

# Implementing IBM WebSphere eXtreme Scale with WebSphere Application Server

Alan Chambers

Orbital Integrated Solutions Ltd  
*alan.chambers@orbital-ltd.co.uk*

## Copyright

- **Copyright Notice:**

This presentation and its content is copyright of Kupon Consulting Ltd - ©Kupon Consulting Ltd 2012. All rights reserved.

Any redistribution or reproduction of part of all of the contents in any form is prohibited.

You may not, except with the express written permission of the copyright owner, distribute or commercially exploit the content. Nor may you transmit it or store it in form of electronic retrieval system.

- **Enquiries and feedback**

For all queries, contact the author at: [tadleygreen@gmail.com](mailto:tadleygreen@gmail.com)

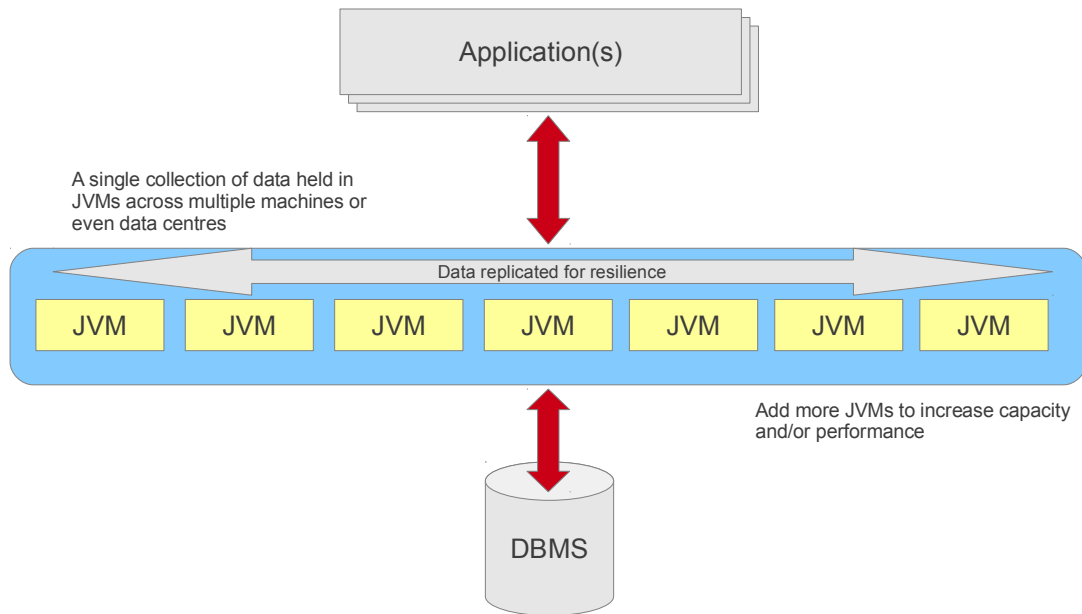
## Topics

- What is WebSphere eXtreme Scale?
- Simple topologies
- Architectural components:
  - Partitioning and replication
  - Scale-out and scale-in
  - The Catalog Service
- Configuring WXS with WebSphere Application Server
- Implementing WAS/WXS Use Cases

## What is WebSphere eXtreme Scale?

- An In-Memory DataGrid (IMDG)
  - A shared in-memory space for data
  - Seamlessly combining the heaps of 100's of JVMs on a network
  - A transactional programming interface for applications to store/retrieve data
- Typically used for high performance, extremely scalable, distributed caching solutions
- A plug-in cache for e.g.
  - HTTP Sessions
  - WAS Dynacache
  - Hibernate/JPA
- Other uses:
  - A high-performance store for shared application state
  - A platform for grid-based application processing

# What is WebSphere eXtreme Scale?



# WXS topology: local cache in single JVM

- Very similar to a Java Map
  - Object pairs: {key, data}
  - put/get semantics
- Adds:
  - Transactional control
    - begin, commit, rollback
  - Concurrency control
    - For access from multiple threads

