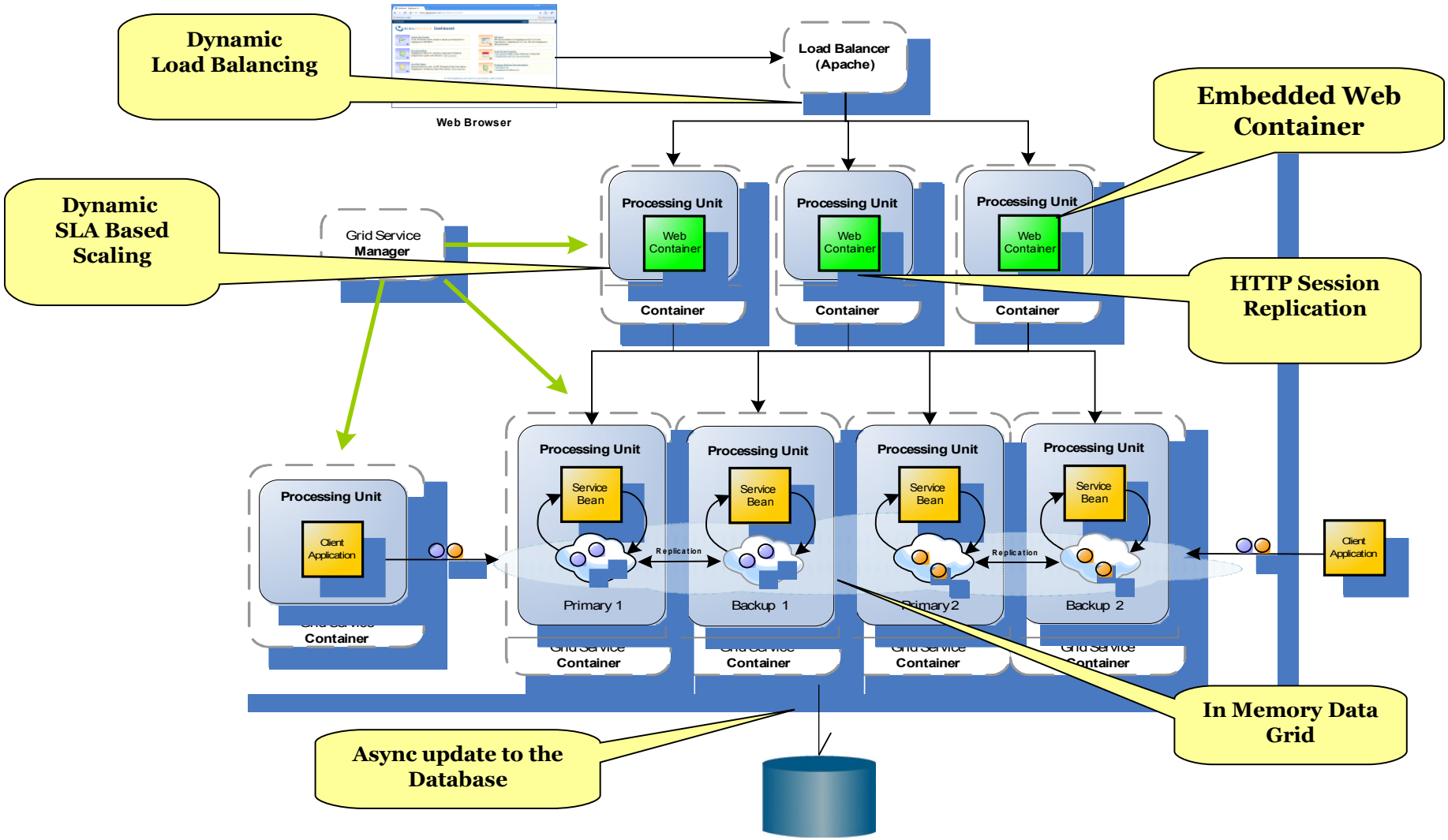
```
if (averageRequests < maxRequestPerInstance) {
    println "Scaling up..."
    //find an agent and start a grid service container on its machine
    admin.gridServiceAgents.agents[0].startGridServiceAndWait(new GridServiceContainerOptions())
    //increment the number of application instances
    pu.incrementInstance()
    //sleep to allow changes to take effect
    Thread.sleep(30000L)
}
```

