Oracle Coherence

By Mustafa Ahmed

Agenda

- The Problem
- Solution
- Data Grids
- Replicated Topology
- Partitioned Topology
- Near Topology
- Events
- Query
- Read Through Caching
- Write Through Caching
- Write Behind Caching
- Coherence Code Examples
- Conclusion
  - Performance
  - Availability
  - Scalability

The Problem

- Data
  - Extreme increase in Access Volume & Complexity of Data
- Driving Data Demand
  - Virtualization
    - Ability to move applications around several machines
  - Service Oriented Architecture (SOA)
    - Integrated services that can be used in Multiple business domains
    - Relying on other services

Solution – Oracle Coherence

- Provide Reliable, Scalable, Universal Data Access and Management.
  - Performance
    - Solves Latency and Bandwidth Problems
  - Availability
    - Having the data available at all times
  - Scalability
    - Handle growing demand of Data Efficiently

Data Grids

- Manages Information in a grid environment
  - Lots of servers working together
  - Servers do not run independently
    - Server manages state
    - Even server failure occurs.
  - Adding more servers
    - Concept of scale out
    - It will manage more data and can handle more transactions per second.
- Data as a Service
  - Middle Tier
  - In App Server
- Data Integration is in Data Service
  - Integration can occur in Domain Model

Data Grids

- Combines Data Management with Data Processing
  - Push processing where data is being managed
  - Read or Write data across any number of servers
- Single System Image
  - No need to show server infrastructure
  - Pretend all the information is Local
  - Logical view of all data in all the servers

Data Grids

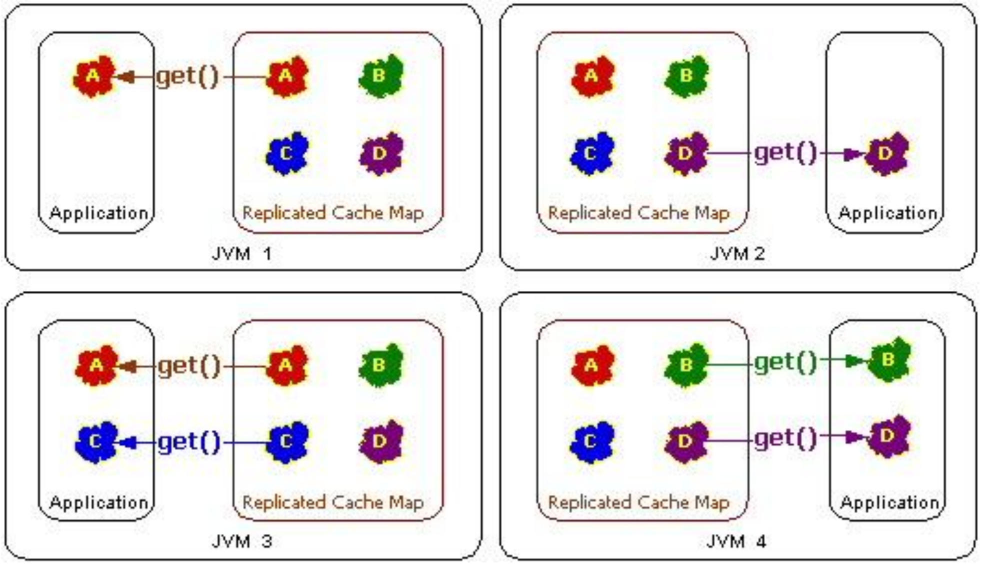
- There are two things you can move in a Distributed Environment
  - State
    - Distribution of a state is referred to as replication
  - Behavior
    - Moving messages
- Data Grids combine these two concepts
  - You can either move data or the processing where data is sitting
    - Push all the processes to the Information

Data Grids

- Locality of Data
  - Most applications spend most of the time waiting for data
  - If the data is partitioned with non overlapping regions the behavior can be moved to the server that owns the data to process
  - Results In lower latency

