JBoss Clustering Technologies

Dimitris Andreadis

Software Engineering Manager
JBoss Application Server
JBoss, by Red Hat

December 1st, 2009

dimitris@redhat.com

Agenda

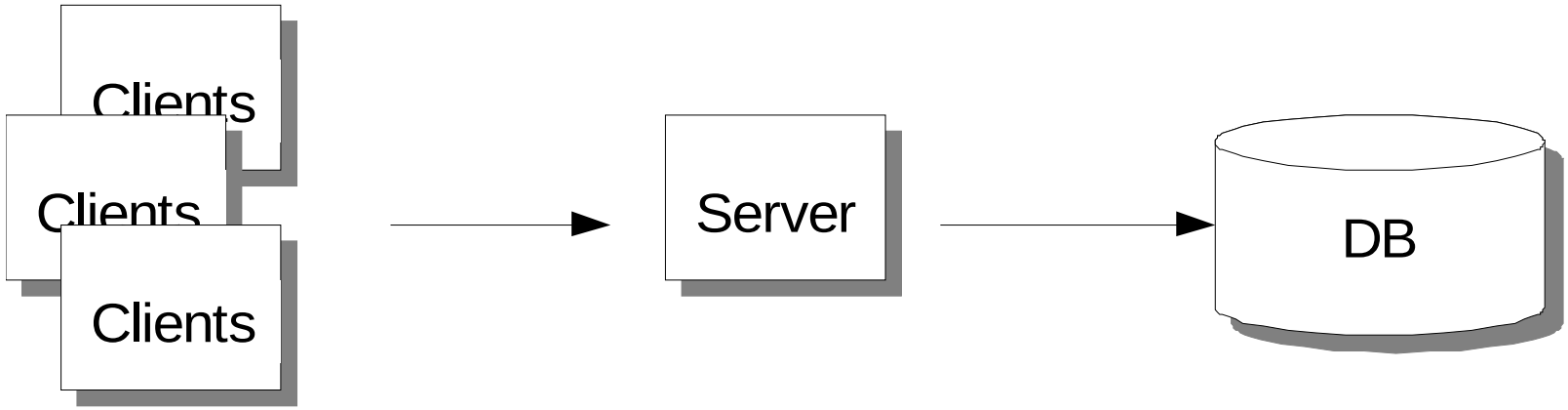
- Who needs Clustering?
- Clustering Terminology
- Clustering in JBoss Application Server
- Clustering infrastructure – JGroups
- Data Grids - Infinispan
- Conclusions

dimitris@redhat.com

Who Needs Clustering?

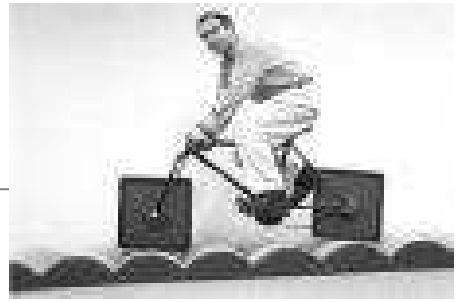
dimitris@redhat.com

Typical System Architecture



dimitris@redhat.com

Πανικός για τα εισιτήρια της Eurovision



- "οι servers της TICKETSTREAM δέχθηκαν ένα πολύ μεγάλο αριθμό επισκέψεων από σήμερα το πρωί στις 10:00 πμ, στην ιστοσελίδα πωλήσεων εισιτηρίων της Eurovision. Αποτέλεσμα αυτού, ήταν η υπερφόρτωση του συστήματος και αδυναμία πρόσβασης στις σελίδες του συστήματος στον κεντρικό server. Οι τεχνικοί της TICKETSTREAM εργάζονται για την επίλυση του προβλήματος. Παρακαλείται το κοινό να επιστρέψει και να προσπαθήσει να αγοράσει εισιτήρια αργότερα".

dimitris@redhat.com

Who Needs Clustering?

- Is your system designed 24x7?
- How many concurrent users it can handle?
- How much does it cost when the system is down?
- Can you afford to loose session data?