Middleware virtualization



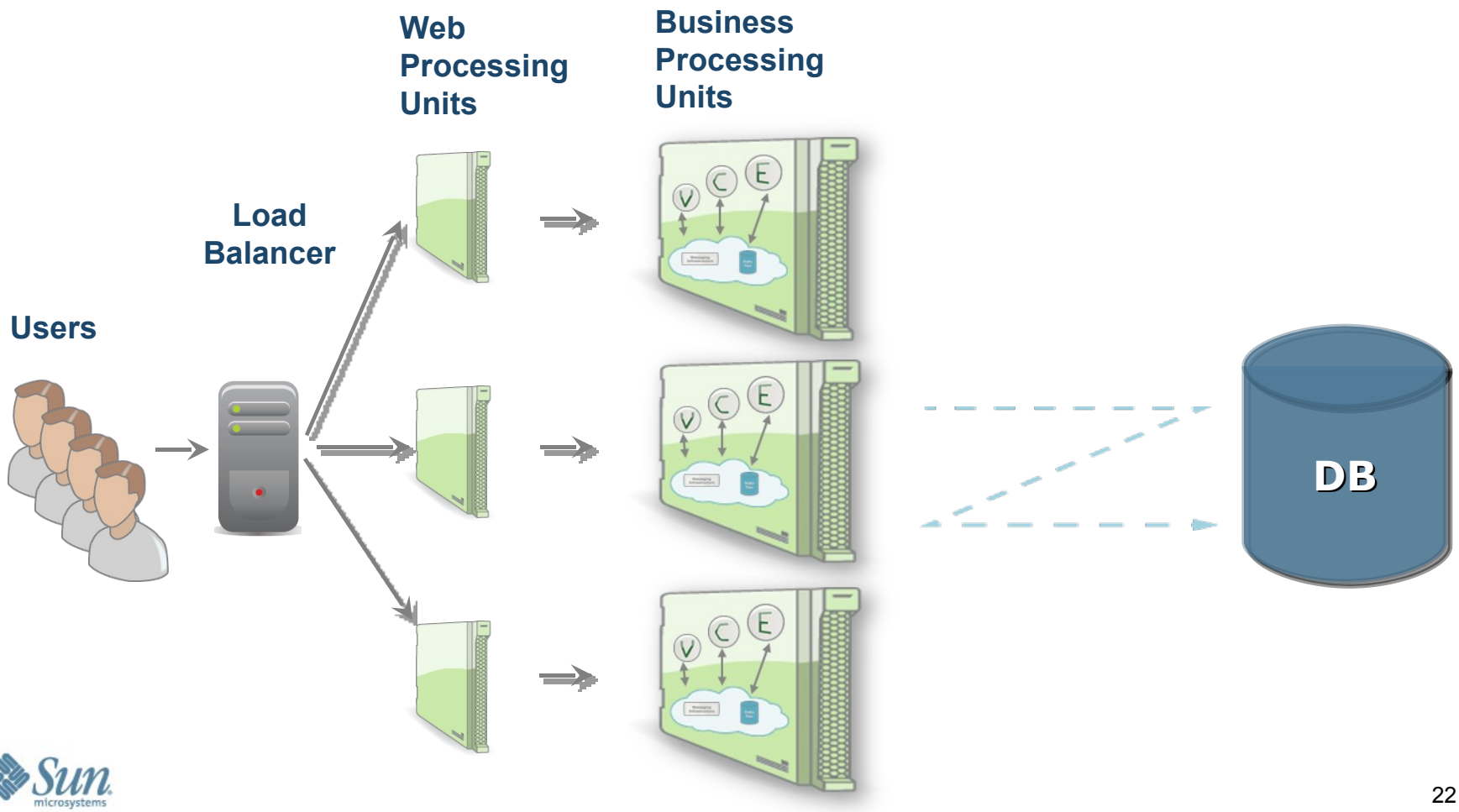
- "*All problems in computer science can be solved by another level of indirection*" (Butler Lampson)
- Similar principles to storage virtualization
- Decouple the application from the deployment environment
- Use partitioning to split the load and the data.

