



JBoss & Infinispan

open source data grids for the cloud era

Dimitris Andreadis

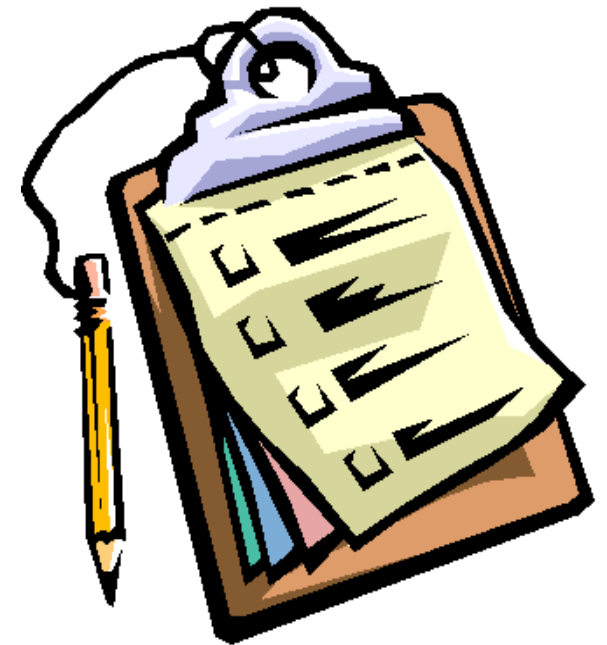
Manager of Software Engineering
JBoss Application Server
JBoss by Red Hat

5th Free and Open Source
Developer's Conference
May 2010, Athens/Heraclion

Adapted from an original presentation by
Manik Surtani - Infinispan Project Lead

Agenda

- Clouds are Today
- JBoss Clustering Architecture
- Cloud Challenges and Solutions
- Introducing Infinispan
- Conclusion





Clouds Are Today

Why now?

We're in a perfect storm!

- OS virtualization is mature
- Bandwidth is cheap and plentiful
- ... and we're in a financial crisis!
 - Everyone wants to cut costs, be more efficient!
 - Making changes is easier now



Clouds are today!



- Clouds are happening
 - *aaS: SaaS, PaaS, IaaS
- You cannot escape them!
 - Public: Amazon, Google, GoGrid, Rackspace
 - Private: Eucalyptus, VMWare, IBM, RedHat
- Clouds become mainstream
 - Traditional datacenters marginalized to niche deployments



Why are clouds popular?

- Piecemeal cost
- Pay for what you use
- Massive, global data centers means high availability
- Everyone benefits from economies of scale
- Ability to scale on demand
- Very fast provisioning
- Proven charging model
- Remember timesharing on mainframes?



