Migrating Java Applications to vFabric

Cloud Application Platform – The Journey to PaaS

March 2012
Migrating to vFabric – Key Takeaways

- Build Cloud Application Platform – Start your journey to PaaS

- Migration has real business benefits
  - Increased agility and reliability
  - Reduced deployment downtimes
  - Scales to future growth
  - Reduced Capex and Opex costs
  - Highly adaptive to business and customer needs
Problem Context

Enterprise custom Java applications are expensive and complex to build, deploy, scale and manage

Typical problems

- Development
  - Obsolete toolkits and no clear choice of toolkit
  - No fully integrated stack, resulting in lack of control and risk
- Operational
  - Lack of support for developer-friendly tools
  - Lack of industry and community
- Industry
  - Shortage of skill sets
  - Fragmented market space leading to uncertainty

VMware problem

- Higher % of downtime leading to poor customer satisfaction
- High cost of development and operations
Why Consider PaaS: The IT Perspective

Business expects agility, reliability and lower costs…

… but I have performance, instability, manageability monitoring and analyzing issues with the current platform!

… what is needed is a scalable, robust and manageable platform that is built on modern tools and technologies…

…faster service execution at lower cost!

VMware IT
Why Migrate?

- **General solution to our problem: migrate to a new platform**

- **Constraints**
  - Retain functionality
  - Address perceived issues

- **Creating a business case**
  - Due to platform issues there is a high cost for
    - Downtimes
    - Ad-hoc fixing of problems
    - Not having the right monitoring tools
    - Development with a disparate technology stack.
    - Hardware costs to scale with complex resource-intensive application servers

- **Migration Costs**
  - New platform creation
  - Code migration
  - Cost of testing
The Business Case for PaaS

- Core value proposition: increase the availability and resiliency, and reduce development and operational costs.

- What can we achieve?
  - Quantitative:
    - Reduce ongoing development and operational costs by up to 15%
    - Reduce hardware costs by up to 30%
    - Reduce downtime support requests by up to 25%
    - Reduce software costs by up to 40%
  - Qualitative:
    - Improve reliability, availability and scale
    - Standardize technology stack with a full fledged integrated platform
    - Increase agility, productivity and reusability
    - Embrace open source with abundant skill set availability
  - Strategic:
    - *Cloud Application Platform – start the journey to the PaaS*
Defining the scope of migration

- **Factors to consider**
  - Changes in architecture?
  - Functionality changes?
  - Code refactoring?
  - Code changes to follow the new patterns for the new platform
  - Fixing known bugs and metrics for code quality

- **Typical choices**
  - Migrate the code as is: minimal changes
  - As part of migration make changes that address current issues
  - Guideline: use business case to create a migration strategy

- **Define phases and success criteria**
  - Guided by business case and visibility requirements
Our scope

Solution robustness
- Architectural pattern changes in middleware
- Support for inversion of control: code refactoring
- Design patterns to incorporate fault-tolerant caching

High availability
- Reduced dependency on back-end: offline mode
- Increased performance through selective code refactoring
- Fault-tolerant application server architecture

Operational enhancements
- Addition of monitoring tools to the deployment and manageability
- Automated provisioning of platform instances.

Summary: we had to change the architecture to support the business case and the code to support those architecture changes; code changes to Spring paradigm such as IoC. **No new functionality.**
Defining the New Target Platform

- General framework
  - Which components of vFabric to use?
  - Which third party (legacy components) to integrate?
  - Other improvement choices
    - Caching (Gemfire)
    - Monitoring and Metrics (Hyperic)
    - Operational improvements: vCloud director, vCO

- What we did
  - Full fledged vFabric stack to unlock the full potential
  - Optimized application performance using Gemfire
  - Stability and availability with fault-tolerant cluster on lightweight tc Servers
  - Proactive monitoring, diagnostics and management using Hyperic
  - Application workloads provisioning and management using vCloud Director and vCO
New Target platform

Cloud Application Platform – Start Journey to PaaS
Architectural changes

Middleware changes

- RMI -> SOAP for fault tolerance and better load balancing
- Added offline mode
- Build time stub generation using JAXWS
- Introduced service virtualization

Portal changes

- Spring WS for lazy loading of WSDLs
- Spring WS interceptor to capture, translate and handle SOAP faults
- Enabled vFabric-based session failover

Caching changes

- Earlier ehCache; now Gemfire DedicatedCache
- Earlier no session failover; now Gemfire cache with tc Server for session failover

Monitoring changes

- Monitoring of vFabric tc Servers, Gemfire with Hyperic
- Log mining and alerting Splunk
Deployment Architecture