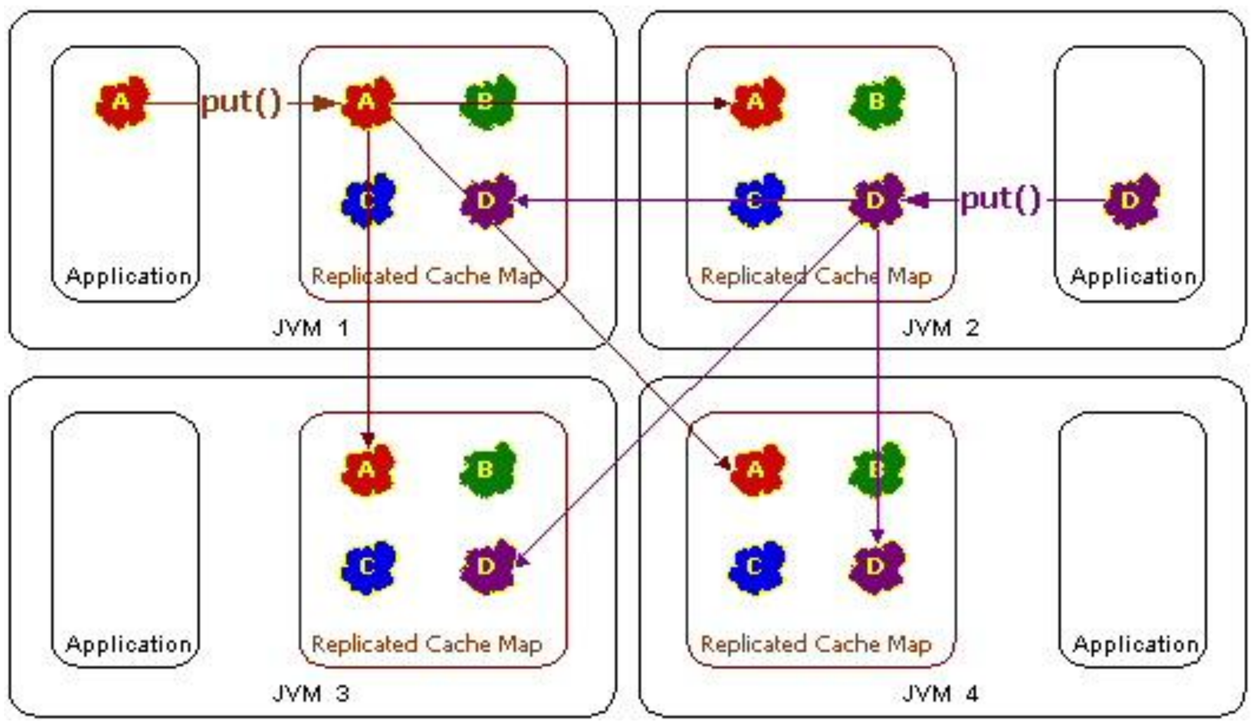
Replicated Topology

- Technology introduced In 2001
- Replicate information among all servers
  - Data is replicated to all members in Data Grid
- Problems
  - Scalability Problem
    - Capacity of Information Stays the same