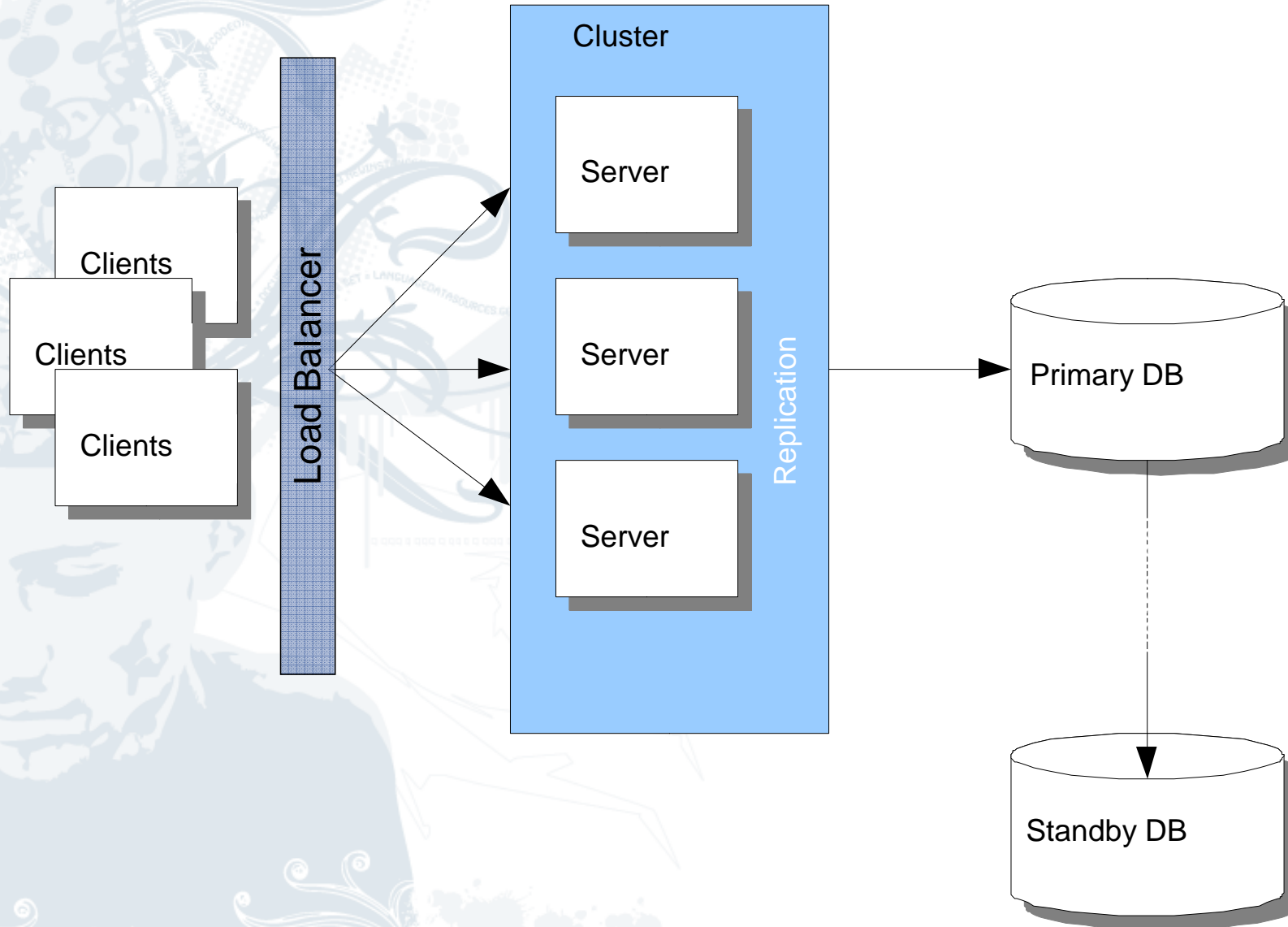
JBoss Clustering Architecture

JBoss AS Clustering

- JBoss brought mission-critical application server clustering services to the masses
- Just start JBoss in the “all” configuration
 - `./run.sh -c all`
- JBoss clusters are dynamic
 - Cluster nodes detect each other automatically
 - No configuration of topology necessary



Typical Clustering Setup



What does it mean for your applications?

- Load balancer settings (e.g. `httpd+mod_jk/mod_cluster`)

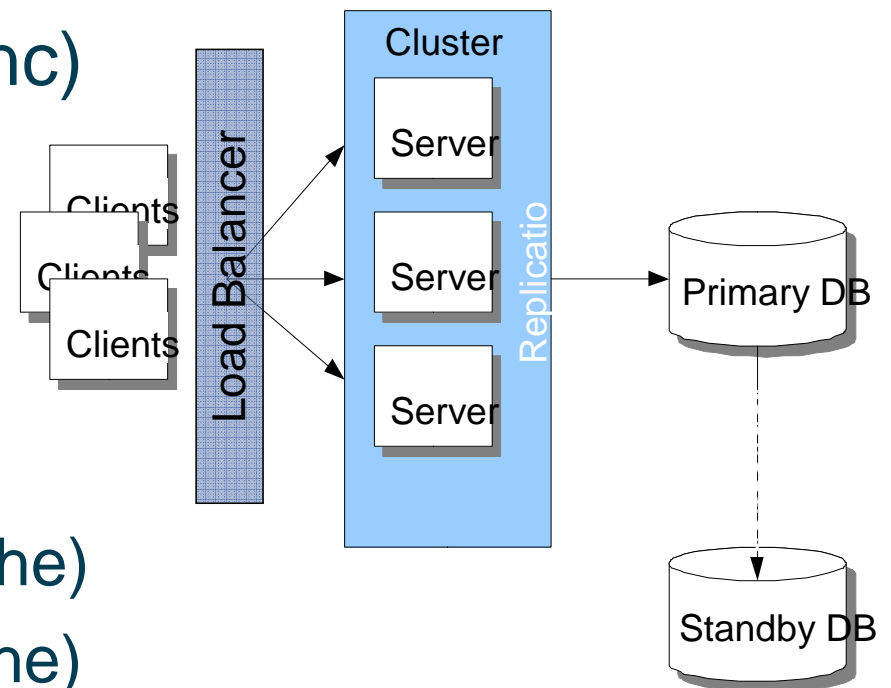
- Session Replication (sync/async)