dimitris@redhat.com

Clustering Terminology

dimitris@redhat.com

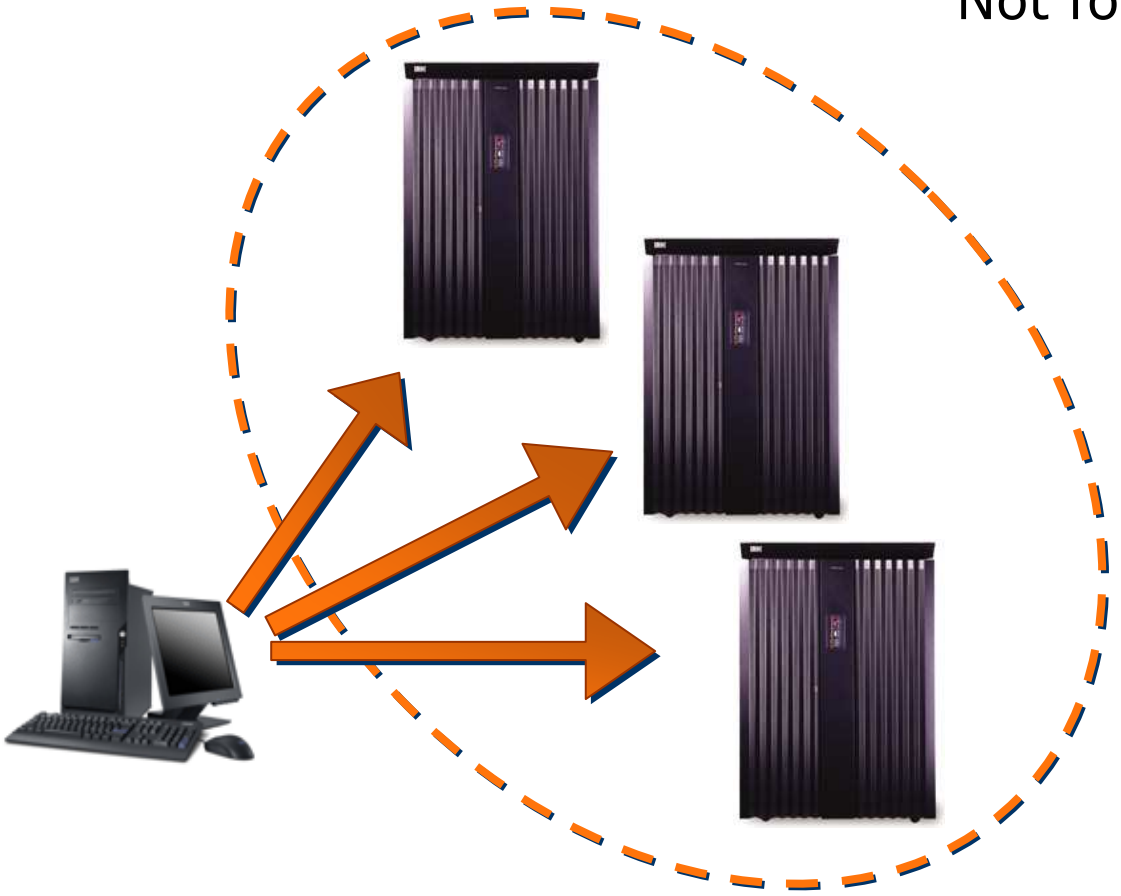
Clustering Terminology

- High Availability
  - ✓ Services are accessible with reasonable (and predictable) response times at any time
    - E.g., 99.999% (5 Nines in Telco -> 5min/year downtime)
  - ✓ No single point of failure
- Scalability
  - ✓ I want to handle x times the number of concurrent access than what I have now
- Fault tolerance
  - ✓ A service that guarantees strictly correct behavior despite system failure

dimitris@redhat.com

Load Balancing is used for scalability

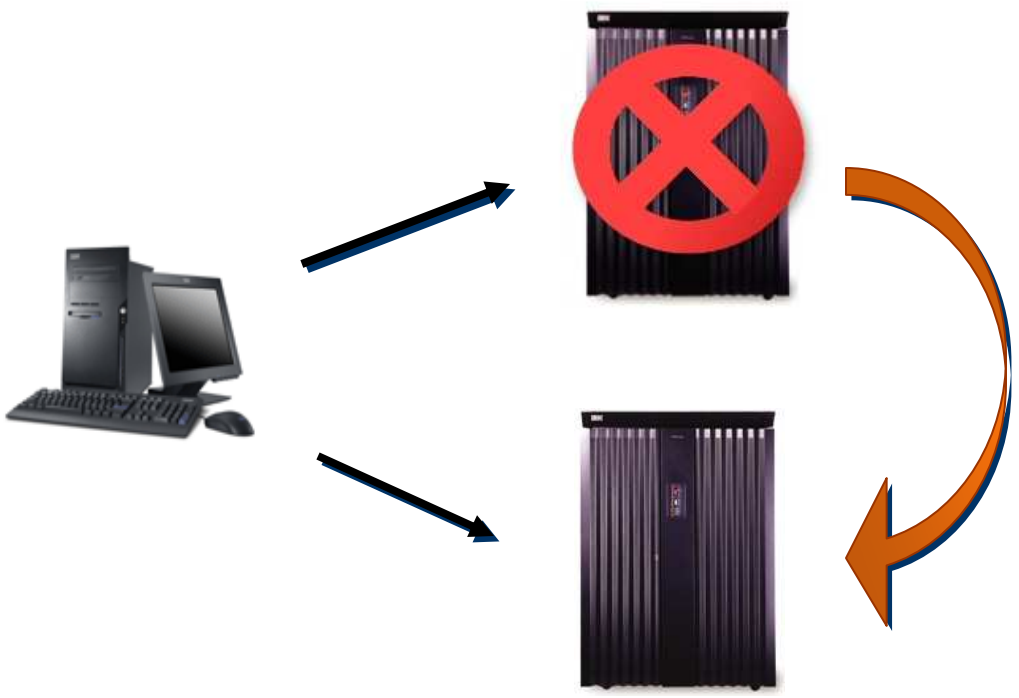
Not for Fault Tolerance!



dimitris@redhat.com

Failover & state replication is used for Fault Tolerance

State Replication has a cost!