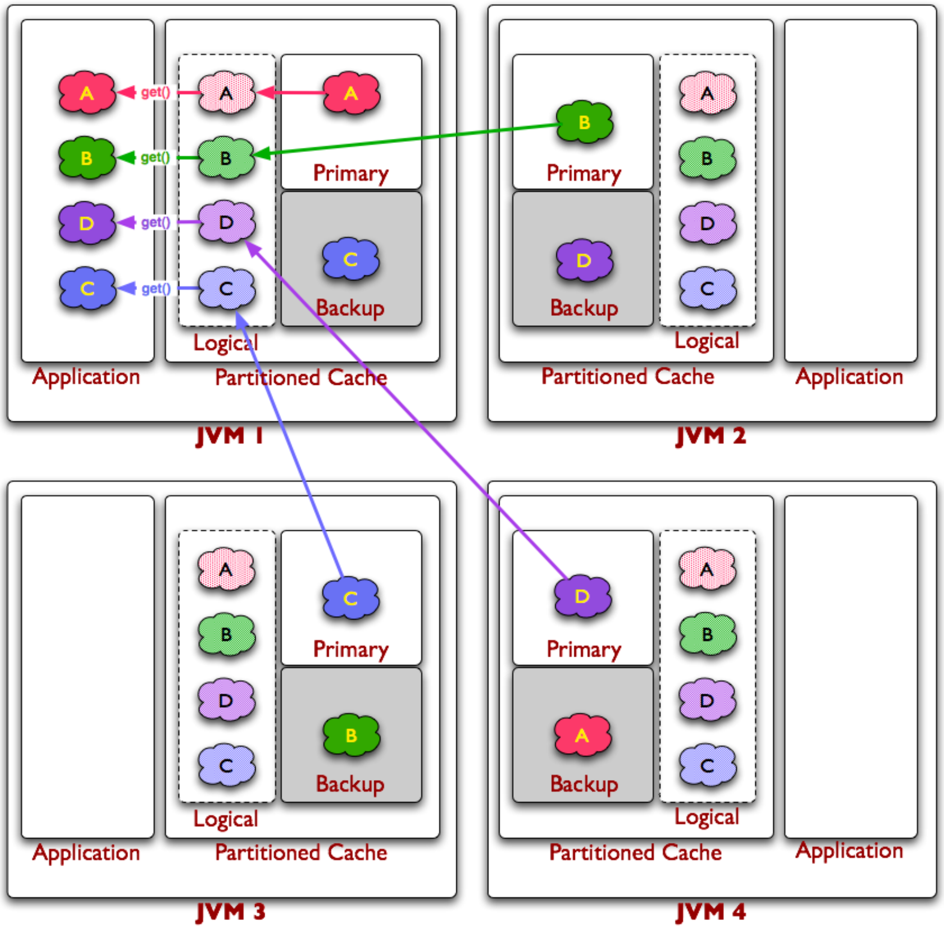
Replicated Topology

- Expensive
  - Update Each Server every time
  - Conceptually its expensive



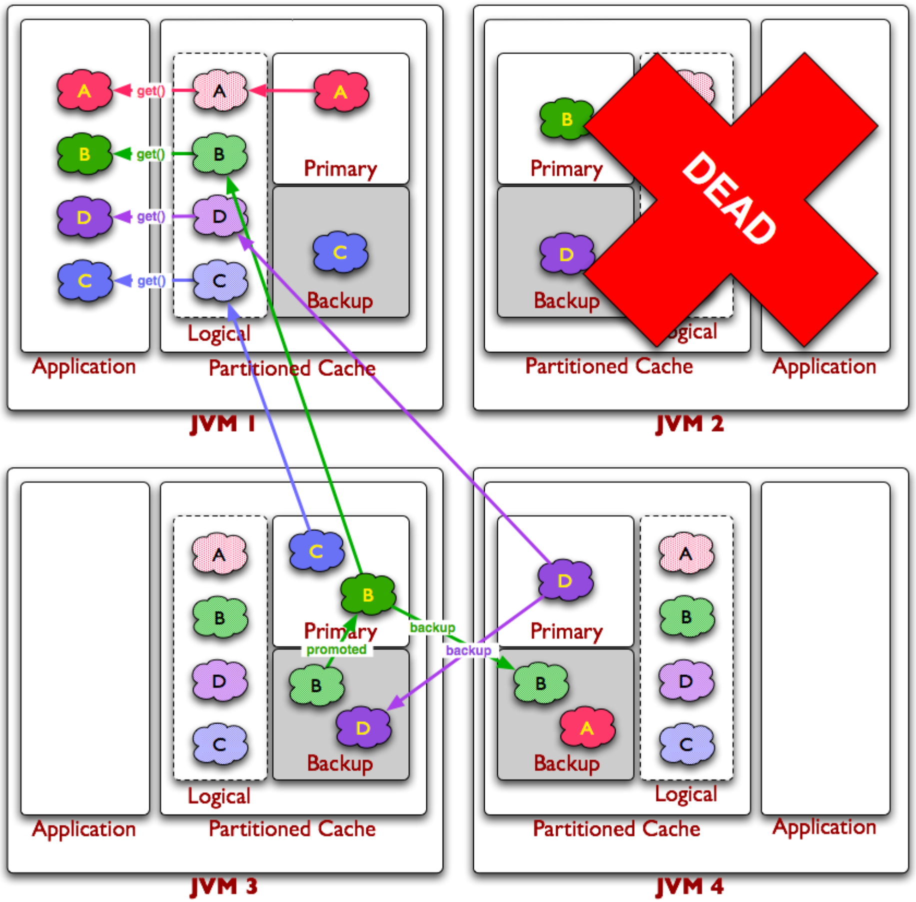
Partitioned Topology

- Each Information is spread out across the servers (Peer to Peer)
- Load Balancer
  - Keeps track of the load
  - Move from one server to another
  - Sends to the server which owns the data
- Exactly one server owns the information
  - Has a sync back up for it