End 2 End Elasticity Load Balancer to Database



Design for linear scalability

Partition the entire application stack



Agenda

- Introduction to Cloud Computing
- AWS vs GAE
- Challenges of Enterprise Applications
- PaaS on EC2
- Case study – Primatics Risk Management as a Service
- Summary & References

Case Study:

Evolv Risk: Cloud Computing Use 2009

PRIMATICS® FINANCIAL

Francis de la Cruz
Manager, Primatics Financial
Argyn Kuketayev
Senior Consultant, Primatics Financial

Use Case Outline

- Business Challenge
  - Client Needs and Application Evolution
  - Business Needs
- Architectural Challenges
  - Risk 1.5 Application Stack
  - Risk 2.0 Application Stack
  - Processing Unit diagram
  - Lessons Learned

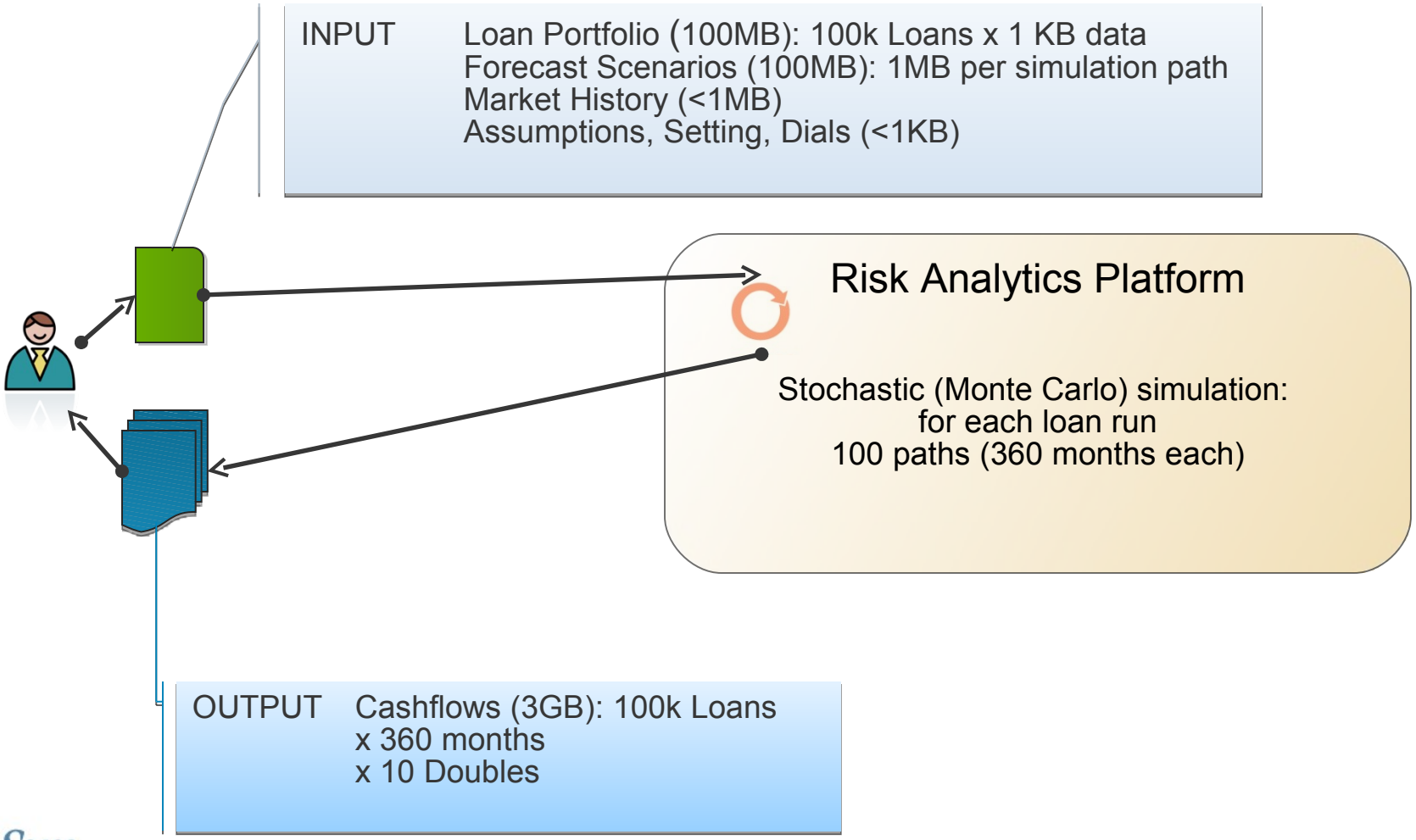
Business Case

- Many mid-size regional banks with sizeable mortgage portfolios
- Outsource sophisticated mortgage credit risk modeling
- Outsource IT infrastructure to perform costly computations, suited for on-demand PaaS