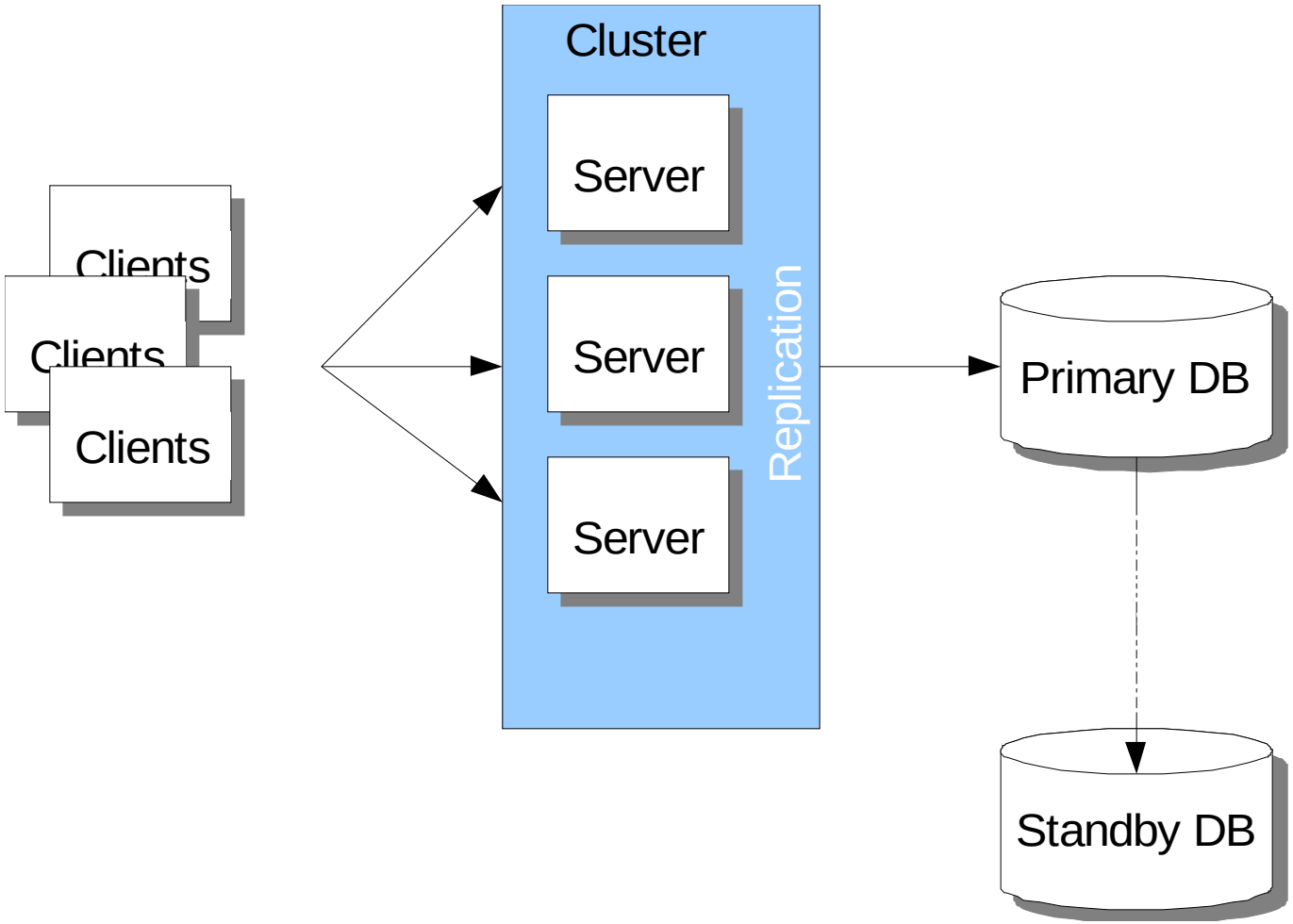


State Replication

Synchronous or Asynchronous

dimitris@redhat.com

Typical Clustered Architecture



dimitris@redhat.com

Clustering in JBoss Application Server

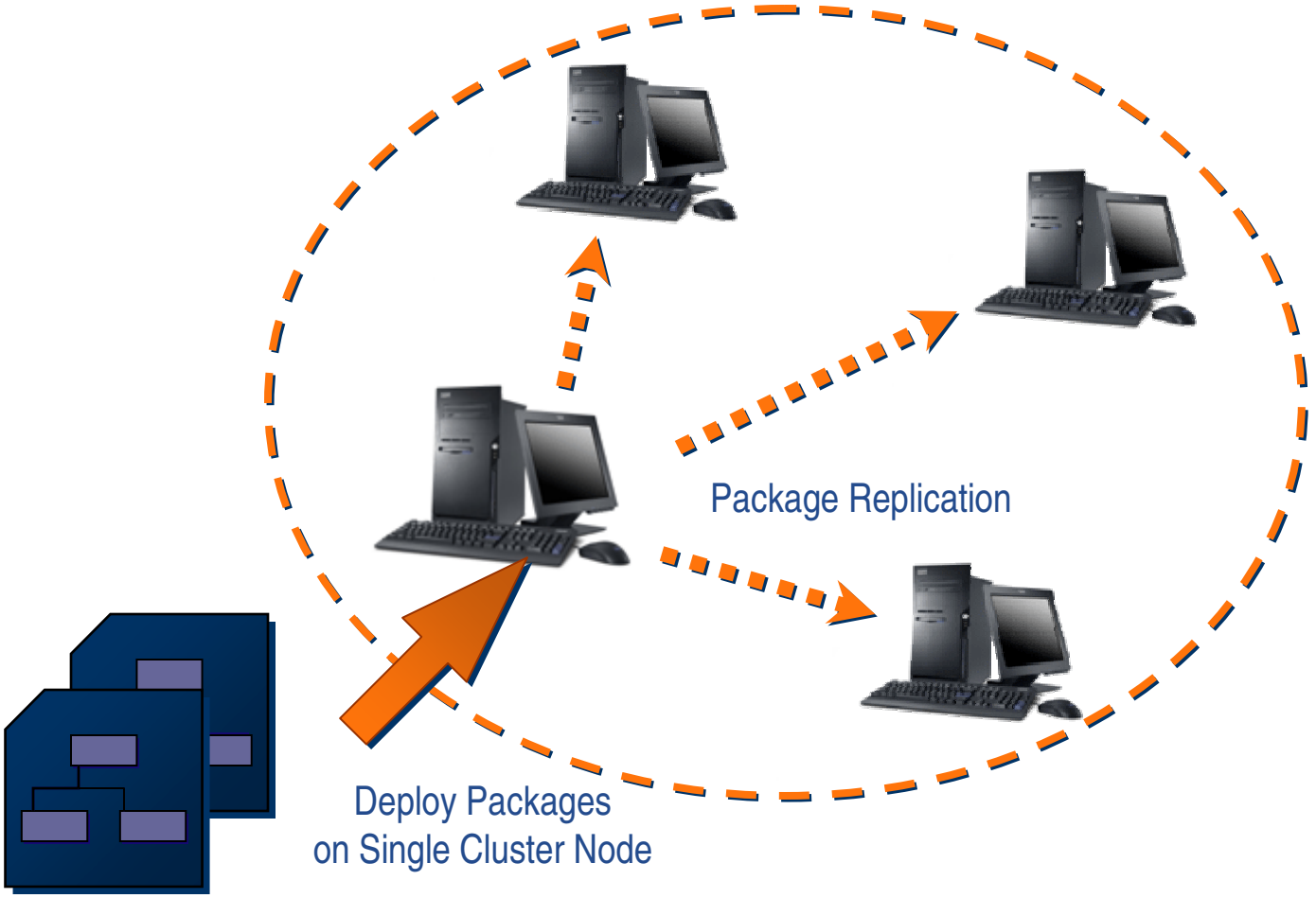
dimitris@redhat.com

Setting up Clustering

- Just start JBoss in the “all” configuration