Partitioned Topology

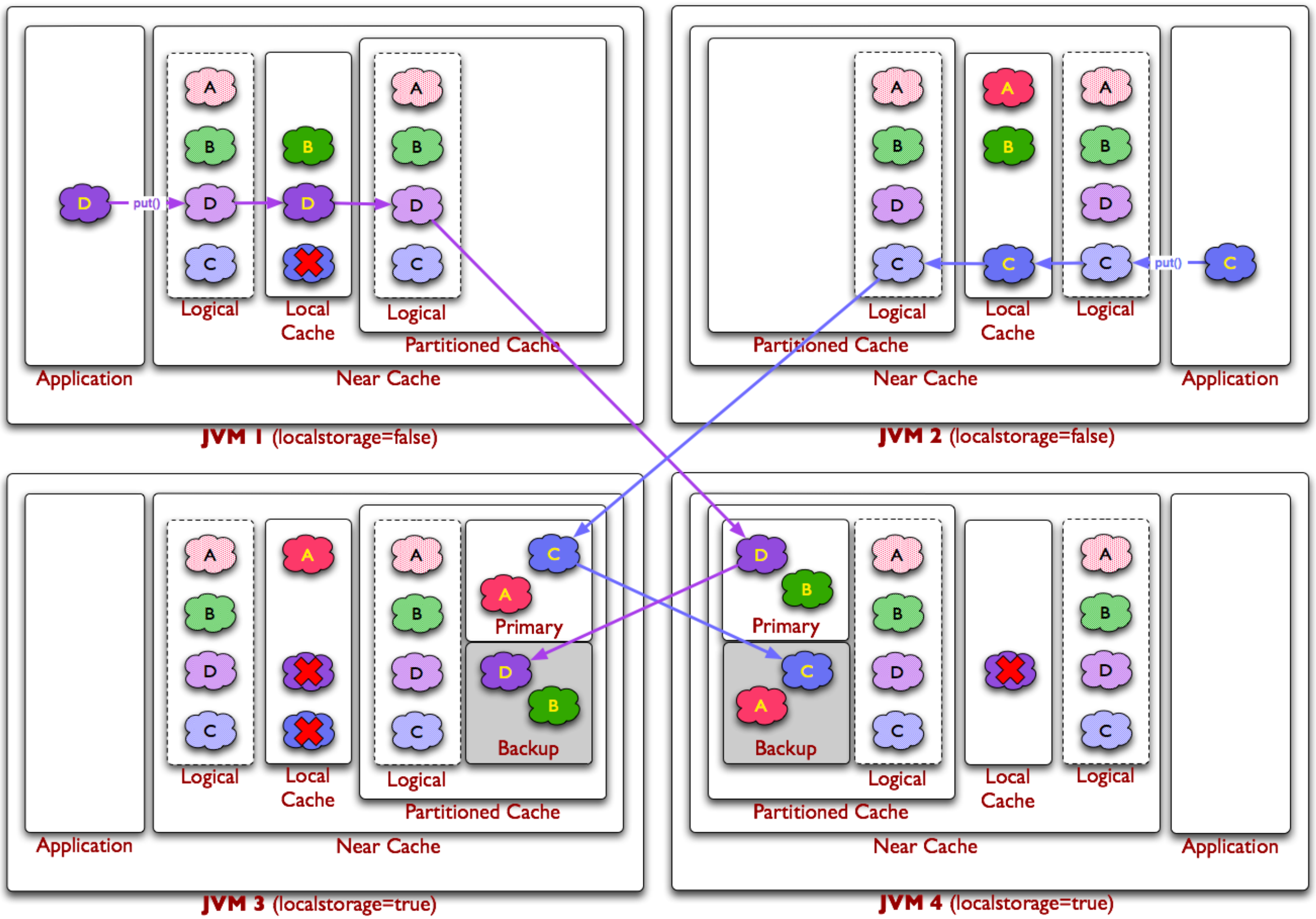
- Failure Occurs
  - The operation still finishes correctly
  - Increase servers from 2 to 2000 servers it increases scalability
  - All servers are disposable at any period of time



