JBOSS®
DATA GRID

Ease into noSQL,
making big data work.

JBoss®
by Red Hat

The Challenge of scale

1) Throw "more" at the problem

2) Start Over

3) Deploy a smarter caching strategy

The Challenge of scale

1) Throw “more” at the problem

More servers, more databases, etc

Expensive, may only work once

May not ever work

The Challenge of scale

1) Throw "more" at the problem

2) Start Over

Sharding, break up app, go stateless

May take more time than you have

Costly to rearchitect / redevelop

The Challenge of scale

1) Throw "more" at the problem

2) Start Over

3) Deploy a smarter caching strategy

NoSQL, Distributed Cache, Data Grids

Elastic demand based scaling

Highly distributed and fault tolerant

Smarter Caching Options

Local Caching

How:
Part of application

Benefits:
Familiar to devs
Used all the time

However,
Single instance
One of many other server tasks

Distributed Caching

How:
Open Source memcached

Benefits:
Low / No Cost
Massive Scale

However,
Requires additional infrastructure
May lack elasticity

Data Grids

How:
Standalone Products

Benefits:
Fault tolerant
Elastic scaling
Managable

By 2014, at least 40% of large organizations will have deployed one or more in-memory data grids.

SOURCE – Predicts 2012 – Cloud and In Memory Drive Innovations in Application Platforms
Gartner 2012

JBoss Data Grid (JDG)

transactional data access and caching services for high-performa

on demand, automatic failover, optionally store to and load from d

nchronous or asynchronous operation. JBoss ON support for pe

Why use JDG

Improve scale and performance – Move data and processing closer, partition workload across machines. JDG manages the complexity of locality, access patterns and persistence models.

Reduce DBMS costs – RDBMS development time, licensing, support teams don’t need to grow with the application

Simplify Development – Application teams focus on application, JDG manages persistence in an appropriate fashion.

Existing Uses

Delivery of **real time** routing, tracking and logistics **information**

Provide rapid access to content for **thousands of users**

Reduce RDBMS workload with **intelligent caching**

Enable i**n-memory speed** with **transactional support** for compute intensive applications

TBD – Add CBOE use-case

Okay, I need that in captain dummy talk, Kaylee

In-JVM, self-managed cache to scale applications otherwise bound by a relational database

Pacemaker for a database

Distributed Key/Value store to support operational workloads using a standard, language-agnostic API

a NoSQL data store

JDG Features

Configurable data distribution

Overflow memory to disk or other 'cachestore'

