Migrating a Java Solution to HP Integrity NonStop
A mix ofWhats, Whys and Hows

Moore Ewing  HPE EMEA NonStop Presales
What 1. A Card Management Solution
Create Programs, Issue Cards, Authenticate Use, Manage Accounts for 5.
What 2.
FSS CMS 5.0
Why 1.
Natural Affinity.

– **HPE**.
  – Looking for new solutions in new technologies
  – Lacking a Card Management solution

– **FSS**
  – Aware of NonStop as the leading top tier financial platform
  – NonStop experience as ACI Base-24 service provider
  – Authorisation is 24/7 and even Back-Office is becoming mission critical
Why 2.

FSS had seen these slides

HA and Availability - SMP Cluster or HPE Integrity NonStop

Configuring an Available, Scalable web application

<table>
<thead>
<tr>
<th></th>
<th>conventional</th>
<th>NonStop</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of systems to manage</td>
<td>6 (x 1 Blade)</td>
<td>1 (x 2 + Blades)</td>
</tr>
<tr>
<td>No. of ports to configure</td>
<td>Many</td>
<td>1</td>
</tr>
<tr>
<td>Webserver/Tomcat/DB mappings</td>
<td>In multiple files on each system</td>
<td>1 Filemap</td>
</tr>
<tr>
<td>Start/Stop scripts</td>
<td>Multiple per system</td>
<td>./start ./stop</td>
</tr>
<tr>
<td>Logfile sets</td>
<td>Multiple per system</td>
<td>1</td>
</tr>
<tr>
<td>Transaction/Resource Managers</td>
<td>Multiple per system</td>
<td>1 (TMF)</td>
</tr>
<tr>
<td>HA DB Cluster configuration</td>
<td>Complex</td>
<td>None</td>
</tr>
<tr>
<td>Expansion</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

Expanding an Available, Scalable web application

<table>
<thead>
<tr>
<th></th>
<th>conventional</th>
<th>NonStop</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of systems to manage</td>
<td>6 (x 1 Blade)</td>
<td>1 (x 2 + Blades)</td>
</tr>
<tr>
<td>No. of ports to configure</td>
<td>Many</td>
<td>1</td>
</tr>
<tr>
<td>Webserver/Tomcat/DB mappings</td>
<td>In multiple files</td>
<td>1 Filemap</td>
</tr>
<tr>
<td>Start/Stop scripts</td>
<td>Multiple, correct order</td>
<td>./start ./stop</td>
</tr>
<tr>
<td>Logfile sets</td>
<td>Multiple per system</td>
<td>1</td>
</tr>
<tr>
<td>Transaction/Resource Managers</td>
<td>Multiple per system</td>
<td>1 (TMF)</td>
</tr>
<tr>
<td>HA DB Cluster configuration</td>
<td>Complex</td>
<td>None</td>
</tr>
<tr>
<td>Expansion</td>
<td>Add systems Adjust the above?</td>
<td>Add cpus Up Numstats</td>
</tr>
</tbody>
</table>
Why 3.

What FSS had...

What FSS would like to have...

Single system web application configuration
What 2.

FSS CMS 5.0 ; Two applications in one solution
What 3.

Two JAVA Technologies

Carduser

Acquirer

Switch

CMS 5.0
Card & Program Management

CMS 5.0
Realtime Authorisation

JEE
JBoss4

CMS 5.0
Users

JSE
Custom Code
How 1.
Two guiding principles in planning

1. Commonality of Code
   - Same business logic in both Linux and NonStop versions

2. Isolation of Migration effort
   - minimise possible delays due to complications in other components
   - simplify scoping of problems
   - early detection of issues in any component
How 2.
Four components …… three teams

- J2EE user-driven card/program management application
  - The Application Team
- JSE Authorisation “messaging” application
  - The DataBase
  - The DataBase Team
- The NonStop System and Software
  - The Infrastructure Team
CMS 5.0 J2EE application migration to NSASJ
What x.

NSASJ : wrapping JBoss AS7 code into TS/MP

HP Integrity NonStop

TCP/IP or CIP

NSASJ-HC
<<tsmp-pp>>
Host-/Domain
Controller

NSASJ-PM
<<tsmp-pp>>
Post
Master

NSASJ (JBoss7
code)

Infinispan
Cache

SQL/MX

tmp-pp…persistent process
What x.
NSASJ Overview

HP Integrity NonStop

NonStop Application Server for Java (NSASJ)

JBoss EJB Client

TCP/IP or CIP

tcp/ip

Post Master
Post Master
Post Master

NSASJ

Infinispan Cache

SQL/MX
What 4.