  - ✓ run.bat -c all
  - ✓ ./run.sh -c all
- JBoss clusters are dynamic
  - ✓ Cluster node detects existing group members automagically
  - ✓ No configuration of topology needed.

dimitris@redhat.com

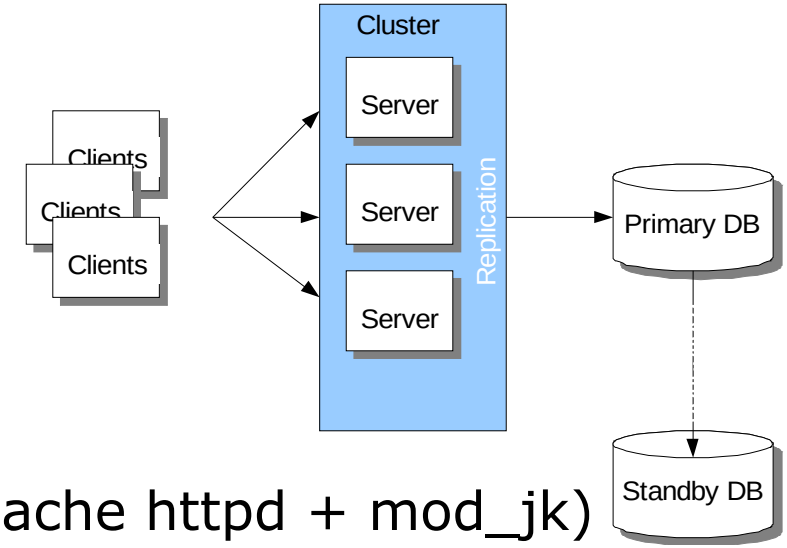
Cluster Manager: Farming



dimitris@redhat.com

What clustering means for the application?

