JBossCache and JBoss Clustering technologies: Providing the High Availability solution in the Microsoft Cluster Environment

Kurt Chou
Northrop Grumman
Army SEC (Software Engineer Center)

Overview Agenda

- DDS (Data Dissemination Services)
  - ✓ Why DDS needs High Availability feature
- Clustering Technologies overview
  - ✓ What is HA (High Availability), Load-balancing, Fail-over, and Fault tolerance.
- DDS HA Design
  - ✓ Requirement
  - ✓ The different between J2EE Clustering and MS cluster
- DDS HA Implementation
  - ✓ How to use the JBossCache and JGroups to implement DDS HA feature
- Demo
  - ✓ HA Simulation
  - ✓ JBossCache CacheLoader
- Q&A

DDS Overview

- SOAP Messaging Application
  - ✓ Advertise
  - ✓ Subscribe
  - ✓ Publish
- DDS nodes inter-communicate SOAP messages among DDS nodes
- Application
  - ✓ Hosted on JBOSS
  - ✓ Developed in Java
  - ✓ Open standards. SOAP, WS-Security etc.
  - ✓ Uses MS windows services or open source
    - MS UDDI or JUDDI
    - MS AD or OpenLDAP
    - SQL Server 2005, Oracle 10g or eXist
  - ✓ Web Based management console
  - ✓ High Availability thru JBOSS clustering
  - ✓ Source provided for COI (Communities Of Interest) upon request
- Client
  - ✓ Built on OSGI (Open Services Gateway initiative) framework
  - ✓ Plugin architecture. Similar to Eclipse.
- Client Libraries
  - ✓ C# and Java libraries available

DDS Objective

- Ensuring data is visible, available, and usable when and where it is needed to accelerate decision-making
- Tagging all data with metadata to enable users to discover the data
- Posting all data to shared spaces, providing access to all users except when limited by security or policy regulations
- Advancing the DoD from defining interoperability through point-to-point interfaces to enabling the "many-to-many" exchanges typical of a net-centric data environment
- Provide an information sharing mechanism that would be useful for *any* COI (Communities Of Interest)

DDS Components - 1

- DDS Node. DDS nodes can be distributed throughout the Global Information Grid (GIG). DDS nodes inter-communicate SOAP messages among DDS nodes to support the services described below:
  - ✓ **Advertisement Propagation** – Advertisements are propagated throughout the DDS.
  - ✓ **Subscription Propagation** – DDS supports federated subscriptions by propagating the subscription request to the "appropriate" DDS nodes.
  - ✓ **Query Propagation** – DDS supports federated queries by propagating the query request to the "appropriate" DDS node.
- Publisher Node. Producers are publisher nodes in DDS. Publisher nodes perform the following:
  - ✓ **Advertise** – A producer can make an information resource visible to the enterprise by advertising it using a DoD Discovery Metadata Standard (DDMS) description of the information resource.
  - ✓ **Unadvertise** – A producer can retract an advertisement, making the information resource invisible to the enterprise.
  - ✓ **Publish** – A producer can post their information to a virtual shared space.
  - ✓ **Retract an Item** – A producer can retract an item that was previously published.
  - ✓ **Request Status** – A producer can request the state/status of a DDS node.
  - ✓ **DDS will store data for archival purposes. <u>This feature is not currently implemented.</u>**

DDS Components - 2

- **Subscriber Node**. Consumers are subscriber nodes in DDS. Subscriber nodes perform the following:
  - ✓ **Subscribe** – A consumer can submit an information subscription. Filter criteria (e.g., geographic region, content filter, or DDMS qualifiers) can be associated with an information request to filter or refine the information request. DDS will asynchronously deliver information that meets the subscription request to the consumer.
  - ✓ **Unsubscribe** – A consumer can retract an information subscription. This removes the information subscription from DDS and stops delivery of this information to the consumer.
  - ✓ **Query** – A consumer can submit an information query. Filter criteria (e.g., geographic region, content filter, or DDMS qualifiers) can be associated with an information request to filter or refine the information request. DDS will synchronously deliver information that meets the subscription request to the consumer.
  - ✓ **Request Status** – A Subscriber can request the state/status of a DDS node.
- **Other services/capabilities**:
  - ✓ ***Subscription Consolidation*** – DDS optimizes information delivery by consolidating "similar" subscription requests.
  - ✓ ***XML Data Agnostic*** – The DDS *architecture* is not tied to any particular COI data model and can distribute information based on any COI's data model.
  - ✓ ***DoD Discovery Metadata Specification*** *(DDMS)* – DDS uses the DDMS to describe the meta-data (e.g., organization information, geographic coverage) about an information source.
  - ✓ ***NCES (Net-Centric Enterprise Services) CES (Core Enterprise Services) Integration*** – DDS nodes and clients utilize NCES services whenever possible