Scalable, HA Deployments: Wildfly (JBoss) and NSASJ

NSASJ derived from JBoss AS

JBoss AS Domain Deployment
How x.
Isolation and verification

JBoss4 -> JBossAS7 with ORACLE DB

JBossAS7 (Wildfly8) -> NSASJ with remote ORACLE (T4) DB

JBossAS7 with 1st cut SQL/MX(T4) DB

NSASJ with 1st cut SQL/MX(T2) DB

JBossAS7 with 2nd cut SQL/MX(T4) DB

NSASJ with 2nd cut SQL/MX(T2) DB

JBossAS7 with 3rd cut SQL/MX(T4) DB

NSASJ with 3rd cut SQL/MX(T2) DB

Database Development Delivery
What x.
The JBoss to NSASJ migration experience

• Went very much as expected. HPE assistance given to avoid issues in thread, pool and heap parameters

• Some minor product issues, e.g. with Hibernate and SQL/MX Sequences. Were quickly resolved by HPE.

• The STARTUP Time issue
  - resolved by improved use of JBoss AS7 packaging and dependency definitions
  - a JBoss4 -> JBoss AS7 change which benefited Linux version as well
  - more obvious in NSASJ test as 8 JBoss instances not one

• Actually done twice
  - migrated and started testing with older FSS ACM product
  - switched to CMS 5.0 when Linux development finalised
CMS 5.0 Database migration
How x.
Database migration in 3+ cuts
Remove it as a Gating Factor in the overall process

1st Cut
Logically correct Statement tests

- Create tables with minimal change
- Default size and placement
- Collect planning/tuning info for tables for test and initial production volumes
- Collect significant SQL statements

2nd Cut
Realistic PDBD first performance test

- Partitioning, location, space allocation
- Secondary indexes for access paths and RI/FK constraints
- RI/FK constraints
- Load test data
- For MXCS, create test/tuning environment

3rd Cut
Refined PDBD SQL statement change?

- Revise PDBD for -datatypes -Keys (Primary, Clustering, Partition) -table splits -access paths and indexes -RI requirement -Triggers -Stored Procedures
- Revise statements for new PDBD
- Re-test and reiterate as required
What x.
Many tables but relatively simple

- Hundreds of tables but mostly small static dimension tables
- No complex Triggers or Stored Procedures
- No complex datatypes or ADTs
- Most complex joins only used during initialisation

PLEASANT SURPRISE!

Easier and faster than anticipated due to lack of any heavy ORACLE dependencies
How x.
Compromise until change is really needed - retain code compatibility

• Used original DDL input scripts not ORACLE generated output. Made mass editing of simple syntax changes easier

• Three separate ORACLE databases retained as three schemas

• Most active tables keyed for partitioning but not implemented. Makes collecting metrics easier.

• Some synthetic keys were replaced as Primary Key by a composite natural key. Allowed the removal of associated indexes and simplified joins

• Redundant joins removed.
• Joins of 3+ small, static tables could be “materialised” but infrequently used

• Three TMF transactions retained due to changes in the business logic required to remove them Provides a known improvement if needed in the future or requirement for common code is relaxed

• Some of the changes benefited the Linux/ORACLE version.
CMS 5.0 Authorisation application migration
What x.

A multi-threaded singleton.

Listener Thread

Authorisation Worker Threads

But

so

was

JBoss
How x.

Java Infrastructure – the secret sauce of NSASJ: turns Socket I/O into Pathsend.

Open Source Component. Uses JI to act as Pathsend client.

JBoss AS. Uses JI to run as a TS/MP server class.

Java Infrastructure – the secret sauce of NSASJ: turns Socket I/O into Pathsend.

EJB Client

TS/MP Pathsend API

Port Mapper

JI library

Port Mapper

JI library

Port Mapper

JI library

Port Mapper

JI library

NonStop Server

CPU 0

CPU 1

CPU 2

CPU 3

NSASJ

JI library

NSASJ

JI library

NSASJ

JI library

NSASJ

JI library

Open Source Component. Uses JI to run as TS/MP server class = NIL!!!
How X.

How to enable JI

- Add ji.jar to classpath
- Add libji.so directory to RLD path
- Export JI_ENABLE=true
- Configure JI in ji.properties
- Reference ji.properties from system as system property or set as export

Sample client code

```java
........
........
(2) Socket sock = new Socket("nonstop.server.com", 8070);
........
OutputStream os = sock.getOutputStream();
InputStream is = sock.getInputStream();
........
(3) os.write();
........
(6) is.read();
........
sock.close();
........
```

Sample server code

```java
........
........
ServerSocket ssock = new ServerSocket(8070);
........
(1) Socket sock = ssock.accept();
........
InputStream is = sock.getInputStream();
(4) is.read();
........
OutputStream os = sock.getOutputStream();
(5) os.write();
........
```
How x.