Near Topology

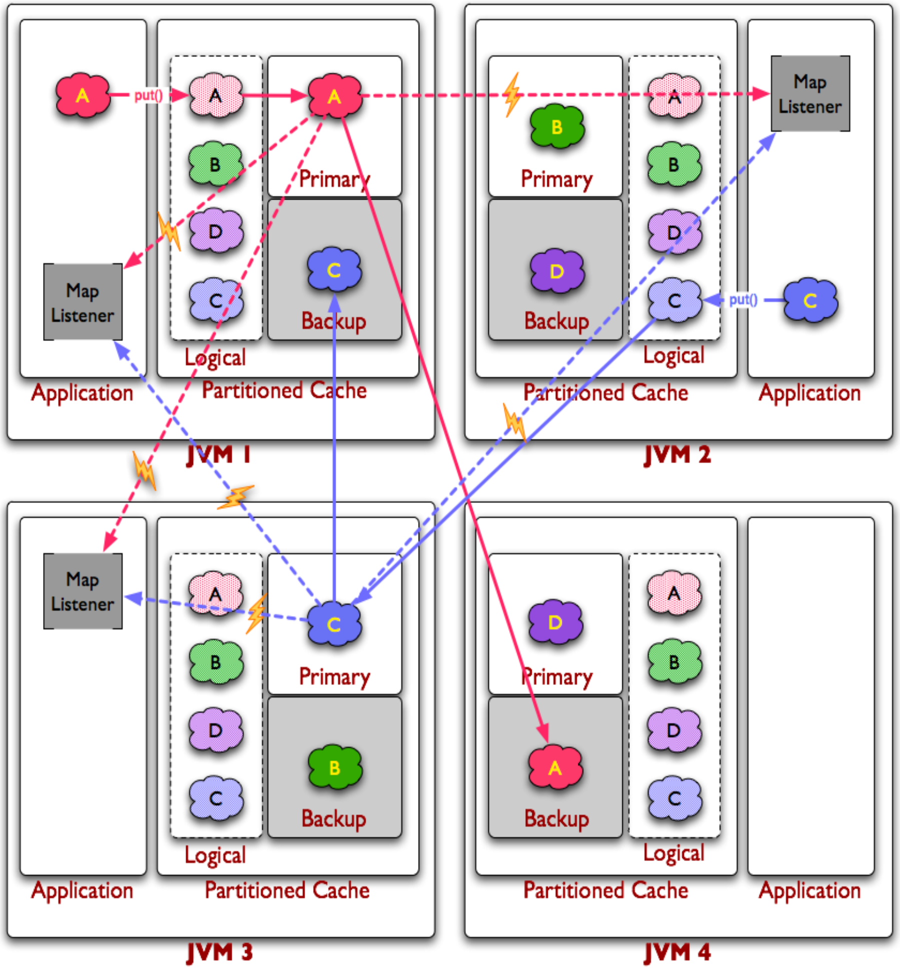
- L2 Cache vs. L1 Cache
  - Partitioned Topology as L2 Cache
  - Near Topology as L1 Cache
- Stores it Locally
  - If asks again then gets it locally
- Demand base replicated caching
- Zero Latency access to recently used data