Technologies Overview -1

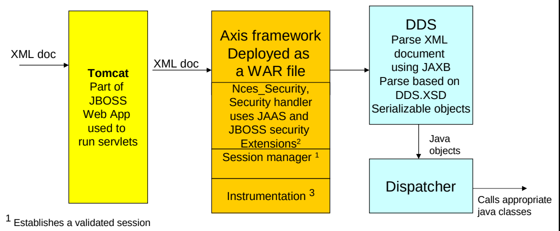
- Clustering Terminology
  - ✓ High availability (HA)
    - Services are continuously operational for a desirably long length of time. Availability can be measured relative to "100% operational." For Examples, (99.999 %) is High-Availability, (99.9999 %) is Very-High-Availability, and (99.99999 %) is Ultra-Availability.
  - ✓ Load Balancing
    - A way to distribute the amount of work between two or more computers so that more work gets done in the same amount of time.
  - ✓ Failover
    - Process can continue when it is re-directed to a "backup" node because the original one fails
  - ✓ Fault tolerance
    - A service that guarantees strictly correct behavior despite system (hardware/software) failure.

Technologies Overview -2

- High Availability vs. Fault Tolerance
  - ✓ HA can be Fault Tolerance, if reliability rate is 100% guaranteed.
- Load Balancing != Fault Tolerance or HA
  - ✓ Load balancing is used for scalability, not for fault tolerance
  - ✓ Failover and state replication are used for fault tolerance

DDS High Availability -1

- DDS Application Flow


XML doc
Tomcat
Part of JBOSS Web App used to run servlets
XML doc
Axis framework Deployed as a WAR file
Nces_Security, Security handler uses JAAS and JBOSS security Extensions[2]
Session manager [1]
Instrumentation [3]
DDS
Parse XML document using JAXB
Parse based on DDS.XSD
Serializable objects
Java objects
Dispatcher
Calls appropriate java classes

[1] Establishes a validated session

[2] Tailored dependent on app server. Denotes which WAR modules require security

[3] Expandable to provide instrumentation tools for validating schemas

DDS High Availability -2

- Requirements
  - ✓ Open Source, Open Standard
  - ✓ Run Web Application Server with J2EE Compliance
  - ✓ Maintain Existing Software Architecture
  - ✓ Performance impact
  - ✓ Default Development Environment – Microsoft Cluster
  - ✓ Transparent to Users

DDS High Availability -3

- Design Consideration
  - ✓ HA-JNDI vs. JBossCache
  - ✓ AOP (Aspect-Oriented Programming) vs. POJO (Plain Old Java Object)
  - ✓ Core vs. Layer
  - ✓ MS Cluster HA vs. J2EE Clustering (Load Balancing)
  - ✓ JBoss Sync vs. Async Replication
- Selection
  - ✓ JBossCache, POJO, Layer
  - ✓ JBoss Async Replication
  - ✓ MS Cluster HA
  - ✓ J2EE Clustering is Optional

DDS High Availability -4

- What is JBossCache?
  - ✓ An in-memory replicated transactional, persistent, and fine-grained cache library
  - ✓ Ideal for state replication
    - Transactional
    - Persistent (passivation)
    - Object-based fine-grained (field level) replication (PojoCache)
  - ✓ Can be used directly by applications, not just by high-level JBoss services
  - ✓ Uses JGroups as the messaging and group membership management layer

DDS High Availability -5

- What is JGroups?
  - ✓ A reliable group messaging library
  - ✓ TCP or Reliable multicasting for intra-group communications
    - TCP is reliable, but needs a lot of bandwidth
    - Reliable Multicasting – Class D, NACK (Negative Acknowledge)
  - ✓ Reliable messaging
    - Message sequence ordering
    - Flow Control
    - Negative Acknowledgement
    - Fragmentation
  - ✓ Manage group membership
    - Join
    - Leave
    - Shun
  - ✓ Highly configurable protocol stacks (e.g., UDP/TCP, GMS, FC, FD, MERGE, etc)