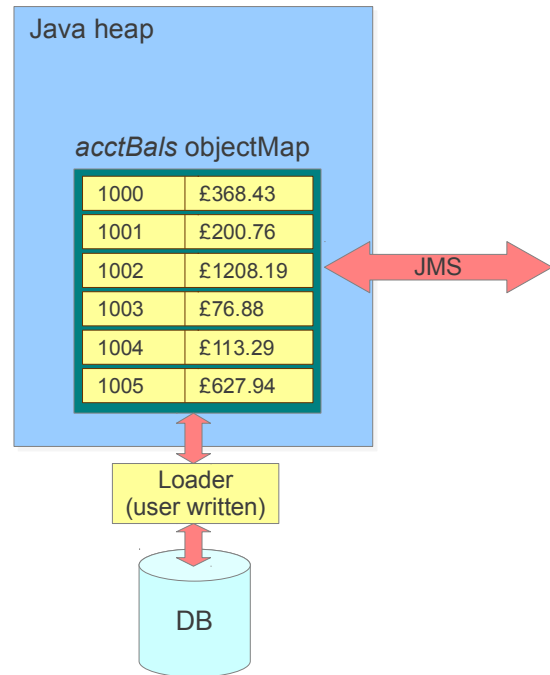
Java heap

*acctBals* objectMap

1000	£368.43
1001	£200.76
1002	£1208.19
1003	£76.88
1004	£113.29
1005	£627.94

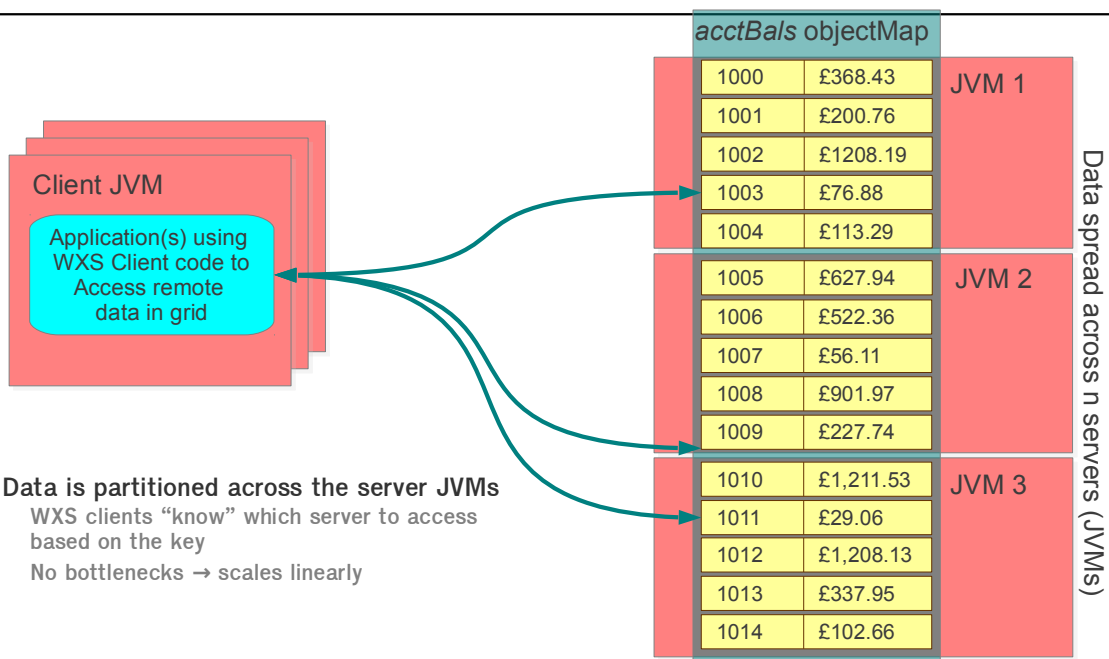
## WXS topology: local cache in single JVM

- Very similar to a Java Map
  - Object pairs: {key, data}
  - put/get semantics
- Adds:
  - Transactional control
    - begin, commit, rollback
  - Concurrency control
    - For access from multiple threads
- And:
  - Can use as write-through or write-behind cache
    - by writing a loader plug-in
  - Can share updates with similar stand-alone caches
    - via JMS or HAMgr (in WAS)
    - Asynchronous updates only



## WXS use-case: remote, distributed cache

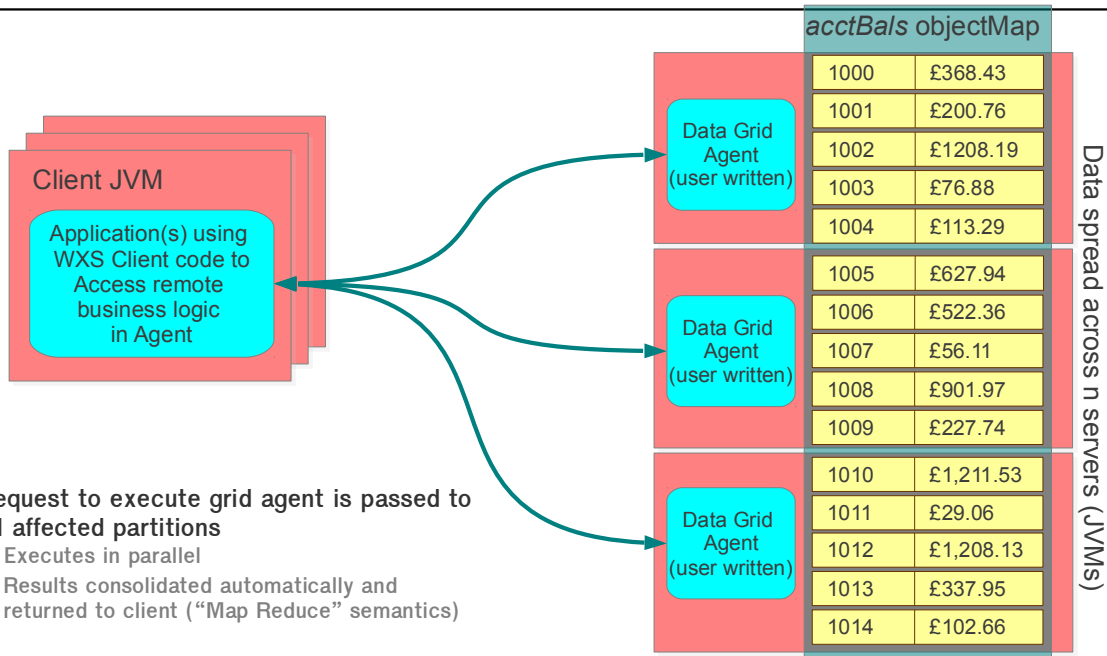
partitioning



- Data is partitioned across the server JVMs
  - WXS clients “know” which server to access based on the key
  - No bottlenecks → scales linearly