Near Topology



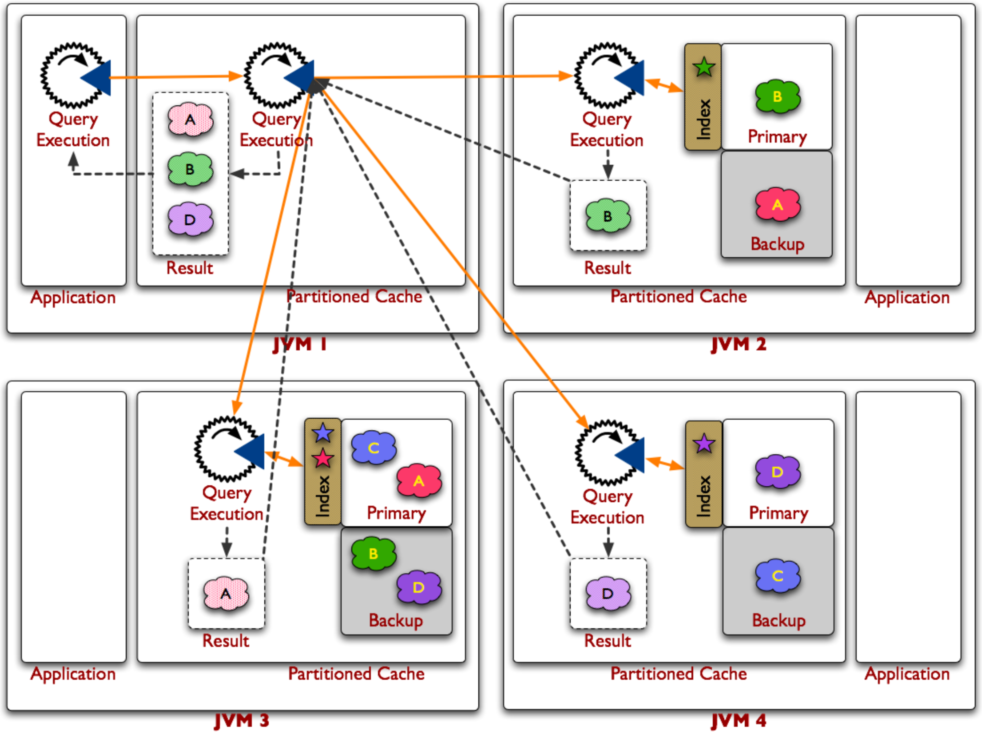
Events

- All the dataset provide events regardless of Topology
- Events are distributed efficiently to the interested listeners



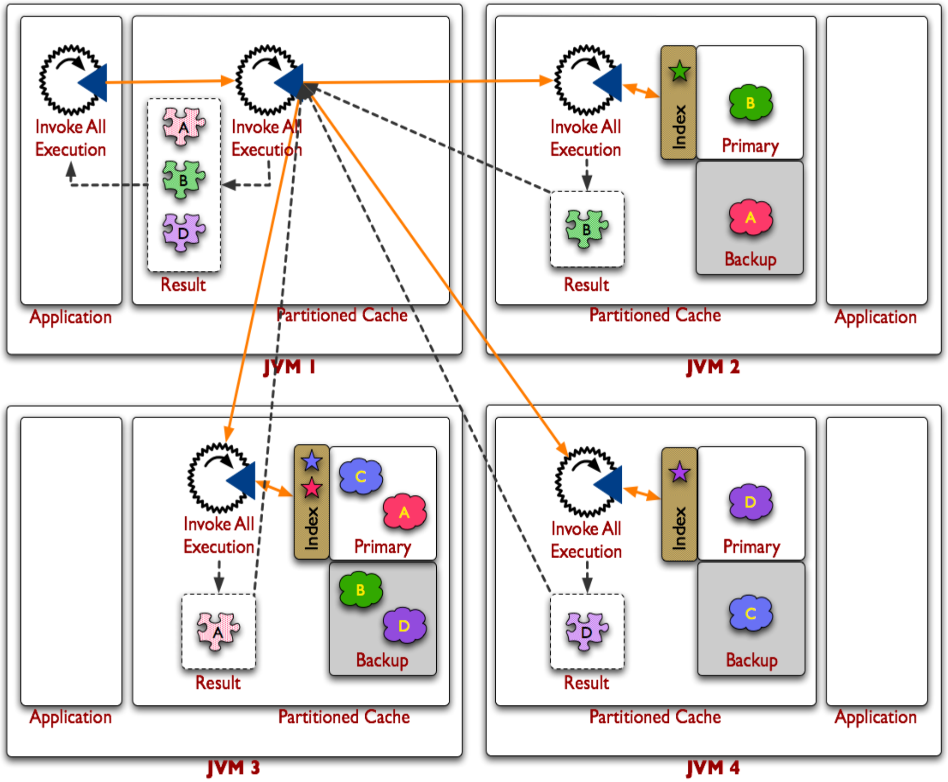
Query

- Parallel Query
  - Query performed parallel across the data grid using indexing
  - All doing the local portion of the Query