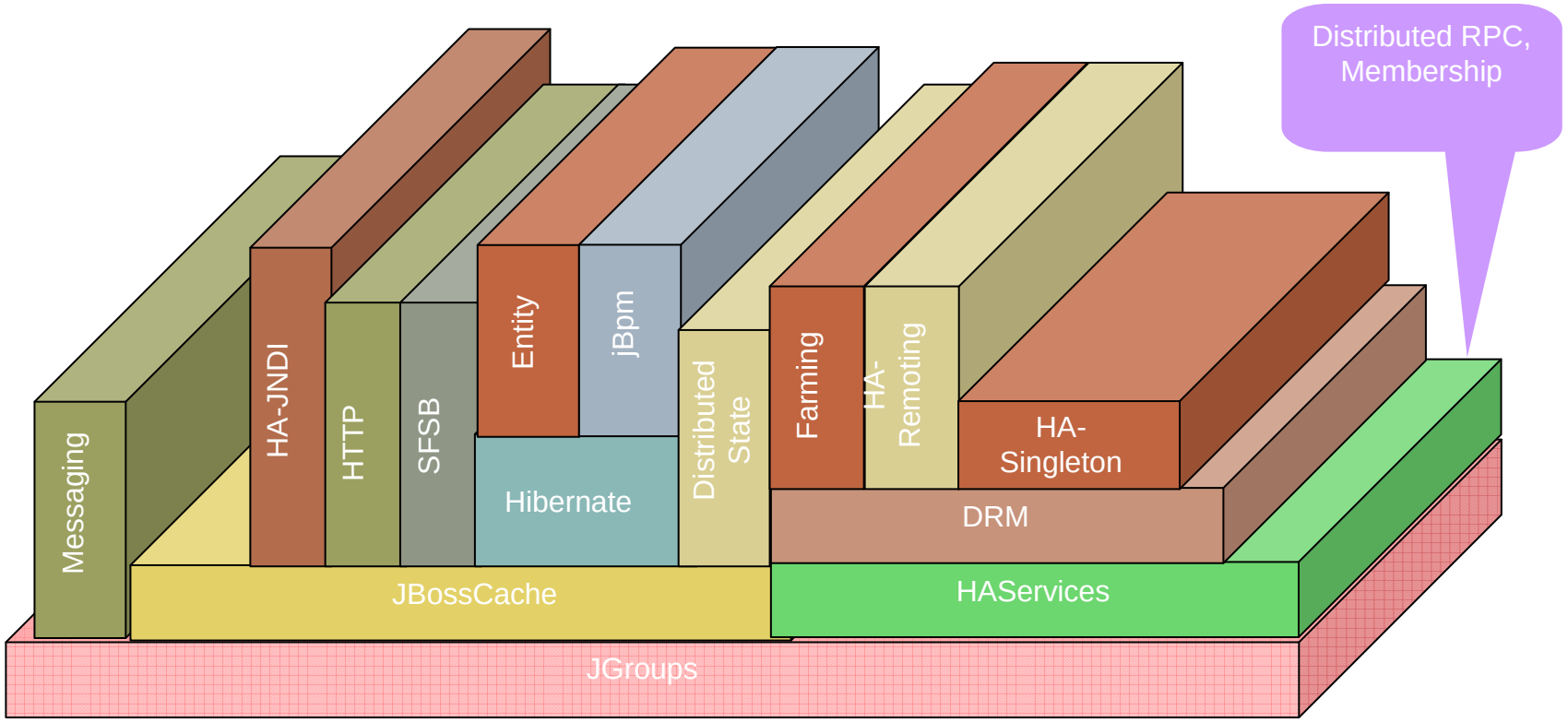
- "Fat" Java Clients
  - ✓ Shared naming space
  - ✓ Statefull bean replication
  - ✓ Fail-over



- Thin Http Clients
  - ✓ Http session replication
  - ✓ Load-balancer settings (e.g. apache httpd + mod_jk)
- Data Access
  - ✓ Query caching (Replication Cache)
  - ✓ Entity caching (Invalidation Cache)
- Singleton Services