## WXS topology: distributed, with grid processing

partitioning



- Request to execute grid agent is passed to all affected partitions
  - Executes in parallel
  - Results consolidated automatically and returned to client ("Map Reduce" semantics)

## Partitions and the Catalog Service

- Partitions define the granularity for placing map entries into particular grid containers (JVMs)
  - You configure the number (n) of partitions to use
  - WXS uses a hashing algorithm to map any key into one of your n partitions
- WXS has then n partitions to place on the available containers (ie JVMs)
  - The fewer containers there are, the more partitions are allocated to each

### The WXS Catalog Service

- Determines the container for each partition
  - Then sends a partition table to all the containers
- Provides first point of contact for client applications
  - And provides clients with partition table so they know where to go for data
- Runs in one or more additional JVMs to manage the grid of container JVMs
- Without the CS, the grid cannot operate
  - So it must be made resilient by having more than one working together
  - But it is not a bottleneck as data accesses/updates do not flow through it

## Resilience and replication

- **Why would you worry about resilience?**
  - In most use-cases data in the grid is only a cache
    - So it can always be recovered from the system of record – e.g. DB etc..
  - But can it be recovered quickly enough?
    - Probably not – otherwise why are you caching it?
- **WXS can automatically maintain *replicas* of data in the grid**
  - A replica of each partition (i.e. the data within that mapped to that partition) is maintained on another container in the grid
  - If the *primary* partition fails, the replica can take over immediately
    - And a new replica is then automatically created asap
  - You can configure:
    - How may replicas (i.e. per partition)
    - Whether replicas are synchronous or asynchronous
    - These are resilience/integrity/performance decisions

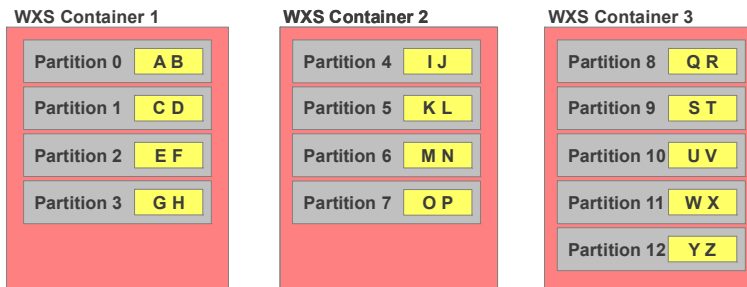
## Horizontal scale-out and scale-in

- **Scale-Out**
  - As new servers are added then shards are migrated from overloaded servers to the new servers.
  - The new server registers with the catalog service
  - The catalog service then starts sending migration instructions to the existing servers.
- **Scale-In**
  - As servers fail then shards are also moved around to balance the system

## Scale-out and scale-in

### • Illustration:

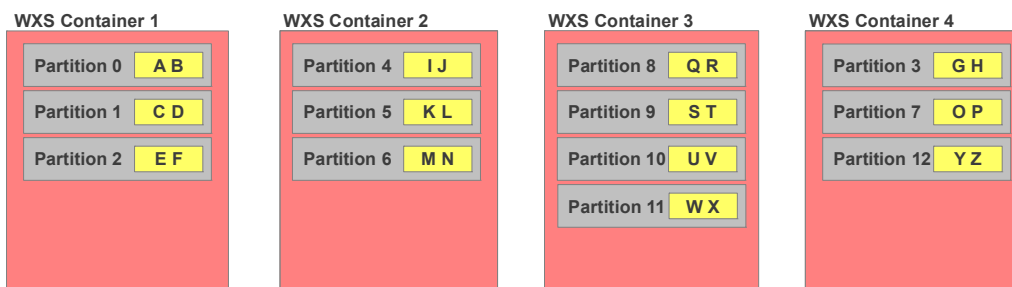
- Key is Customer Name (String) and we've configured 13 partitions
- For simplicity, assume that WXS allocates two initial letters to each partition
  - i.e. keys beginning with "A" and "B" are in the first partition, etc..
- We've configured no replicas
  - So all the shards are primary partitions



## Scale-out and scale-in

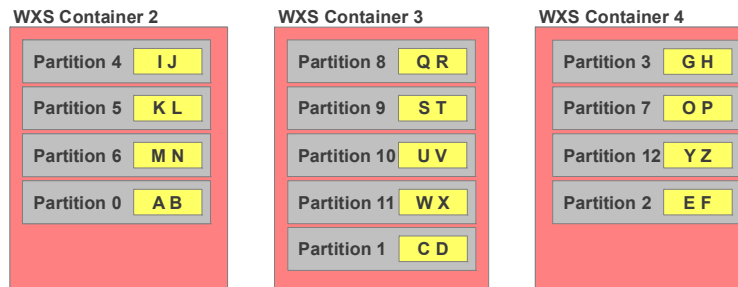
### • Add a container to the grid:

- WXS moves partitions (and the all the data in them) to the new container
  - To keep the spread of shards (partitions & replicas) as even as possible
  - But moves as little as it needs to to achieve this
- This can be at runtime



## Scale-out and scale-in

- **If a container fails:**
  - WXS re-allocates affected partitions on remaining containers
    - Still keeping the spread of shards as even as possible
  - Because we had no replicas, data in those partitions will be lost
    - And need to be reloaded from back-end
    - If we had replicas configured, they would be upgraded to be the new primaries
      - Then new replicas would be allocated to maintain resilience