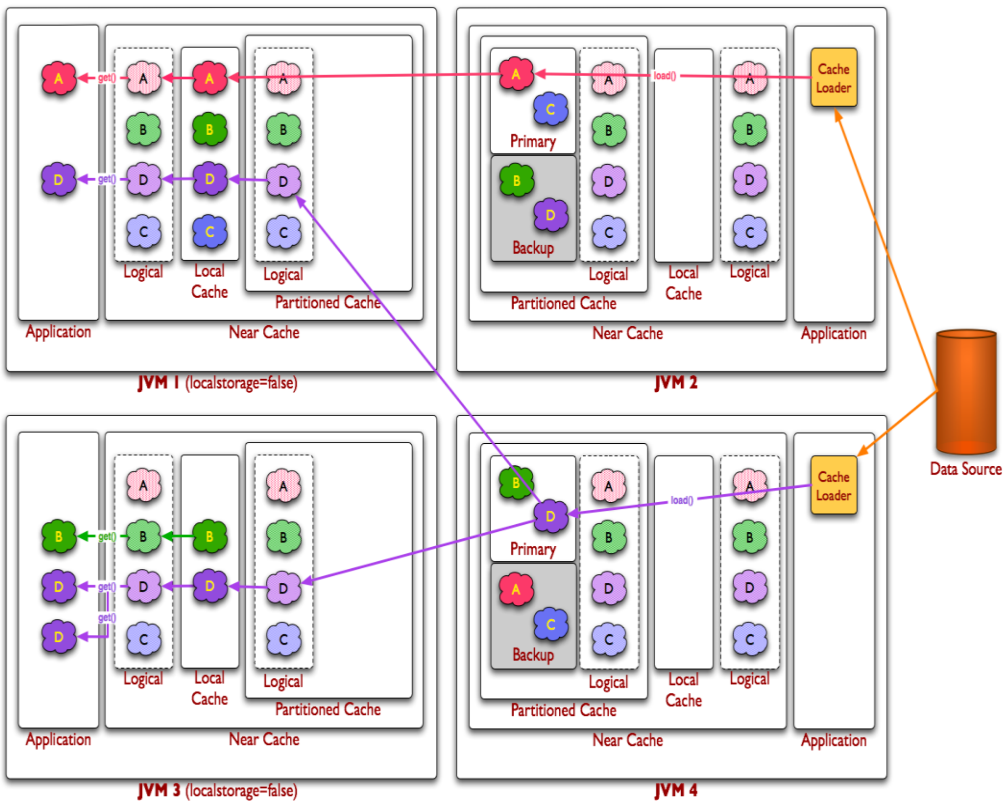
Query

- Continuous Query
  - Combines a Query with Events to provide a local materialized view
  - Result is up to date in real time
  - Like in near topology but always contains the desired data



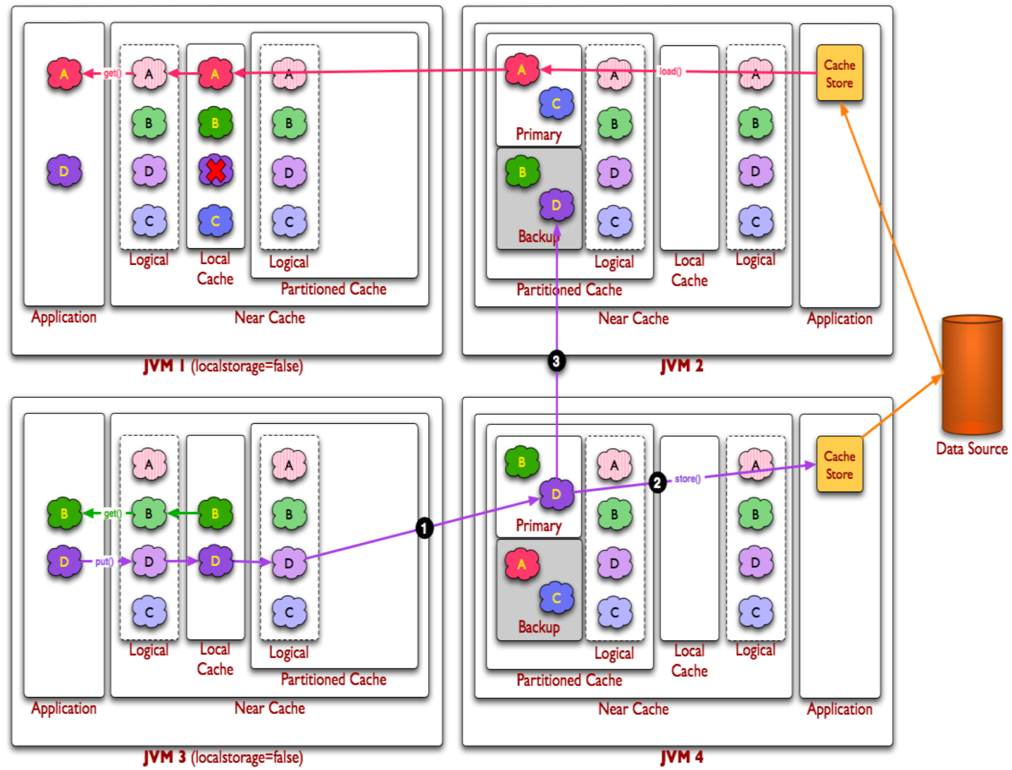
Read Through Caching

- Finds it in L1 or L2 Cache
  - Otherwise sends a request to the database
- Only sends one requests
- Coalesces multiple reads to reduce the database load