13

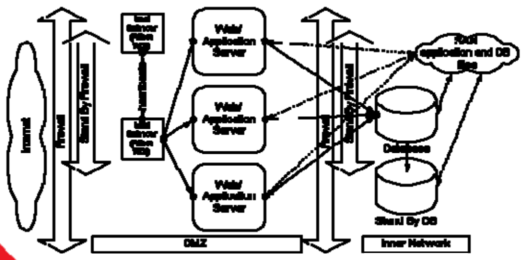
DDS High Availability -6

- Sync vs. Async Replication
  - ✓ Synchronous replication
    - Uses the same thread as the user's request thread
    - Response not completed until *all* servers acknowledge session is replicated
    - If user fails over, the replicated session data will be there
    - Significantly increases response times
    - In most cases not really necessary
  - ✓ Asynchronous replication
    - Replication occurs in a separate thread *but* the message is still guaranteed to be delivered
    - Faster application response times
    - If user fails over within millisecond, replicated session may be there yet
    - Failover is not a likely event; is risk acceptable?

14

DDS High Availability -7

- J2EE Clustering Architecture



15

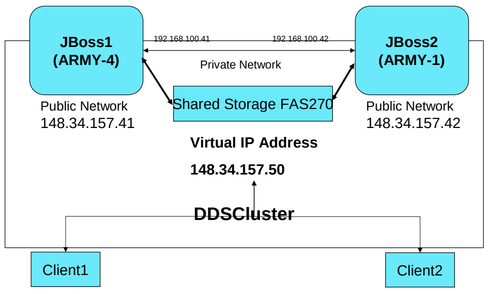
DDS High Availability -8

- Microsoft Cluster
  - ✓ A group of independent servers working collectively via Microsoft Cluster Service (MSCS) software, such as MS exchange and MS SQL
  - ✓ High availability, failover, scalability, and manageability for resources and applications
  - ✓ Uninterrupted client access to applications and server resources in the event of failures and planned outages
  - ✓ Hardware configuration
    - A sharing storage device (SAN or SCSI)
    - At least two network cards
      - ✓ Internal private IP address
      - ✓ Virtual IP address for public

16

DDS HA Implementation -1

- MS Cluster – Development Environment



17

DDS HA Implementation -2

- Install JBoss AS 4.0.2 and Install DDS.war under server/default folder.
  - ✓ Not under server/all folder, the JBoss Clustering server
- Run on MS Cluster Environment
  - ✓ No MS Cluster Awareness API
  - ✓ Active/Standby
- TreeCacheListener – extended AbstractTreeCacheListener

18

DDS HA Implementation -3

- Async Replication
  - ✓ Significantly faster vs. Sync replication
- CacheLoader
  - ✓ File Cache Loader
    - Can be shared, no performance reduction
  - ✓ JDBC Cache Loader (SQL 2005)
    - Can be shared, no performance reduction
- Encryption
  - ✓ sym_algorithm=“Blowfish”
  - ✓ Uses JBossCache-1.2.4 jgroups.jar and jboss-cache.jar, avoid 100% CPU Usage.
- Runs on JBoss & JOnAS
  - ✓ May need to modify some *.xml files

19

DDS HA Implementation -4

- ✓ **public void nodeModify(Fqn fqn, boolean pre, boolean isLocal)**
  - **Add a new adv, subs, pubs jaxb objects**

```
public static class Listener extends AbstractTreeCacheListener {
    :
   public void nodeModify(Fqn fqn, boolean pre, boolean isLocal)
     if (isLocal == false && pre == false){  //not local and after
     String fqstr = fqn.toString().trim();
     if (fqstr.contains("Active") || fqstr.contains("Standbys")) {
      JBCverifyState();
     } else if (fqstr.contains("/DDS/Advertise")) {
         int pos = fqstr.lastIndexOf("/");
         String guidstr = fqstr.substring(pos+1);
         if (guidstr != null) {
            Fqn f= Fqn.fromString(CLUSTERING_Advertise +"/" + guidstr.trim());
            Advertise adv = (Advertise) cache.get(f,"Adv");
            if (adv != null) {
                Response resp = (Response) cache.get(f,"Resp");
      dds.CLadvertise(GUID.create(guidstr),adv,resp,0);
            }
      } else if (fqstr.contains("/SSDDS/Advertise")) {
    :
}
```

20

DDS HA Implementation -5

- ✓ **public void nodeRemove(Fqn fqn, boolean pre, boolean isLocal)**
  - **Delete a jaxb object**

```
public void nodeRemove(Fqn fqn, boolean pre, boolean isLocal) {
        if (isLocal==false &&  pre==false){  //not local and after
             String fqstr = fqn.toString().trim();
             if (fqstr.contains("/DDS/Advertise")) { // remove
          Advertise
                int pos = fqstr.lastIndexOf("/");
                String guidstr = fqstr.substring(pos+1);
                if (guidstr != null) removeAdvertisement(guidstr);
            } else if (fqstr.contains("/DDS/Subscribes")) {
                int pos = fqstr.lastIndexOf("/");
       :
       :
      }
```