## WXS Packaging & Integration

- **Installation is using InstallShield (up to V7.1.x)**
  - V8.5 uses Installation Manager instead
- **The core of WXS is just a small number of small JAR files**
- **Does not require WAS ND but integrates well if required:**
  - WXS Containers run as HA'ed services in WAS
  - Catalog Service runs in DMgr or in nodeagents or application server processes
  - WXS service lifecycle managed by WAS (start/stop)
  - WXS can inherit transaction contexts from EJBs etc.
  - PMI/TPV integration
- **Also works with:**
  - WAS base and Community Edition
  - Other application servers
  - Java SE
  - IBM and other JREs
  - Non-Java clients via REST service



## Usage Scenarios (1): side cache

- **Application reads/writes data from/to backend data source**
  - Explicitly stores all or some in a side-cache managed by the application
  - Checks the side cache for required data before reading from back-end
  - Application can pre-load any data expected to be needed to improve performance at runtime
  - Application has full control of access to backend and management of the cache
    - More coding than alternatives but very flexible design

## Usage Scenarios (2): write-through cache

- **Application reads/writes data only from/to WXS cache**
  - All application access is to the cache
  - If data in cache it is read from there
    - If data not there, the loader fetches it from the back-end and caches it
  - Writes and updates written to cache and also synchronously written to the back-end by the loader
    - So back-end and cache always consistent
  - Loader can be designed to pre-load the cache from back-end
  - Once the loader is written, the application has a simpler view of the data

## Usage Scenarios (3): write-behind cache

- **Application reads/writes data only from/to WXS cache**
  - All application access is to the cache
  - If data in cache it is read from there
    - If data not there, the loader fetches it from the back-end and caches it
  - Writes and updates written to cache and but only written to the back-end by the loader in batches, asynchronously
    - So cache is more up-to-date than the back-end – that it's the “system of record”
  - Loader can be designed to pre-load the cache from back-end
  - Once the loader is written, the application has a simpler view of the data
  - Load on the back-end is greatly reduced
    - This design sometimes called a “shock-absorber”

## In-memory DataGrid vs. Database

IMDG	IMDB
Map or Entity model	Relational or Entity (via JPA) model
Limited query capability	Extensive query capability + joins etc..
Data spread evenly over large number of servers	Data held on one server, possibly replicated to another for resilience
Almost unlimited capacity by adding servers	Capacity limited by memory on server
Data partitioned	Data not partitioned
Automatic scale-out/scale-in	Can't scale beyond one server
Near linear scalability	Scalability limited by capacity and speed of server and network
Examples: WXS, Coherence, Gemfire, Terracotta...	Examples: IBM SolidDB, Oracle TimesTen...

# WebSphere eXtreme Scale with WebSphere Application Server Network Deployment

## Installing WXS with WAS ND

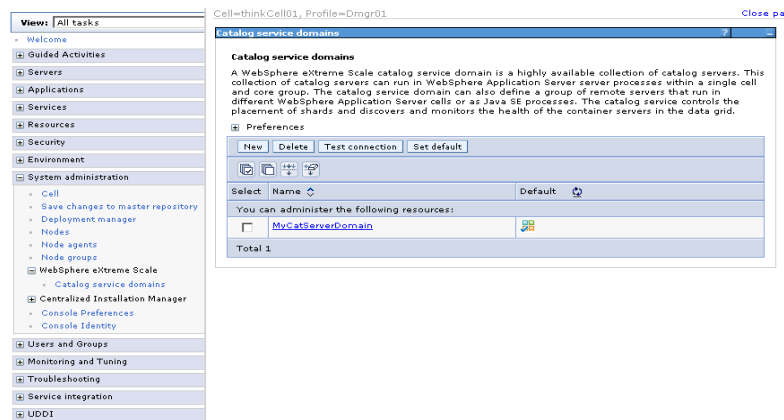
- During WXS install specify WAS directory as target
  - e.g. *c:\Program files\IBM\WebSphere\AppServer*
  - Installer installs WAS version of WXS
- Augment all profiles for cell to use WXS
- At first sight nothing much seems to have changed, but:
  - Look at dmgr log and you'll see a WXS Catalog Service starts there
  - Other WAS processes show various CWOBJ messages
  - A few additions in the Admin Console UI
    - Well hidden!
- At this point you have a working Catalog Service – that's all
  - no grid containers

## Location of Catalog Service

- The dmgr is a bad place for the catalog service
  - There's only one dmgr
    - So it becomes a single point of failure for the grid
  - Installations sometimes re-cycle the dmgr during production
    - This is fine for WAS but will break your XS grid
  - Unwanted and unpredictable mutual impacts
    - E.g. operators do something demanding in Admin Console and grid performance suffers
  - You may want to tune the JVM differently for dmgr and Catalog Service
- There is a better way:
  - Define a *Catalog Service Domain*

## Catalog Service Domain

- Defines a group of servers (> 1) to host the Catalog Service
- This Catalog Service is then resilient
  - because a single server failure won't stop the service
- Look in “WebSphere eXtreme Scale” within “System Administration” in the Admin Console



Cell=thinkCell01, Profile=Dmgr01 Close pa...