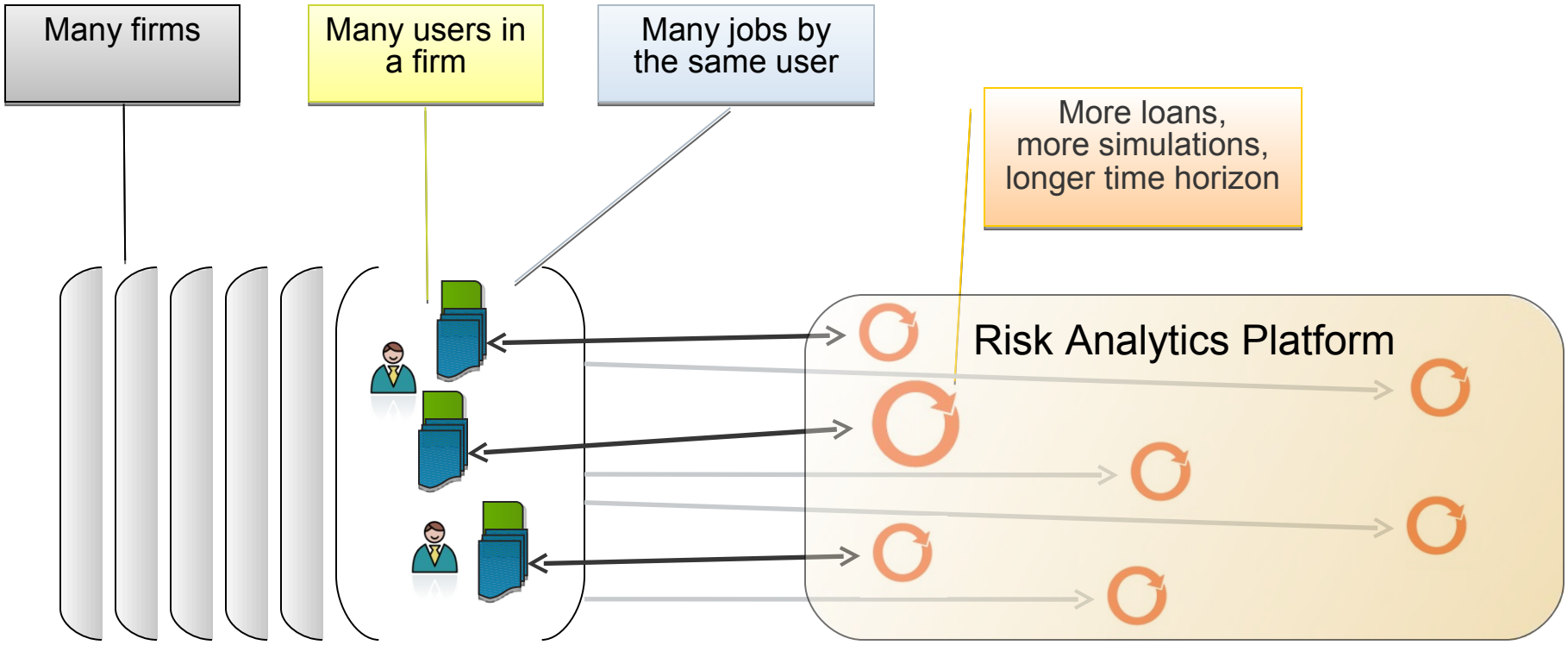
Primatics' Story in Cloud Computing

- Company with roots in financial services and IT
- Multiple clients in financial services, 2 products
- 2007
  - EVOLV Risk V1: Web-based hosted solution, a regional bank
- 2008
  - V2: embrace Cloud computing, Amazon EC2
  - V1.5: big national bank M&A, portfolio valuation, AWS EC2
- 2009
  - V2 "reloaded": ground up redesign, GigaSpaces