Integrates with existing data stores

Multiple locking strategies

Scale linearly, without complexity

Low latency - Fast and Deterministic access

Highly Available and resilient

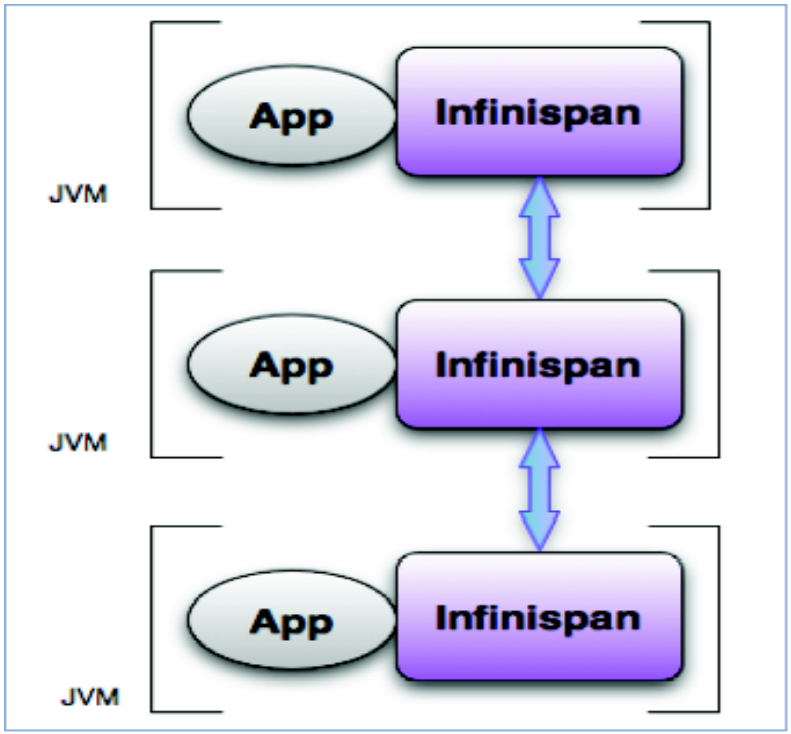
Elastic – scale out and back again

Manageable – JBoss ON is JDG aware

Using JBoss Data Grid

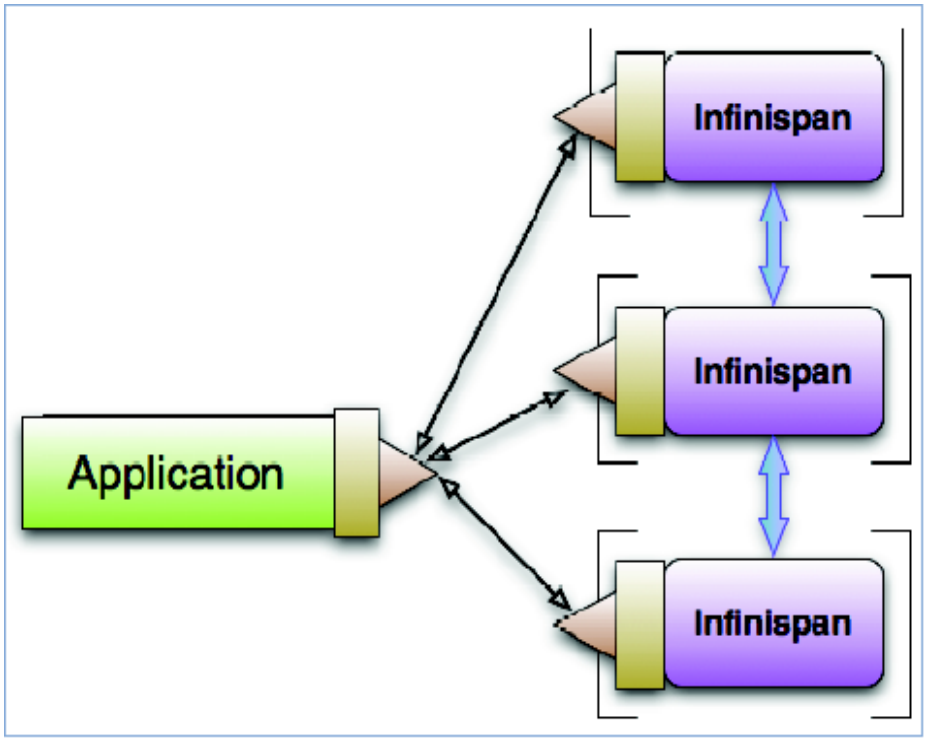
Option 1 – As a library

Application co-located with cache node



Option 2 – Remote Client/Server

Remote Client Access, multiple protocols



Library Features - In-memory data grid

Distributed, Off-Heap HashMap

Application container agnostic

InVM - Java Only

Synchronous/asynchronous API

Event aware

Data can be grouped

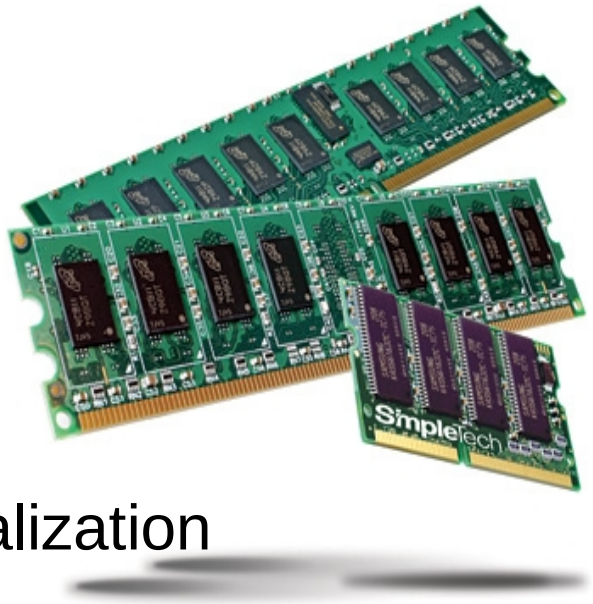
JTA compatible

JMX monitoring

Library Benefits

Performance booster

In-memory access for data that is expensive to retrieve and frequently accessed



Simplify Large Scale Applications

Built-in eviction to manage JVM heap