**Catalog service domains**

A WebSphere eXtreme Scale catalog service domain is a highly available collection of catalog servers. This collection of catalog servers can run in WebSphere Application Server server processes within a single cell and core group. The catalog service domain can also define a group of remote servers that run in different WebSphere Application Server cells or as Java SE processes. The catalog service controls the placement of shards and discovers and monitors the health of the container servers in the data grid.

Preferences

New Delete Test connection Set default

Select	Name	Default
<input type="checkbox"/>	MyCatServerDomain	

Total 1

## Creating a Catalog Service Domain

Default CS removes need for programmer to specify host & port for CS in code.

If adding an existing app server to contain a CS, you only specify the client port (typically 6601, 2...)

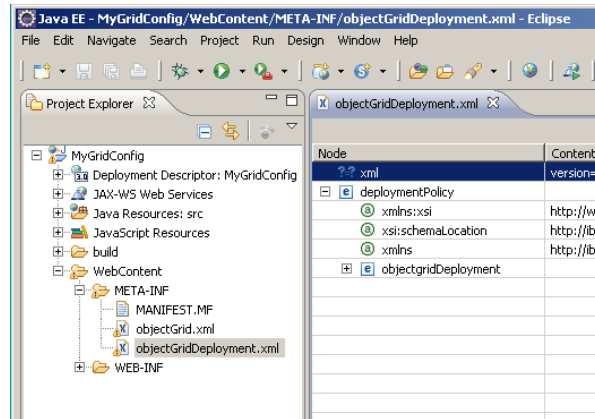
You can add a remote standalone Catalog Server instead. In this case you only specify its Listener Port (typically 2809, etc.)

## Creating grid containers

- There is no provision in the Admin Console for creating or configuring WXS containers
- To do this requires you to deploy a WXS application to one or more WAS servers:
  - If your application is going to use an in-process grid:
    - Just include the ObjectGrid.xml and ObjectGridDeployment.xml files in the WAR file
    - WAS sees these files, and configures a grid using the servers the application is deployed to
  - If you prefer separate grid containers:
    - Create a WAR file with nothing in it except ObjectGrid.xml and ObjectGridDeployment.xml
    - Create a WAS cluster whose members are to host the grid containers
    - Deploy the “empty” WAR file to this cluster
- In both cases, the grid is known to the default Catalog Service
  - So can be accessed by any applications via the CS
- If you're not a programmer, and don't know what Eclipse is:
  - Just create the XML files then ask a friendly programmer to help you package them into a WAR file

## Creating a WAR file to Define your Containers

- Easiest way is to use Eclipse
  - Or an Eclipse-based tool such as Rational Application Developer
- Create a new Dynamic Web Project
- Expand WebContent/META-INF
- Drag/drop or Import the two xml files there
- Export the project as a WAR
- Important:
  - the names of these XML files are significant.
  - This is different from stand-alone WXS where you can call them anything you like



## Creating a WAR file to Define your Containers

- Or, use the Java jar command:
  - Create an empty temporary directory
  - Create a subdirectory called *META-INF*
  - Put the two .XML files in the META-INF directory
  - In the temporary directory issue this command:

```
jar -cvf MyWarFileName.war *
```

```
Directory of C:\temp
06/12/2011 11:16 <DIR> .
06/12/2011 11:16 <DIR> ..
12/10/2011 13:54      712 objectGridDeploymentStandAlone.xml
12/10/2011 13:54      1,299 objectGridStandAlone.xml
                2 File(s)      2,011 bytes
                2 Dir(s)      2,256,904,192 bytes free

C:\temp>md META-INF
C:\temp>mv *.xml META-INF
'mv' is not recognized as an internal or external command,
operable program or batch file.
C:\temp>move *.xml META-INF
C:\temp>objectGridDeploymentStandAlone.xml
C:\temp>objectGridStandAlone.xml

C:\temp>dir
Volume in drive C has no label.
Volume Serial Number is 3449-B8B9

Directory of C:\temp
06/12/2011 11:16 <DIR> .
06/12/2011 11:16 <DIR> ..
06/12/2011 11:16 <DIR> META-INF
                0 File(s)      0 bytes
                3 Dir(s)      2,256,904,192 bytes free

C:\temp>jar -cvf WMSessionGrid.war *
added manifest
ignoring entry META-INF/
adding: META-INF/objectGridDeploymentStandAlone.xml(in = 712) (out = 340)(deflated 52%)
adding: META-INF/objectGridStandAlone.xml(in = 1299) (out = 470)(deflated 63%)
```

## HTTPSession Persistence with WXS

## Session Persistence/Failover

### Why do it?

- If a server process fails, user's HTTPSession can be accessed on another server
  - Apart from a short pause, user isn't affected by the failure
    - Reduced customer dissatisfaction
    - Flexibility to schedule server restarts during working day

### Any drawbacks?

- Persisting the HTTPSession involves:
  - Serialisation/de-serialization costs (CPU, response time)
  - Network load when session updated or retrieved
  - Possibly increased heap usage
    - Depending on which method is used

### Remember

- Requires that all data in HTTP Session is Serializable
  - Not an arbitrary limitation – it's unavoidable

## Options for session persistence

- **Provided by WAS ND as standard:**
  - Persist session to a database
    - Shared by all servers in cluster
  - Persist session to one or more other servers in cluster
  - Persist session to one or more servers in a dedicated cluster
- **Available via WAS admin console if WXS installed:**
  - Use WXS to persist sessions to a grid
    - In process: sharing heaps of existing servers
    - Out of process: in a dedicated grid, within or outside WAS