- Earlier pain points
  - 14 portals on 42 VMs; 28 managed nodes and 14 admin. nodes
  - Unequal load – rarely used apps. also consumed a VM
  - Lack of proactive monitoring of application server resources
  - No centralized management; maintenance and support troubles
  - Long downtimes during upgrade/maintenance

- New deployment architecture
  - Cloud-ready application platform – Journey to the PaaS
  - Consolidated to 14 VMs from 42 VMs
    - Critical apps: 4 nodes; rest 2 nodes each + 2 admin/Hyperic nodes + 2 nodes Gemfire
  - App. Clusters by SLA, usage
    - Customer-facing critical; customer-facing non-critical; non-web apps; internal apps
Deployment Architecture

Application Security
- Access Manager Cluster
- OVD Cluster
- OID Cluster

Web Layer
- Apache Webserver Cluster
- Webgate Cluster

Application Layer
- Critical Cluster – 1 (TcServer)
- Critical Cluster – 2 (TcServer)
- Non-Critical Cluster – 1 (TcServer)
- Internal Apps Cluster (TcServer)
- GemFire Cluster

Middleware
- WebServices Cluster (TcServer)
- OSB Cluster (Oracle 11g)
- BPEL Cluster (Oracle 10g)
- OHS Cluster

Back Office
- Oracle EBS (Oracle 10g)
- Oracle UCM (Oracle 10g)

Application Management
- Server Management (Hyperic)
- Application DB (Oracle 11g)
- Mgmt Database (MySQL)
Deployment Architecture

- **Easier maintainability**
  - Hyperic reduced the need for admin nodes
  - Proactive Monitoring and Alerting

- **Optimized performance**
  - Gemfire in-memory data grid improved performance
    - Optimized the JVM heap usage on tc Servers using Gemfire caching
    - Provided tc server session failover
    - Provided a high-performance, scalable and fault-tolerant cache solution
    - Provide application-level, user-level caching
Lessons learned

- Lockdown the scope and avoid functionality scope creep
- Be prepared to re-factor code, as there is no one-to-one pattern translations for all the patterns
- Lockdown the target platform components, and avoid introducing new components
- Define usage patterns of new frameworks, components for faster on-ramp and code quality
- Define the criteria and the scope of different caching levels usage for optimal performance
- Allocate large amount of time for performance tests, as tuning of new platform is an iterative process
- Minimize business UAT test time as no functionality change involved and complement with automated regression testing
Benefits and Outcome

- Operational consolidation and reduced hardware footprint

**Pre Migration Environment**
(Legacy App Server Clusters)

- App Cluster1
- App Cluster2
- App Cluster3
- App Cluster12
- App Cluster13
- App Cluster14

**Post Migration Environment**
(Spring tc App Server Clusters)

- App Cluster1
- App Cluster2
- App Cluster3
- App Cluster4
- App Cluster5

- Reduced hardware footprint by ~66%
- Reduced OPEX costs by at least 15%
- Significant $ savings with the retirement of legacy application licenses

<table>
<thead>
<tr>
<th>Legacy Application Servers</th>
<th>vFabric tc Servers</th>
<th>Reduction of Footprint</th>
</tr>
</thead>
<tbody>
<tr>
<td># Number of App Servers (incl. admin nodes)</td>
<td>42 App Servers VMs – 14 portals</td>
<td>14 App Server VMs – 14 Portals</td>
</tr>
<tr>
<td># Number of App Server Clusters</td>
<td>14 Clusters – 2 App Servers/Cluster</td>
<td>4 Clusters – 2 App Servers/Cluster (Clusters by functionality/volume/criticality)</td>
</tr>
</tbody>
</table>
Benefits and Outcome

- Improved deployment agility and efficiency
- Reduced downtimes by five times
- Improved performance of key transactions by using Gemfire
- Improved Security using Spring security and Spring LDAP

<table>
<thead>
<tr>
<th></th>
<th>Legacy Server (before)</th>
<th>vFabric tc Server (After)</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. deployment time (minutes)</td>
<td>20 mins / cluster</td>
<td>5 mins / cluster</td>
<td>4 times</td>
</tr>
<tr>
<td>Avg. shutdown/eestart time (minutes)</td>
<td>20 mins / Application (10 mins / node)</td>
<td>2 mins / node</td>
<td>5 times</td>
</tr>
<tr>
<td>Overall downtime of Web applications (minutes)</td>
<td>30 mins / Application</td>
<td>6 mins / application</td>
<td>5 times</td>
</tr>
</tbody>
</table>

*Data sampled over 3 month averages before and after deployment*
For More Information

- Thirumalesh Reddy, Director of Applications Architecture and Tech Labs
  - treddy@vmware.com