21

DDS HA Implementation -6

- ✓ **public void viewChange(View view)**
  - **Active/Standby**

```
public void viewChange(View view) {
       int size = 0;
       Vector v =view.getMembers();
       if (CacheType.equalsIgnoreCase("JBossCache"))
             size = (v.size() + 1) / 2;
       else
             size = v.size();
       if(State) { //Active
             if (Standbys == 0) Standbys = size -1; // discount Active
             else JBCverifyState();
       } else { // swith to active
             if (size <= 1) cache.remove(MSActiveFqn,"localHostname");
             JBCverifyState();
       }
}
```

22

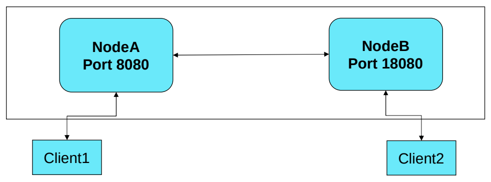
Preliminary Performance

- Average creation time
  - ✓ Advertisement
    - 100 MB (Local domain) – 155 Millisecond per transaction
    - 1 GB (Labs) – 16 Millisecond per transaction
  - ✓ Subscription
    - 100 MB – 920 Millisecond per transaction
    - 1 GB – 90 Millisecond per transaction
  - ✓ Published items
    - 100 MB – 12 Millisecond per transaction
    - 1 GB – 1 Millisecond per transaction
- No impact with CacheLoader features with file persistence or Sql 2005 persistence, because the JBoss uses the different thread
- Some impact with JGroups encryption

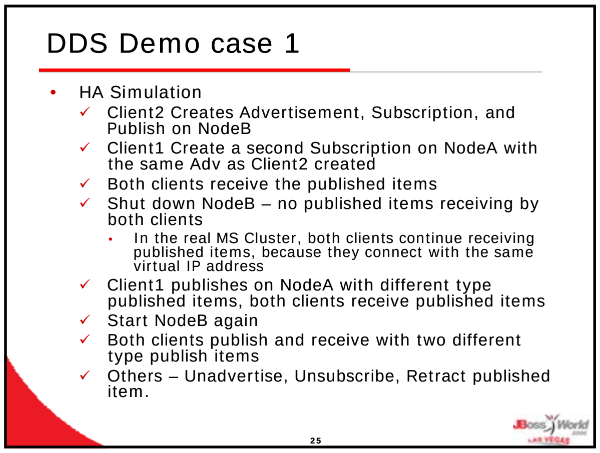
23

DDS Demo Environment

- Not using Microsoft Cluster Environment
- Uses Load-Balancing to manually simulate High Availability case
- Two JBoss Instances: NodeA (Local Port 8080) and NodeB (port 18080)



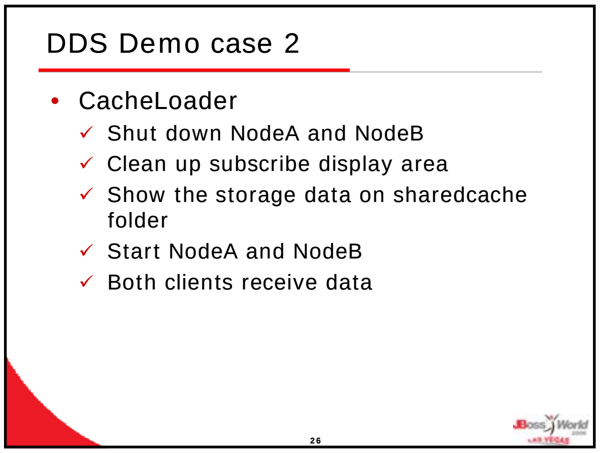
24



DDS Demo case 1

- HA Simulation
  - ✓ Client2 Creates Advertisement, Subscription, and Publish on NodeB
  - ✓ Client1 Create a second Subscription on NodeA with the same Adv as Client2 created
  - ✓ Both clients receive the published items
  - ✓ Shut down NodeB – no published items receiving by both clients
    - In the real MS Cluster, both clients continue receiving published items, because they connect with the same virtual IP address
  - ✓ Client1 publishes on NodeA with different type published items, both clients receive published items
  - ✓ Start NodeB again
  - ✓ Both clients publish and receive with two different type publish items
  - ✓ Others – Unadvertise, Unsubscribe, Retract published item.