Business Needs: Accounting-Cashflows



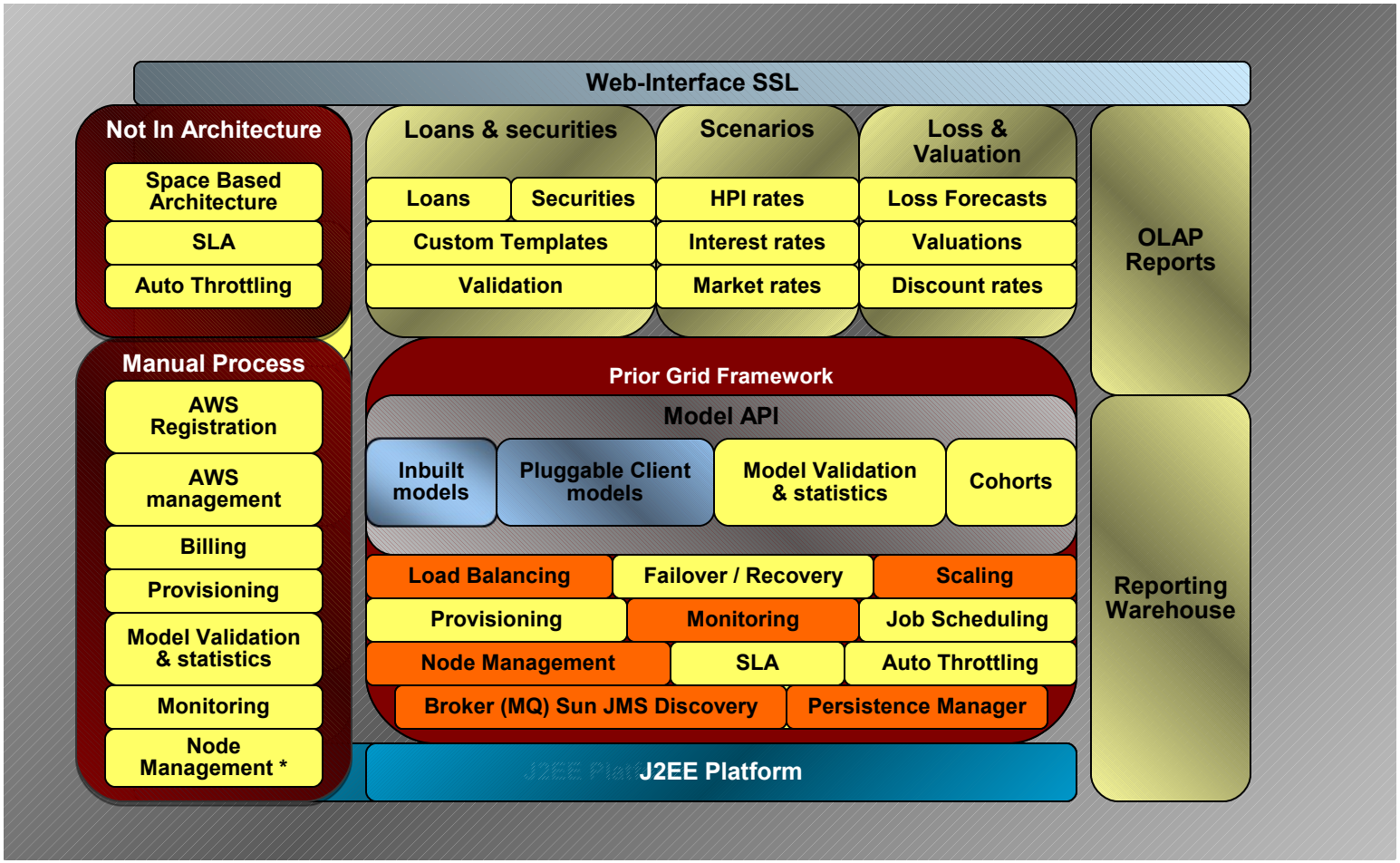
Business Needs: Scale



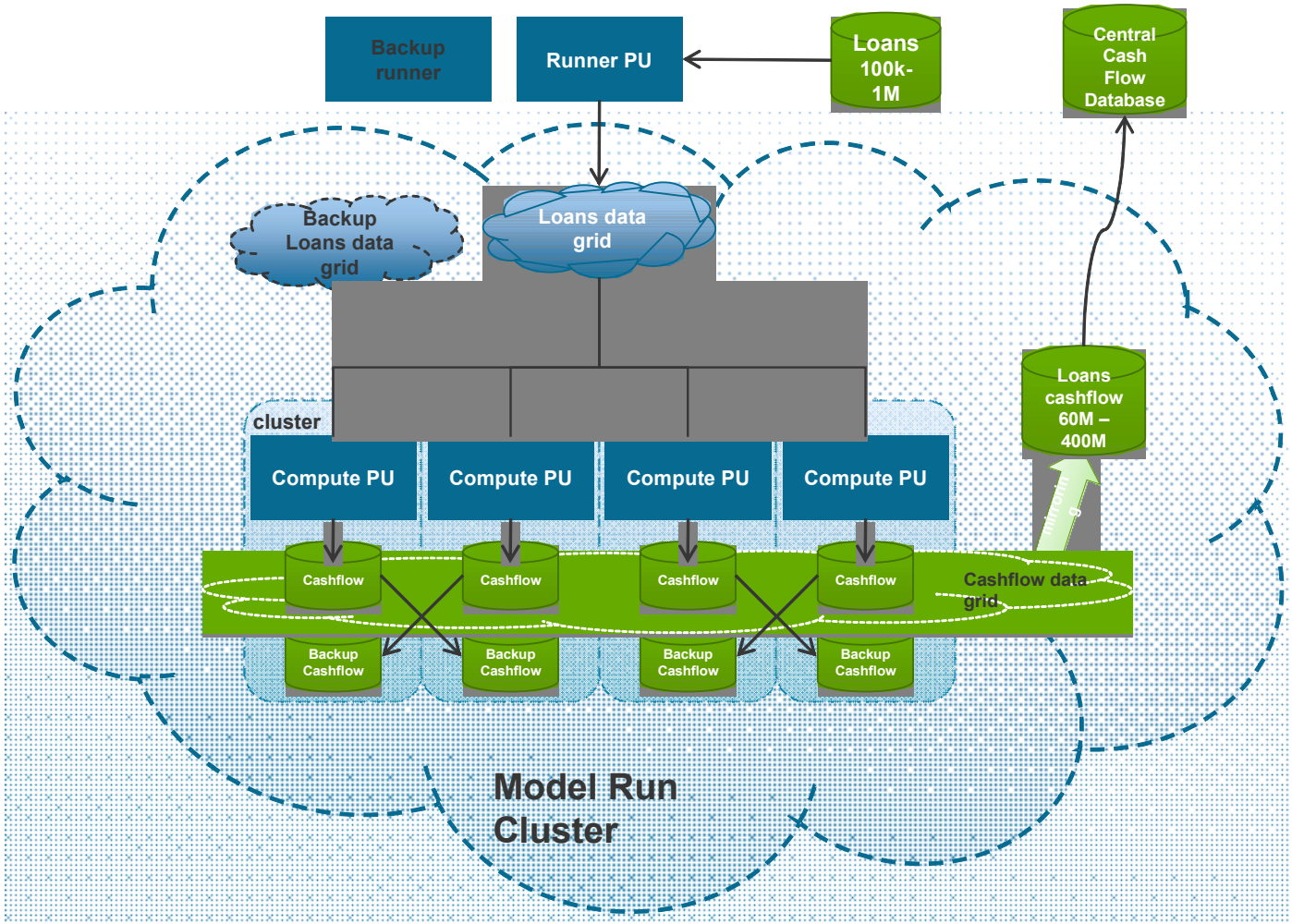
Overarching Challenges

- Quantitative financial applications (models) with heavy CPU load
- Large volumes of data I/O
- Infrastructure to support growing number of clients, and users
- Varying load on the system, with spikes of heavy load
- Avoid lock-in to cloud vendors