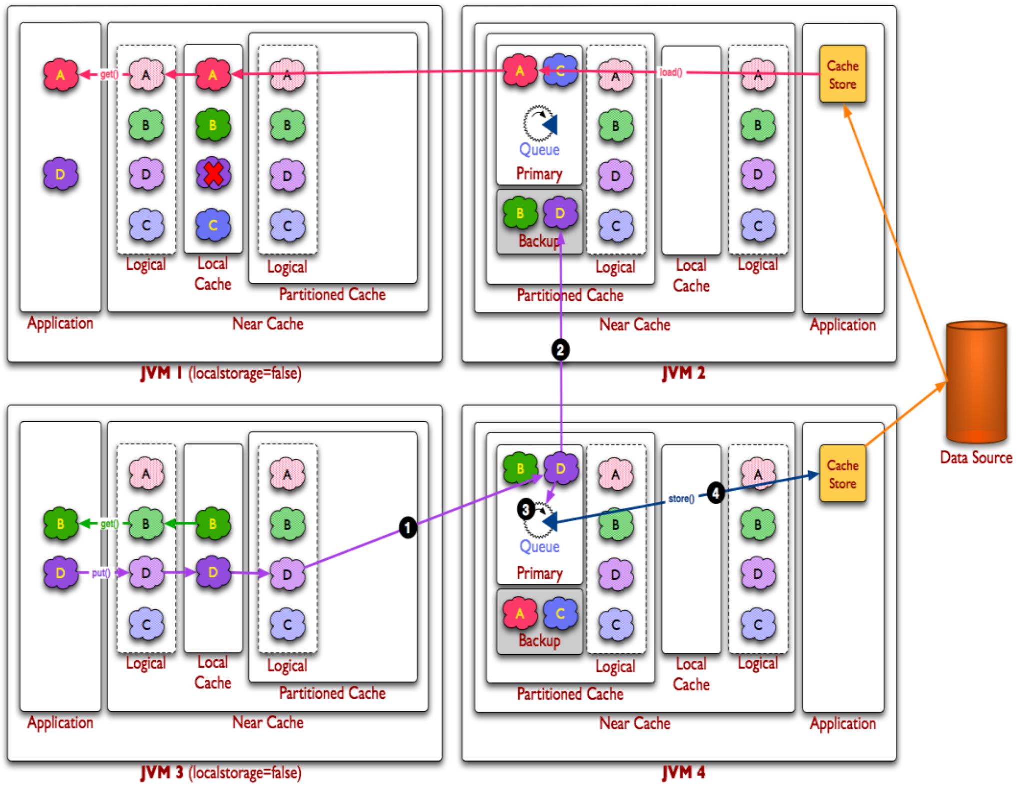
Write Through Caching

- Writes first to the database and then commits to the cache
- Not a Two-Phase Commit
- Keeps the in-memory data and the database in sync.



Write Behind Caching

- First writes it to the cache
  - Later commits it to the database
  - This assures the latest version of the cache
- Batches all the writes into one object
- Geico uses it
  - Improved performance
  - 90% reduction in database usage



Coherence Code Examples

- Joins an existing cluster or forms a new one

```
Cluster cluster = CacheFactory.ensureCluster();
```

- Leaves the current cluster

```
CacheFactory.shutdown();
```

Coherence Code Examples

```
NamedCache nc = CacheFactory.getCache("mine");

Object previous = nc.put("key", "hello world");

Object current = nc.get("key");

int size = nc.size();

boolean exists = nc.containsKey("key");
```

Coherence Code Examples

- Observe changes in real time as the occur

```
NamedCache nc = CacheFactory.getCache("stocks");

nc.addMapListener(new MapListener() {
    public void onInsert(MapEvent mapEvent) {
    }

    public void onUpdate(MapEvent mapEvent) {
    }

    public void onDelete(MapEvent mapEvent) {
    }
 });
```

Conclusion – Performance

- Performance
  - Solves Latency Problems And Preserve network bandwidth
    - Cache recently used data
    - Ability to execute tasks parallel across the data grid
    - Moving the process where the data is

Conclusion – Availability

- Availability
  - Remove all single point of failure
  - Added redundancy to improve availability
  - Able to Queue updates if database is not available
  - Increase availability from 11 days to 2.5 hours per year

Conclusion – Scalability

- Scalability
  - Scale Out functionality
    - Database Sharding
  - Coherence eliminates Database Sharding
  - Distributed cache
  - Updates performed against the cache data
  - Scaling both capacity and throughput
    - Adding more nodes to the Coherence Cluster

Any Questions