dimitris@redhat.com

Clustering Building Blocks



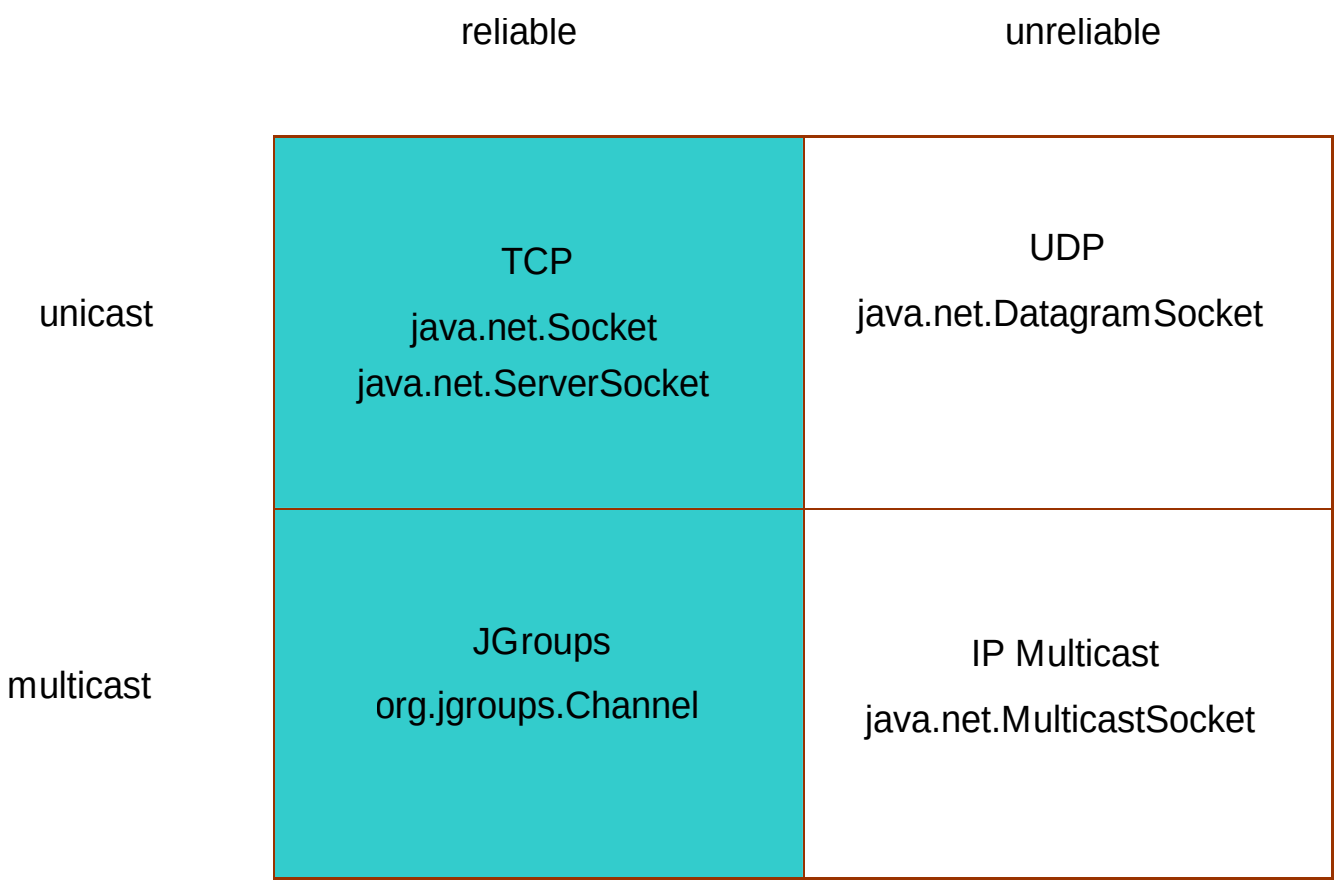
dimitris@redhat.com

Clustering infrastructure

JGroups – reliable multipoint communication

dimitris@redhat.com

What is missing?



dimitris@redhat.com

What is unreliable?

IP Multicasting offers no guarantees

- No notion of Group Membership
- Message size limitations
- Messages can get
  - ✓ Lost in transit
  - ✓ Dropped (buffer overflows at switch or receiver)
  - ✓ Reordered
- No Flow Control
  - ✓ Fast sender overwhelms slow receiver(s)

dimitris@redhat.com

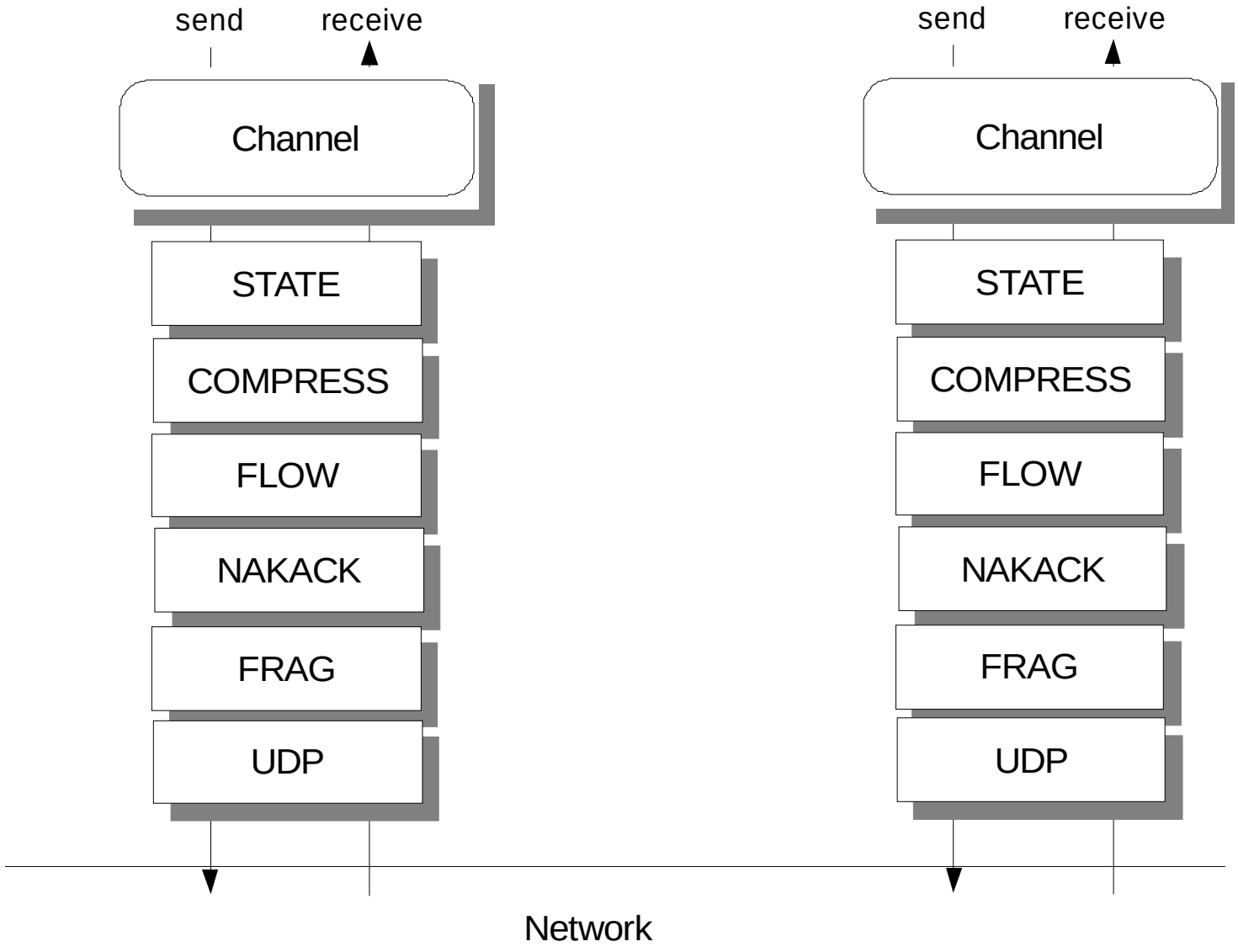
So what is JGroups?