## WXS Session Persistence

- **Benefits:**
  - Share sessions between:
    - Two or more WAS cells
    - Base WAS instances (i.e. not ND)
    - Different version levels of WAS
    - WAS or WAS ND and WAS Community Edition
    - WAS and a non-IBM application server
    - Non-IBM application servers
  - Possible performance and resilience benefits
    - E.g. reliable replication vs. “best efforts” in standard WAS mechanism
    - Not normally a reason for using WXS for sessions
- **Implemented via a servlet filter**
  - Inserted (“spliced”) into application WAR file
  - Tools provided for non-WAS application servers
  - For WAS it's automatically performed
    - Via the Admin Console or scripting



# Enabling WXS Session Persistence

## 1. In the application, click on *Session management*

**Enterprise Applications > DefaultApplication.ear**  
Use this page to configure an enterprise application. Click the links to access pages for further configuration of its modules.

## 2. Then select *eXtreme Scale session management...*

**Enterprise Applications > DefaultApplication.ear > Session management**  
Use this page to configure session manager properties to control the behavior of Hypertext Transfer Protocol (HTTP) support. These settings apply to both the SIP container and the Web container.

## 3. Panel shows settings for WXS and DataPower XC10

**Enterprise Applications > DefaultApplication.ear > Session management > eXtreme Scale session management settings**  
Configure this application to be associated with eXtreme Scale.

## 4. Enable, then select from pull-down list

# Enabling WXS for Session Persistence

- Choose either embedded or remote data grid

- **Embedded:**
  - Default config sets everything up for you
  - Or you can provide your own config files
- **Remote:**
  - Select Catalog Server domain previously configured
  - Select a grid from those catalogued in that CS domain

# Managing grid defaults for embedded option

- Edit the master copy of “splicer.properties”
  - A custom property defined on the cell gives its location
  - Don't edit copy on nodes etc., as it will be overwritten at next sync
- Also, objectGridDeployment.xml
  - in <appserverroot>\optionalLibraries\ObjectGrid\session\samples

```

# Generated by writeSplicerFile() at Mon Nov 28 12:00:37 GMT 2011
objectGridXML=
sessionTableSize=2000
objectGridName=session
replicationInterval=2
objectGridDeploymentXML=
sessionManagementType=XSEmbeddedSessionManagement
objectGridType=EMBEDDED
fragmentedSession=true
    
```

# Managing grid defaults for embedded option

- Edit the master copy of “splicer.properties”
  - A custom property defined on the cell gives its location
  - Don't edit copy on nodes etc., as it will be overwritten at next sync
- Also, objectGridDeployment.xml
  - in <appserverroot>\optionalLibraries\ObjectGrid\session\samples

If you're testing this on a single machine, don't forget to set **developmentMode=true** in your objectGridDeployment.xml (otherwise no replicas will be created)

```

# Generated by writeSplicerFile() at Mon Nov 28 12:00:37 GMT 2011
objectGridXML=
sessionTableSize=2000
objectGridName=session
replicationInterval=2
objectGridDeploymentXML=
sessionManagementType=XSEmbeddedSessionManagement
objectGridType=EMBEDDED
fragmentedSession=true
    
```

## splicer.properties

- Properties file used by splicer
  - when updating EAR/WAR files to use WXS for session persistence
  - When using the auto-splicing performed by WAS admin console/scripting
- Options described in sample file:

```
<WAS_INSTALL_ROOT>\optionalLibraries\ObjectGrid\session\samples\splicer.properties
```

- Example: `useURLEncoding = true`

## Defining a grid for the remote option

- If grid containers to be within WAS
  - Define a cluster of WAS servers
  - Create a WAR file containing the two xml files:
    - `objectGridStandAlone.xml`
    - `objectGridDeploymentStandAlone.xml`
      - Use eclipse or the command line JAR command
      - These files can be found in the ObjectGrid/session/samples directory
  - Deploy this WAR file to your cluster of containers
- If grid containers to be outside WAS (i.e. standalone WXS)
  - Start some containers using the `startOgServer` command, referencing the two XML config files

## WXS with DynaCache

## Introduction to Dynamic Caching in WAS

- Defined at the container level within WAS
- Intended to cache results of invocations of
  - Servlets
  - JSPs
  - Calls to EJBs
  - Etc...
- Cacheable objects defined in cachespec.xml file
  - Within WEB-INF for a web module
  - Within META-INF for EJBs
- DynaCache can provide dramatic performance gains
  - But needs careful planning when design applications to exploit it fully
- If WXS is installed, DynaCache can be configured to use WXS to hold the cached objects
  - Gives a high-performance, scalable, shared cache across for best results

## Enabling DynaCache with WXS

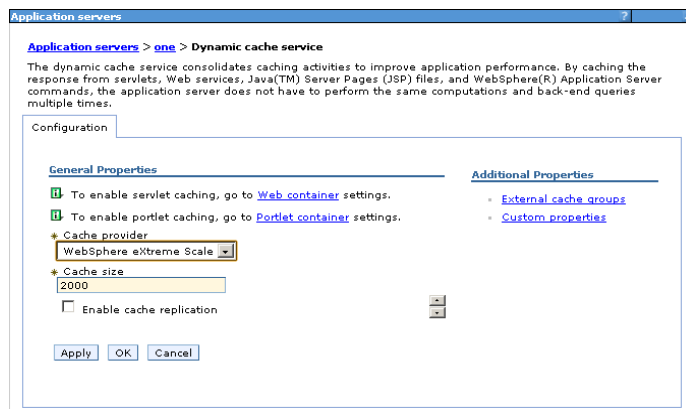
- First enable caching for the server
  - Navigate to the server
  - Then expand *Container Services*
  - Click on *Dynamic Cache Service*