- Http Sessions
- Statefull Session Beans

- Data Access

- Query caching (Replication Cache)
- Entity caching (Invalidation Cache)

- Highly Available (HA) Singleton services





Cloud Challenges and Solutions

Cloud Challenges

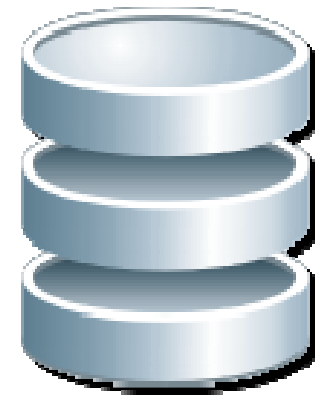
- Data Governance
- Manageability
- Monitoring
- Availability
- Security
- ...

• Scalability

- Going from dozens to hundreds of nodes
- Memory footprint & communications overhead
- Dynamicity, nodes that join and leave anytime

Major Challenge: the Data Storage

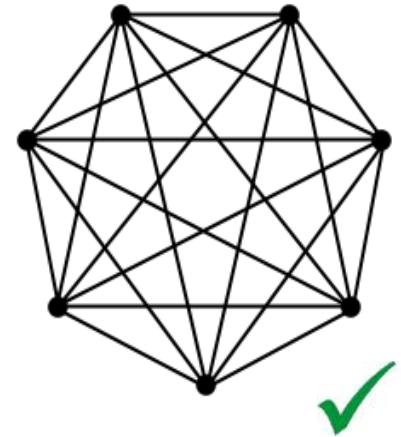
- Databases on clouds don't scale
 - DBs become the bottleneck
 - and a single point of failure!
- Various attempt to solve it
 - E.g. Amazon Web Services & Elastic Block Store (EBS) or S3
 - Native DB clustering (e.g. Oracle RAC)
 - ...



The solution: Data Grids!

or Elastic Caching Platforms

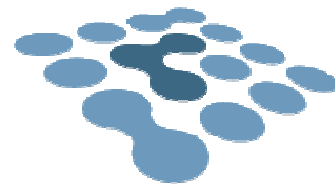
- Data grids are perfect for clouds
- Highly scalable
- Highly concurrent
- Very low access latency
- No single point of failure
- Memory 2 orders of magnitude faster than disk



- Data grids
- Amazon SimpleDB uses Dynamo
- Many other commercial and OSS offerings



Introducing Infinispan



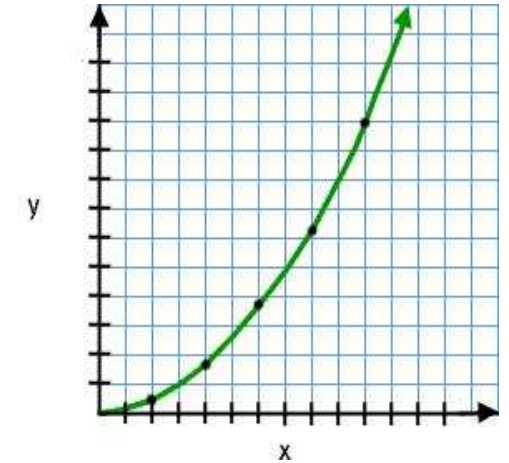
Infinispan

What is Infinispan?