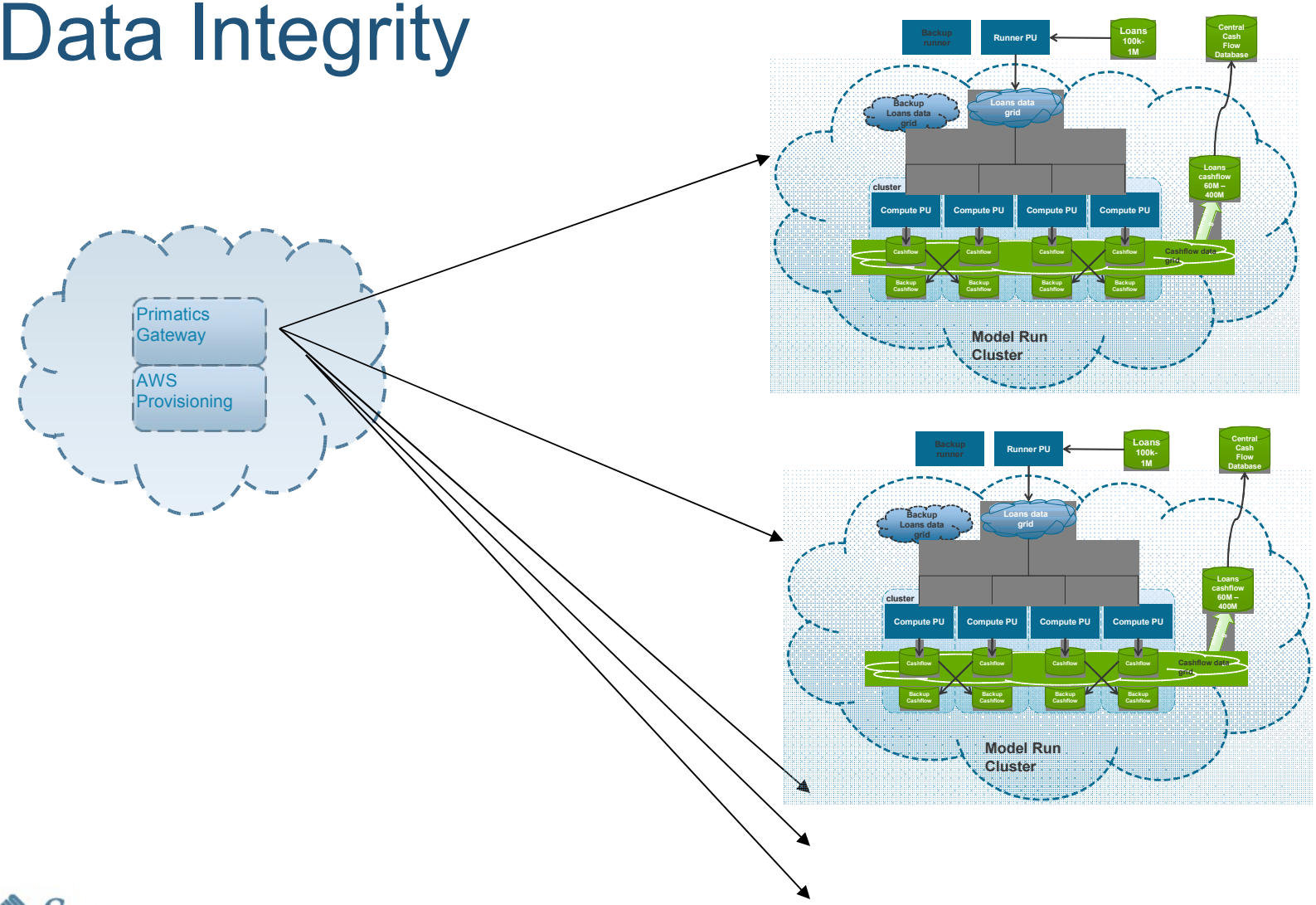
Evolv Risk 1.5 vs 2.0 Software Stack



Performance, Scalability, Data Security, Data Integrity



Performance, Scalability, Data Security, Data Integrity



Lessons Learned : Cloud Integration

- **Provisioning**
  - Configuration API vs. Using Scripts
- Auto Throttling vs. Manual Throttling
- Failover and Recovery provided by framework
- Monitoring VIA GS Console and API vs in house application.
- Infrastructure took 5 weeks vs 8 weeks to develop.
- Achieving linear scalability was a matter of creating Processing Units.

Lessons Learned : Data and Processing

- Data Distribution
  - Uses Spaces Architecture vs. In house developed JMS Broker and Discovery to push to Grid
- Deployment
  - Uses GS API vs. Pushing data to grid or using machine images.

Lessons Learned:

- Problems are now high class problems.
  - GigaSpaces Framework spared us from low level problems.
  - Code written is for more answering ‘why’ not answering ‘how’.
- GigaSpaces:
  - Provisioning out of the box
  - Failover and Recovery much easier
  - Monitoring through API and Console
  - SLA out of the box
  - Integrated into the Cloud
  - **Significantly lowers the barrier of entry into Cloud computing**

Agenda

- Introduction to Cloud Computing
- AWS vs GAE
- Challenges of Enterprise Applications
- PaaS on EC2
- Case study – Primatics Risk Management as a Service
- Summary & References

PetClinic in the Clouds: Scaling a Classic Enterprise Application
Wednesday 1:35 PM - 3:15 PM LAB-5564BYOL Hall E 132

Free drinks Webtide & GigaSpaces party – Tuesday 8 PM
SOMA 201 3rd St (gigaspaces.com/javaone)