## Enabling DynaCache with WXS

- Then use pull-down to switch provider to select *WebSphere eXtreme Scale*
  - Most of the DynaCache options are no longer relevant so disappear!
- Check *Enable cache replication* checkbox
  - If you don't this will be a cache that is local to this JVM only
  - *On older versions you had to set a custom setting:*

`com.ibm.ws.cache.CacheConfig.enableCacheReplication=true`



## Enabling DynaCache with WXS

- Next, specify the WXS replication topology

`com.ibm.websphere.xs.dynacache.topology`

- Value is one of:

`embedded`, `embedded_partitioned` or `remote`

- If using *embedded*, recommended to specify a number of initial containers

- Specify number slightly less than expected number of containers in this grid

[Application servers](#) > [one](#) > [Process definition](#) > [Java Virtual Machine](#) > [Custom properties](#)

Use this page to specify an arbitrary name and value pair. The value that is specified for the name and value pair is a string that can set internal system configuration properties.

☑ Preferences

Select	Name	Value	Description
You can administer the following resources:			
<input type="checkbox"/>	<a href="#">com.ibm.websphere.xs.dynacache.num_initial_containers</a>	2	Initial WXS containers
<input type="checkbox"/>	<a href="#">com.ibm.websphere.xs.dynacache.topology</a>	embedded	WXS replication topology
Total 2			

## Confirming that WXS is the cache provider

- Restart the server
- then check SystemOut.log to confirm WXS is being used as the cache provider

```
[01/12/11 17:04:52:828 GMT] 00000000 ObjectCacheSe I DYN1056I: Dynamic Cache (object cache) initialized successfully.
[01/12/11 17:04:52:859 GMT] 00000000 CacheProvider I CWOBJ4500I: WebSphere eXtreme Scale Dynamic Cache provider is successfully initialized.
[01/12/11 17:04:53:203 GMT] 00000000 XmlObjectGrid I CWOBJ4701I: Template map IBM_DC_PARTITIONED_* is configured in ObjectGrid DYNACACHE_EMBEDDED.
[01/12/11 17:04:53:203 GMT] 00000000 XmlObjectGrid I CWOBJ4701I: Template map IBM_DC_PARTITIONED_* is configured in ObjectGrid DYNACACHE_EMBEDDED.

.
.
.
[01/12/11 17:04:58:796 GMT] 00000000 ObjectGridImp I CWOBJ4700I: The map name IBM_DC_EMBEDDED_baseCache matched the regular expression of template map
IBM_DC_EMBEDDED_*. The IBM_DC_EMBEDDED_baseCache map has been created for ObjectGrid DYNACACHE_EMBEDDED.
[01/12/11 17:04:58:984 GMT] 0000001a ReplicatedPar I CWOBJ1511I: IBM_SYSTEM_xsastats.server:IBM_SYSTEM_ENTITYMANAGER_MAPSET:0 (primary) is open for business.
[01/12/11 17:04:59:031 GMT] 0000001c ReplicatedPar I CWOBJ1511I: DYNACACHE_EMBEDDED:IBM_SYSTEM_ENTITYMANAGER_MAPSET:0 (primary) is open for business.
[01/12/11 17:04:59:062 GMT] 00000000 CacheProvider I CWOBJ4508I: The WebSphere eXtreme Scale provider has created a Dynamic Cache instance with name
baseCache using topology embedded.
[01/12/11 17:04:59:296 GMT] 00000035 ReplicatedPar I CWOBJ1511I: DYNACACHE_EMBEDDED:DYNACACHE_PARTITIONED:32 (primary) is open for business.
[01/12/11 17:04:59:296 GMT] 00000022 ReplicatedPar I CWOBJ1511I: DYNACACHE_EMBEDDED:DYNACACHE_PARTITIONED:12 (primary) is open for business.
[01/12/11 17:04:59:296 GMT] 00000037 ReplicatedPar I CWOBJ1511I: DYNACACHE_EMBEDDED:DYNACACHE_PARTITIONED:0 (primary) is open for business.
[01/12/11 17:04:59:296 GMT] 00000033 ReplicatedPar I CWOBJ1511I: DYNACACHE_EMBEDDED:DYNACACHE_PARTITIONED:8 (primary) is open for business.

.
.
.
[01/12/11 17:04:59:656 GMT] 00000027 ReplicatedPar I CWOBJ1511I: DYNACACHE_EMBEDDED:DYNACACHE_PARTITIONED:44 (primary) is open for business.
[01/12/11 17:04:59:671 GMT] 0000002a ReplicatedPar I CWOBJ1511I: DYNACACHE_EMBEDDED:DYNACACHE_PARTITIONED:33 (primary) is open for business.
[01/12/11 17:04:59:671 GMT] 00000022 ReplicatedPar I CWOBJ1511I: DYNACACHE_EMBEDDED:DYNACACHE_PARTITIONED:14 (primary) is open for business.
[01/12/11 17:04:59:703 GMT] 0000002f ReplicatedPar I CWOBJ1511I: DYNACACHE_EMBEDDED:DYNACACHE_PARTITIONED:18 (primary) is open for business.
[01/12/11 17:04:59:703 GMT] 00000034 ReplicatedPar I CWOBJ1511I: DYNACACHE_EMBEDDED:DYNACACHE_PARTITIONED:39 (primary) is open for business.
[01/12/11 17:04:59:718 GMT] 00000033 ReplicatedPar I CWOBJ1511I: DYNACACHE_EMBEDDED:DYNACACHE_PARTITIONED:5 (primary) is open for business.
[01/12/11 17:04:59:718 GMT] 00000020 ReplicatedPar I CWOBJ1511I: DYNACACHE_EMBEDDED:DYNACACHE_PARTITIONED:28 (primary) is open for business.
[01/12/11 17:04:59:734 GMT] 00000031 ReplicatedPar I CWOBJ1511I: DYNACACHE_EMBEDDED:DYNACACHE_PARTITIONED:37 (primary) is open for business.
[01/12/11 17:05:00:906 GMT] 00000000 JvmMemoryFull I CWOBJ4542I: Basic BackingMap memory sizing is enabled.
[01/12/11 17:05:00:921 GMT] 00000000 ObjectGridImp I CWOBJ4700I: The map name IBM_DC_EMBEDDED_baseCache matched the regular expression of template map
IBM_DC_EMBEDDED_*. The IBM_DC_EMBEDDED_baseCache map has been created for ObjectGrid DYNACACHE_EMBEDDED.
[01/12/11 17:05:00:984 GMT] 00000000 ServerCache I DYN1001I: WebSphere Dynamic Cache instance named baseCache initialized successfully.
[01/12/11 17:05:00:984 GMT] 00000000 ServerCache I DYN1071I: The cache provider "com.ibm.ws.objectgrid.dynacache.CacheProviderImpl" is being used.
[01/12/11 17:05:01:093 GMT] 00000000 ServletCacheS I DYN1055I: Dynamic Cache (servlet cache) initialized successfully.
```