- A highly scalable data grid platform
 - a Map (extends ConcurrentHashMap), like JSR-107's JCache
 - Java-based, 100% open source licensed (LGPL)
 - Uses JGroups for group communication
- Standalone or Clustered
 - Clustered modes includes, invalidation, replication and distribution
 - Sync or Async communication

More scalable than JBoss Cache

- . Data organised in Map-like structures
 - . As opposed to a tree, more memory efficient
- . Designed for concurrency
 - . minimising synchronized blocks, mutexes
- . Containers are naturally ordered
 - . makes eviction much more efficient
- . Uses JBoss Marshalling
 - . smaller payloads + poolable streams = much faster remote calls



Local / Standalone Operation

- Simple standalone cache
 - Highly concurrent with Tx Isolation
 - Write-through CacheStore
 - Eviction
 - Etc.
-
- Front DBs or other expensive non-scalable data stores

A

| | |
|------|--------|
| id | 322649 |
| name | Bela |
| key | value |
| | |

Invalidated Data Grid

- A set of local standalone caches
 - that are aware of each other!
- Each node fill in its own cache
- When an entry is changed in any cache, the others nodes must flush it
- E.g. Hibernate 2nd level cache

A

| | |
|------|--------|
| id | 322649 |
| name | Bela |
| key | value |
| | |

B

| | |
|------|-------|
| | |
| name | Bela |
| key | value |
| | |

C

| | |
|------|--------|
| id | 322649 |
| name | Bela |
| | |
| | |

Replicated Data Grid

- Changes are replicated to all cluster nodes
- Every node has the same state
- Useful for smaller data sets and/or clusters

A

| | |
|------|--------|
| id | 322649 |
| name | Bela |
| key | value |
| | |

B

| | |
|------|--------|
| id | 322649 |
| name | Bela |
| key | value |
| | |

C

| | |
|------|--------|
| id | 322649 |
| name | Bela |
| key | value |
| | |

Distributed Data Grid

with optional L1 caching

- 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

$N = 1$

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)

A

| | |
|-----|--------|
| id | 322649 |
| | |
| key | value |
| | |

B

| | |
|------|-------|
| | |
| name | Bela |
| key | value |
| | |

C

| | |
|------|--------|
| id | 322649 |
| name | Bela |
| | |
| | |

Data distribution

- Consistent hash based data distribution
 - Goal of efficient scaling to 1000's of nodes
 - Peer to peer communication
- Lightweight, “L1” cache for efficient reads
 - On writes, “L1” gets invalidated
 - Or entries simply expire
- Dynamic rebalancing



“Borrowed” from JBoss Cache

- JTA transactions
- Replicated data structure
- Eviction, cache persistence
- Notifications and eventing API
- JMX reporting
- Fine-grained replication
- MVCC locking
- Non-blocking state transfer techniques
- Query API
- Custom (non-JDK) marshalling