Summary: Key Takeaways

- Adding PaaS to AWS brings the flexibility and performance of AWS and ease of use of PaaS

- Enterprise applications can run on the cloud today:
  - No need to re-write your application
  - Preventing lock-in to specific cloud provider
  - Enabling seamless portability between your local environment to cloud environment

- Choose simple applications first
  - Avoid dealing with complex security issues
  - Application with Clear path to ROI (Fluctuating load, large scale testing, DR,..)

References

- PetClinic in the Clouds: Scaling a Classic Enterprise Application
  - Wednesday 1:35 PM - 3:15 PM LAB-5564BYOL Hall E 132
- Cloud Mailing List
  - http://groups.google.ca/group/cloud-computing
- Amazon Cloud:
  - aws.amazon.com
- GigaSpaces PaaS on AWS
- gigaspaces.com/mycloud

Thank You

Nati Shalom
Natishalom.typepad.com
Twitter.com/natishalom

Run your own demo
http://www.gigaspaces.com/mycloud

Speakers

Argyn Kuketayev
Senior Consultant

akuketayev@primaticsfinancial.com

703 342 0053

>quant development team lead

>using Java since 1998

>degrees in Physics and Finance

>started in scientific computing

Francis de la Cruz
Manager

fcruz@primaticsfinancial.com

703 342 0438

>Has been using the Java Programming Language since the days of Java 1.2

>Previously worked on the DoD and Security domain before moving on to the Financial Domain.

>Degree in Electrical Computer Engineering

>Has created many dead end open source projects notables including

- Active Record framework using annotations, proxy and Apache Torque
- HTML based web framework
- Java based batching and processing application