Change the Authorisation architecture; not the code.
HOW (not) x.

Do not jump in the deep end to see if you can swim

– Concerns about NSJ multi-threading capability
– Created a test program to compare single-threading and multi-threading
– Ran a stress test load against it

– Guess what? It did not do very well!!

– Unrealistic thread levels
– Inconsistency between TS/MP links, thread pools and connection pools
– Badly tuned SQL statements
– UNNECESSARY LIBRARIES

– Incremental Testing with the real application code introduced
Incremental Performance Testing
And Sizing Metric Collection
FSS CMS 5.0   Performance Validation

Testing is not in “benchmark” format but has multiple goals
- verify product is functionally correct on NS
- verify that performance is acceptable and deemed competitive with CMS on other platforms
- Experiment with configuration options to provide some best practices

NSASJ CM application -  people-driven + batch like file and report processing
Performance at or above expectation

Realtime Authorisation -  not lightweight ; 12 medium SQL statements per request
Performance again acceptable and competitive

Tests on HP NS2404 with limited number of discs are very satisfactory.
Testing on NSX7 in various core configurations is underway.
Not appropriate to provide figures in this presentation.
Incremental performance and sizing approach

– Performance Metrics and Scalability Testing for system sizing

– This is a progressive and cyclic process to establish
  -- the servicetime and costs of a single transaction
  -- the maximum number of concurrent transactions per NSASJ instance
  -- the maximum number of concurrent transactions per cpu core
    - the maximum number of concurrent transactions per logical cpu
  -- that linear scalability exists across an increasing number of logical cpus.
– The maximum number in any test is determined by a trade-off between the acceptable response time of a transaction and the throughput of the component.
  – (The relationship of responsetime to servicetime under increasing load as described in Queuing Theory.)

Sizing calculations will depend upon the maximum cpu utilisation deemed acceptable with regard to business risk of cpu failure during peak periods.
# Incremental Testing to find concurrencies and throughputs

<table>
<thead>
<tr>
<th>Test Number</th>
<th>No. of CPUs</th>
<th>No. of cores per cpu</th>
<th>No. of instances per core</th>
<th>No. of threads per instance</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Driver should supply requests to keep application busy at a concurrency of one. This will establish servicetime and resource costs from which other estimates can be calculated.</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>as available</td>
<td>1</td>
<td>1</td>
<td>1 To ensure no contention between cores.</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>incrementing Target No. based upon cpu usage metric and acceptable responsiveness from test 3 Target No. based upon cpu usage metric and acceptable responsiveness from test 3</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>Incrementing</td>
<td>1</td>
<td>from test 4</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>as available</td>
<td>from test 4</td>
<td>1</td>
<td>from test 3 To ensure no contention between cores.</td>
</tr>
<tr>
<td>6</td>
<td>Incrementing</td>
<td>as available</td>
<td>from test 4</td>
<td>1</td>
<td>from test 3 To ensure no contention between cpus.</td>
</tr>
</tbody>
</table>

Throughput metrics gathered from dedicated cpus and cpus with light and heavy disc access for variable sizing.
The bottom line
Using the HPE NonStop Java ecosystem delivered
FSS CMS on HP NonStop: Simple yet Deep Port

CMS on Linux
- Single instance JBOSS CM (or cluster)
- Single process m-t r/t Authorisation

CMS on NS
- NSASJ CM
- Scalable multi-cpu architecture

NS Fundamentals Inherited By CMS
- Scalability: 2-4000 cpus
- Availability: AL4 FT system, no SPOF
- Small fault zones: cpus and serverclasses
- Distributable FT RDBMS: co-located, no network costs
- Ease of Management: integrated software; no separate cluster or RDBMS or BC
What!
FSS project receives Frost and Sullivan award for process innovation

FSS was awarded the first runner up for process innovation category as part of the Project Evaluation and Recognition Program (PERP) 2015 by Frost & Sullivan. FSS won the award in the Service Sector for its project titled “Re-engineering FSS Card Management Suite on HP NonStop platform.”

FSS Card Management Suite is a single platform for debit, prepaid and virtual card from pre-issuance to post-issuance

The objective of the project was to build a next generation card management product on HP NonStop and innovate to help clients fully leverage the advantages of HP NonStop platform.
Thank you
moore.ewing@hpe.com