## DynaCache with WXS

- This process is much simpler than it used to be:
  - Before WAS 7 + WXS 7.1
  - Custom settings were required
    - To specify cache replication
    - To specify location of catalog servers
- This all assumes you know how to exploit DynaCache in your application and to configure the DynaCache specific options
  - For example, that you have turned on Servlet or Object caching in the server
- The steps described are the minimum to configure the WXS DynaCache provider
  - In some cases, it will be more complex
  - See the WAS and WXS InfoCenters for more details
    - WAS:
      - *Configuring dynamic cache (DynaCache) to use the WebSphere eXtreme Scale dynamic cache provider*
    - WXS:
      - *Configuring the dynamic cache provider for WebSphere eXtreme Scale*
    - Also refer to Redbook:
      - *Scalable, Integrated Solutions for Elastic Caching Using IBM WebSphere eXtreme Scale (SG24-7926)*
        - Section 6.3: *How to integrate eXtreme Scale with WebSphere Commerce* has the best description of how to set up WXS with DynaCache.

## Benefits of using WXS with Dynacache

- Optimise site-recovery times
  - Improve startup performance under load
  - Minimise cache warm-up delays/load
- Improve cache currency
- Reduce load on back-end, DB, network
  - As data cached once, rather than once per server
- Reduce/eliminate need for Dynacache disk offload
  - And need to tune this
- Greatest benefit:
  - Where large number of WCS JVMs in use/large clusters
  - Where frequent cache invalidations are needed (e.g. for promotions)

## More information

- ***Enhancing WebSphere Commerce performance with WebSphere eXtreme Scale***
  - [http://www.ibm.com/developerworks/websphere/techjournal/1008\\_genkin/1008\\_genkin-pdf.pdf](http://www.ibm.com/developerworks/websphere/techjournal/1008_genkin/1008_genkin-pdf.pdf)
- ***Scalable, Integrated Solutions for Elastic Caching Using IBM WebSphere eXtreme Scale (SG24-7926)***
- ***WebSphere eXtreme Scale Best Practices for Operation and Management (SG24-7964)***
- ***Users Guide to WebSphere eXtreme Scale (SG24-7683)***

## Summary and Close



## Topics covered

- What is WebSphere eXtreme Scale?
- Simple topologies
- Architectural components:
  - Partitioning and replication
  - Scale-out and scale-in
  - The Catalog Service
- Configuring WXS with WebSphere Application Server
- Implementing WAS/WXS Use Cases

## Finally...



- I am available for training or consultancy on:
  - WebSphere eXtreme Scale
  - WebSphere Application Server
  - WebSphere Virtual Enterprise
- I offer a two-day hands-on technical training course on WXS installation and administration
- I work independently and through:
  - Orbital Integrated Solutions
  - Verhoef Training Ltd
  - Diegesis Ltd



IBM Partner Marketing Platinum Award 2012!!



## Finally...

- I am available for training or consultancy on:
  - WebSphere eXtreme Scale
  - WebSphere Application Server
  - WebSphere Virtual Enterprise
- I offer a two-day hands-on technical training course on WXS installation and administration
- I work independently and through:
  - Orbital Integrated Solutions  IBM Partner Marketing Platinum Award 2012!!
  - Verhoef Training Ltd 
  - Diegesis Ltd 