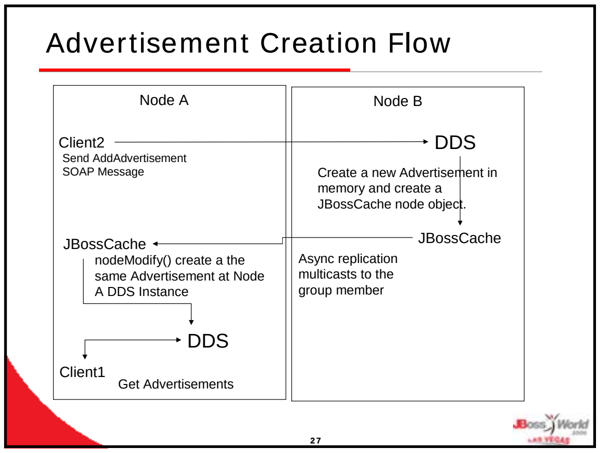
25



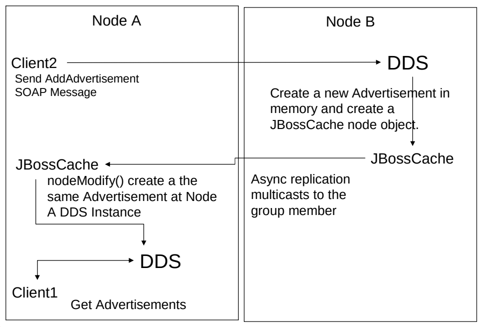
DDS Demo case 2

- CacheLoader
  - ✓ Shut down NodeA and NodeB
  - ✓ Clean up subscribe display area
  - ✓ Show the storage data on sharedcache folder
  - ✓ Start NodeA and NodeB
  - ✓ Both clients receive data

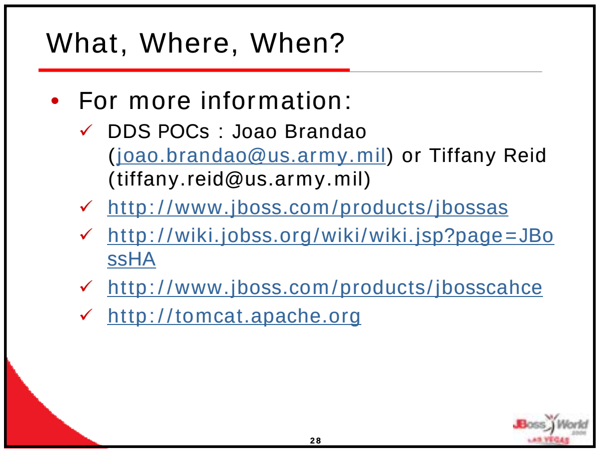
26



Advertisement Creation Flow


Node A
Node B
Client2
Send AddAdvertisement SOAP Message
DDS
Create a new Advertisement in memory and create a JBossCache node object.
JBossCache
JBossCache
Async replication multicasts to the group member
nodeModify() create a the same Advertisement at Node A DDS Instance
DDS
Client1
Get Advertisements

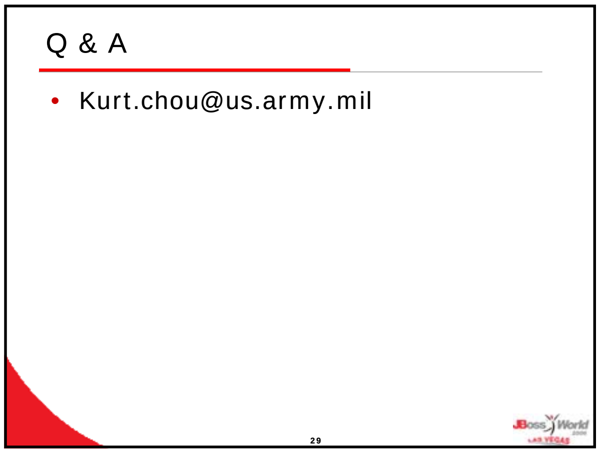
27



What, Where, When?

- For more information:
  - ✓ DDS POCs : Joao Brandao (joao.brandao@us.army.mil) or Tiffany Reid (tiffany.reid@us.army.mil)
  - ✓ http://www.jboss.com/products/jbossas
  - ✓ http://wiki.jobss.org/wiki/wiki.jsp?page=JBossHA
  - ✓ http://www.jboss.com/products/jbosscahce
  - ✓ http://tomcat.apache.org

28



Q & A

- Kurt.chou@us.army.mil

29