A reliable group messaging library

- Manages group membership
  - ✓ Join, Leave, Shun
- Reliable messaging
  - ✓ Ordering
  - ✓ Flow control
  - ✓ Fragmentation
  - ✓ Acknowledgement/Re-transmission
- LAN or WAN based
  - ✓ IP multicasting default for LAN
  - ✓ TCP transport default for WAN

dimitris@redhat.com

JGroups Architecture



dimitris@redhat.com

Available Protocols (1)

- Transport
  - ✓ UDP (IP multicasting), TCP, TCP_NIO
  - ✓ Message batching
- Merging, failure detection (hangs, crashes)
- Reliable transmission and ordering
  - ✓ Using sequence numbers, dropped messages are retransmitted
- Distributed garbage collection
  - ✓ Consensus on received messages, older ones are purged

dimitris@redhat.com

Available Protocols (2)

- Group membership
  - ✓ Installs new views across a cluster when members join, leave or crash
- Flow control
  - ✓ Fast sender is throttled down to the pace of the slowest receiver
- Fragmentation
  - ✓ Large packets are fragmented into smaller ones and unfragmented at the receiver side
- Compression, encryption, authentication

dimitris@redhat.com

Available Protocols (3)

- State transfer
  - ✓ State transferred to a joining member without stopping the cluster
- Virtual Synchrony
  - ✓ All messages sent in view V1 are delivered in V1
  - ✓ Flushes unstable messages before a new view is installed
    - Makes sure all members have received all messages sent in V1 before installing V2
- Ordering
  - ✓ Total, causal, FIFO
- …70+ protocols

dimitris@redhat.com

JGroups Abstractions - Message

- src, dest: Address
  - ✓ Address: identity of a member (of the cluster)
  - ✓ src: filled in when sending (by JGroups)
  - ✓ dest: null == send to all members of the group
- buffer: byte[]
- headers: hashmap of headers
  - ✓ each protocol can add/remove its own headers
  - ✓ example: sequence number for reliable retransmission
- Message travels across the network

dimitris@redhat.com

JGroups Abstractions - Address

- A cluster consists of a number of members
- Each member has an address
- The address uniquely identifies the member
- Address is an abstract class
  - ✓ Implemented as a UUID
  - ✓ A UUID maps to a physical address
- An address can have a logical name
  - ✓ E.g. “A”
  - ✓ If not set, JGroups picks the name, e.g. „myhost-16524”

dimitris@redhat.com

JGroups Abstractions - View

- List of members (Addresses)
- Is the **same** in all members:
  - ✓ A: {A,B,C}
  - ✓ B: {A,B,C}
  - ✓ C: {A,B,C}
- Updated when members join or leave
- All members receive all views in the same order
- Channel.getView() returns the current view

dimitris@redhat.com

JGroups API

- Channel: similar to java.net.MulticastSocket
  - ✓ plus group membership, reliability
- Operations:
  - ✓ Create a channel with a config (XML file)
  - ✓ Connect to a group named "X". Everyone that connects to "X" will see each other
  - ✓ Send a message to all members of X
  - ✓ Send a message to a single member
  - ✓ Receive a message
  - ✓ Retrieve membership
  - ✓ Be notified when members join, leave (including crashes)
  - ✓ Disconnect from the group
  - ✓ Close the channel