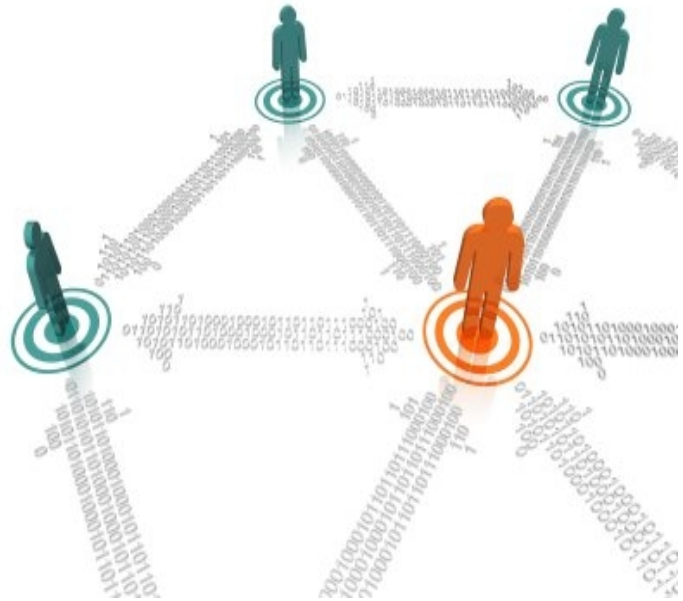
Customizable

Very rich API - data grouping, notifications, serialization

Supporting Scaffolding for existing frameworks

CDI, SEAM,Apache Camel, ESBs

Remote Client / Standalone Grid Features



- Client/Server topology
- REST/Memcache/Hot Rod client access
- Manageable via JON

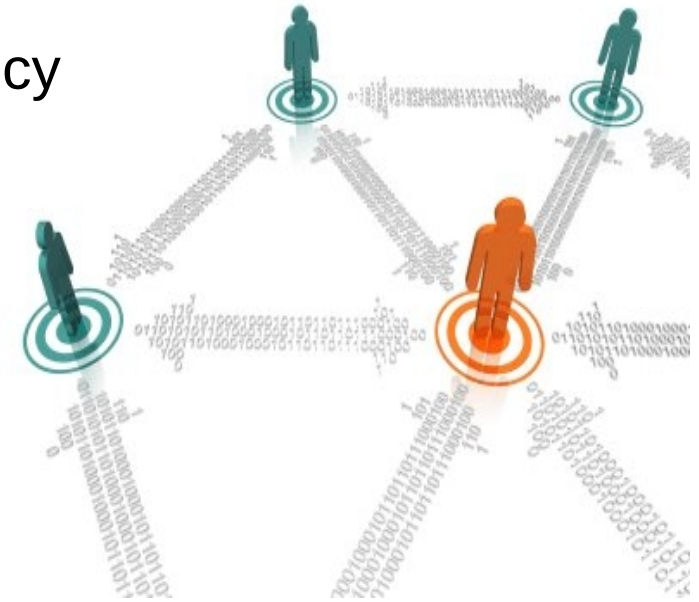
Standalone Grid Benefits

Fast - In-memory, low latency, high concurrency

Available - Inherent redundancy

Distributed - Data locality

Data managed independently of application



Available Grid APIs

	Protocol	Client Libraries	Clustered ?	Smart Routing	Load Balancing/ Failover
REST	**Text**	N/A	**Yes**	**No**	Any HTTP load balancer
Memcached	**Text**	Plenty	**Yes**	**No**	Only with predefined server list
Hot Rod	**Binary**	Currently only Java	**Yes**	**Yes**	Dynamic

Supported Beta – out **now**

GA – June (before Summit)

6.x – Summery

Library mode out of Tech Preview

6.x + 1 – Q4 2012

JDG Beta

OUT NOW