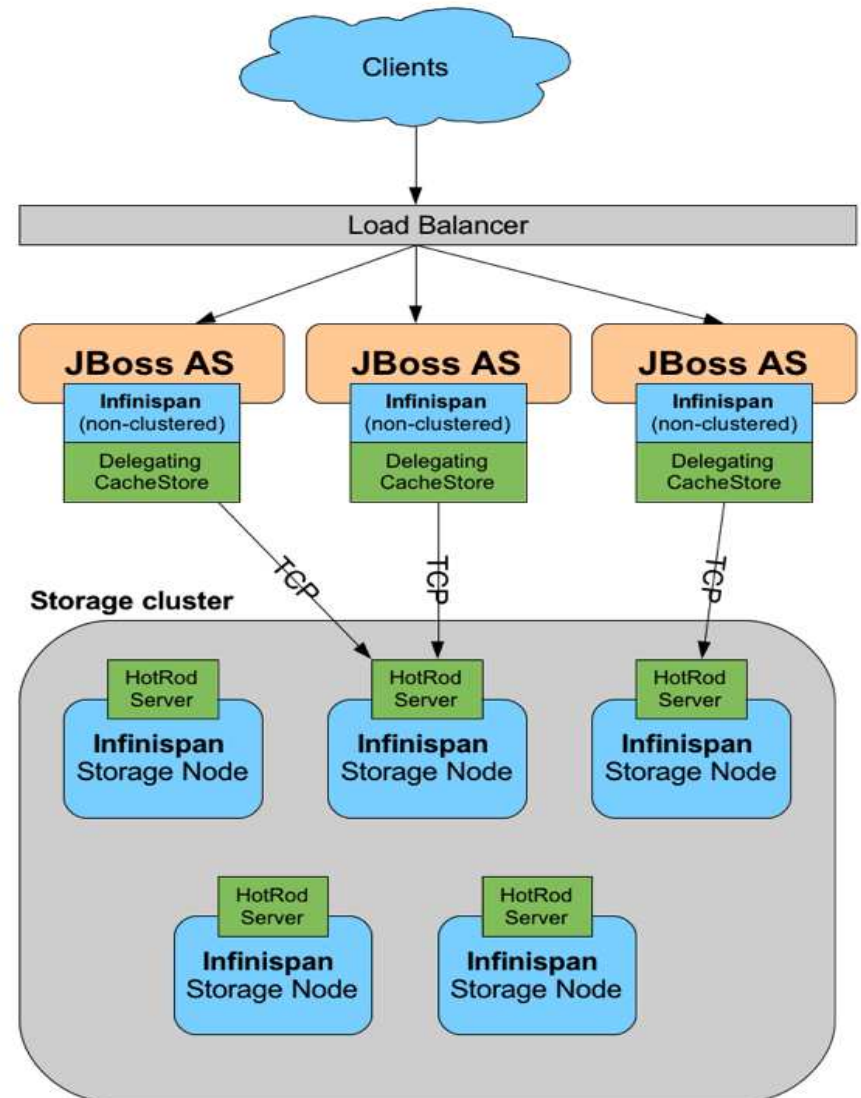
... and new features!

- REST API - REST-* caching spec effort
- Client/server module with memcached compatibility
- JPA API
- Ability to be consumed by non-JVM platforms
- JOPR based GUI management console
- Distributed executors
- Map/reduce programming model made easy!

Infinispan in JBoss AS 6

- Http session & SFSB replication
 - Infinispan Distributed Grid
- Entity Caching
 - Infinispan Invalidation Grid

Could also use Infinispan as a Storage Cluster!



Recap - Why is Infinispan sexy?

- Transparent horizontal scalability
- Fast, low latency data access
- Ability to address a very large heap
- Cloud-ready datastore
- Not just for Java
- Free and doesn't suck!





Conclusion

To sum it up

- Clouds are becoming mainstream
 - But create a whole new set of problems
- DBs and clouds pose many challenges
 - Data grids offer a viable alternative
- Infinispan, a new open source data grid
 - Used inside or outside JBoss AS
- Memory is the new disk, disk is the new tape!

How can YOU participate?

- Download and try it out!
 - Report bugs. Not just in code, even docs, wikis, etc.
 - Suggest new features!
- Test with your own use cases
 - We love to hear how people use our stuff!!
- Lend a hand with development
 - Open and democratic dev process
 - Helps prioritise features you want!
 - Several non-Red Hat core committers already!



More Info

Infinispan – www.jboss.org/infinispan

JBoss AS – www.jboss.org/jbossas

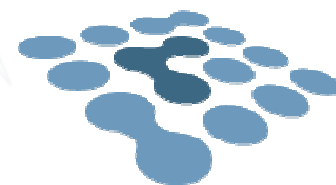
Join the Discussion – Get the News!
GR-JBUG – Greek JBoss User Group

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

Dimitris Andreadis
dimitris@redhat.com



Extra Slides - Roadmap



Infinispan

Roadmap

Infinispan 4.0.0

New Map API, Async API

Distributed cache

Management tooling

REST API

Infinispan 4.1.0

Client/server API, memcached module, language bindings

Query API



Roadmap

Infinispan 5.0.0

JPA API

Fine-grained replication

Infinispan 5.1.0

Distributed executors, map/reduce model

Dynamic provisioning

Infinispan 5.2.0

Distributed querying based on map/reduce