dimitris@redhat.com

JGroups API

```
JChannel ch=new JChannel("/home/bela/udp.xml");
ch.setReceiver(new ReceiverAdapter() {
    public void receive(Message msg) {
        System.out.println("msg from " + msg.getSrc() + ": " +
                           msg.getObject());
    }
    public void viewAccepted(View new_view) {
        System.out.println("new view: " + new_view);
    }
});
ch.connect("demo-group");
System.out.println("members are: " +
ch.getView().getMembers());
Message msg=new Message(null, null, "Hello world");
ch.send(msg);
ch.close();
```


dimitris@redhat.com

Demo

- java –cp jgroups-2.8.0.jar org.jgroups.demo.Draw

dimitris@redhat.com

Data Grids

Infinispan

dimitris@redhat.com

What is Infinispan?

- A hashmap (extends ConcurrentHashMap) build on top of JGroups
- Local, replicated or distributed
  - ✓ Keys and values can be stored redundantly in a cluster
  - ✓ Sync or async communication (JGroups)
- Transactional or non-transactional
  - ✓ Non-Transactional
    - Updates are shipped after each modification
  - ✓ Transactional
    - Updates are shipped at TX commit
    - Isolation levels supported: RR and RC
    - Support for pluggable TxManagers

dimitris@redhat.com

Replication

- Changes are replicated to all cluster nodes
- Every node has the same state

A

id	322649
name	Bela
key	value

B

id	322649
name	Bela
key	value

C

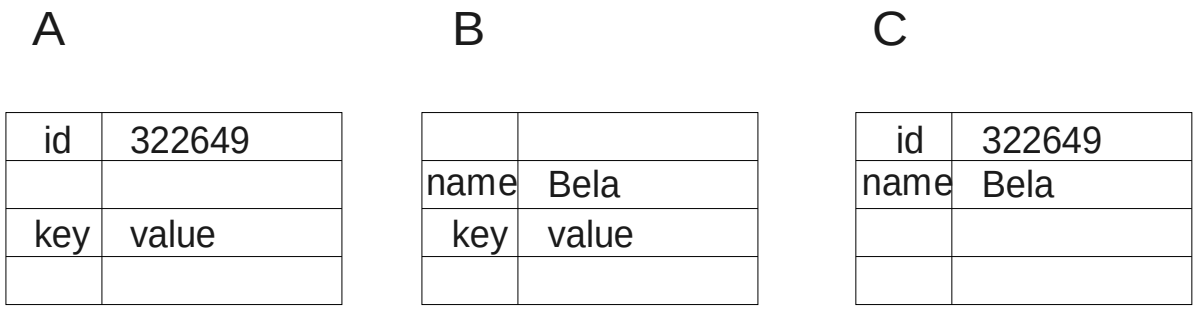
id	322649
name	Bela
key	value

dimitris@redhat.com

Distribution

- There are only N copies of each key/value pair
- The servers on which the data resides are determined via consistent hashing

E.g. N = 2, each key/value pair is stored on 2 servers



dimitris@redhat.com

Distribution

- N = 1
  - ✓ No redundancy
  - ✓ Data is spread across the cluster
  - ✓ RAID 0
- N > 1
  - ✓ Variable redundancy
  - ✓ ~ RAID 5
- N == -1
  - ✓ Data is stored everywhere
  - ✓ Same as replication
  - ✓ ~ RAID 1

dimitris@redhat.com

Other Features

- Cache loaders, Evictions policies, Persistence
- Synchronous and Asynchronous Replication
- Concurrency and Isolation Levels
- Transactions and Locks
- Fine-grained replication
- JMX Metrics
- Query API
- …

dimitris@redhat.com

Data Grids and Clouds

- Databases on clouds don't make sense
  - Traditional modes of data storage won't work

- Scalability is crucial
  - Databases still are a bottleneck
  - … and single point of failure!

dimitris@redhat.com

Conclusions

dimitris@redhat.com

Recap

- JBoss brings state-of-the-art clustering facilities to the masses, with zero license costs
- Clustering facilities are useable at different levels (JGroups, Infinispan, JBossAS, etc.)
- Clustering always has a performance cost, state replication is expensive.
- Memory is the new disk, disk is the new tape.

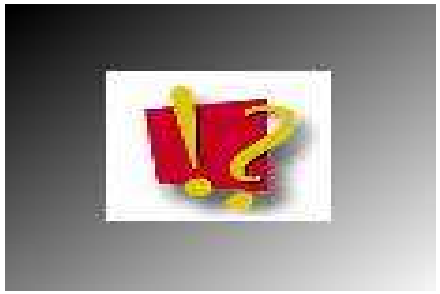
dimitris@redhat.com

More Info

- JGroups – www.jgroups.org
- Infinispan – www.jboss.org/infinispan
- JBoss AS – www.jboss.org/jbossas

Join the Discussion – Get the News!
GR-JBUG – Greek JBoss User Group

http://groups.google.com/group/gr-jbug



dimitris@redhat.com