Building Scalable Applications with Hazelcast

FUAD MALIKOV
CO-FOUNDER

© 2015 Hazelcast Inc. Confidential & Proprietary

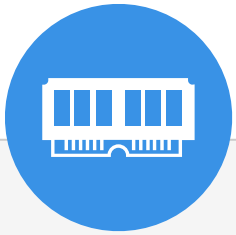
What Is Hazelcast?

Hazelcast is a distributed, highly available and scalable

Open Source In-Memory Data Grid

© 2015 Hazelcast Inc. Confidential & Proprietary

In Memory Data Grid



In Memory Data Storage

+

In Memory Data Messaging

+

In Memory Data Computing

© 2015 Hazelcast Inc. Confidential & Proprietary

java.util.Map

```
import java.util.HashMap;
import java.util.Map;


public static void main(String[] args) {
        Map<Integer, String> map = new HashMap<>();
        map.put(1, "Paris");
        map.put(2, "London");
        map.put(3, "San Francisco");

        String oldValue = map.remove(2);
}
```

© 2015 Hazelcast Inc. Confidential & Proprietary

java.util.concurrent.ConcurrentMap

```
import java.util.concurrent.ConcurrentHashMap;
import java.util.concurrent.ConcurrentMap;

public static void main(String[] args) {
        ConcurrentMap<Integer, String> map = new ConcurrentHashMap<>();
        map.put(1, "Paris");
        map.put(2, "London");
        map.put(3, "San Francisco");

        String oldValue = map.remove(2);
}
```

© 2015 Hazelcast Inc. Confidential & Proprietary

Distributed Map

```
import java.util.concurrent.ConcurrentMap;
import com.hazelcast.core.Hazelcast;
import com.hazelcast.core.HazelcastInstance;

public static void main(String[] args) {
      HazelcastInstance h = Hazelcast.newHazelcastInstance();

      ConcurrentMap<Integer, String> map = h.getMap("myMap");
      map.put(1, "Paris");
      map.put(2, "London");
      map.put(3, "San Francisco");

      String oldValue = map.remove(2);
}
```

© 2015 Hazelcast Inc. Confidential & Proprietary

Ecosystem Traction

Dozens of Commercial and Open Source Projects Embed Hazelcast

© 2015 Hazelcast Inc. Confidential & Proprietary

Demo

© 2015 Hazelcast Inc. Confidential & Proprietary

Why Hazelcast?

Scale-out Computing enables cluster capacity to be increased or decreased on-demand

Resilience with automatic recovery from member failures without losing data while minimizing performance impact on running applications

Programming Model provides a way for developers to easily program a cluster application as if it is a single process

Fast Application Performance enables very large data sets to be held in main memory for real-time performance

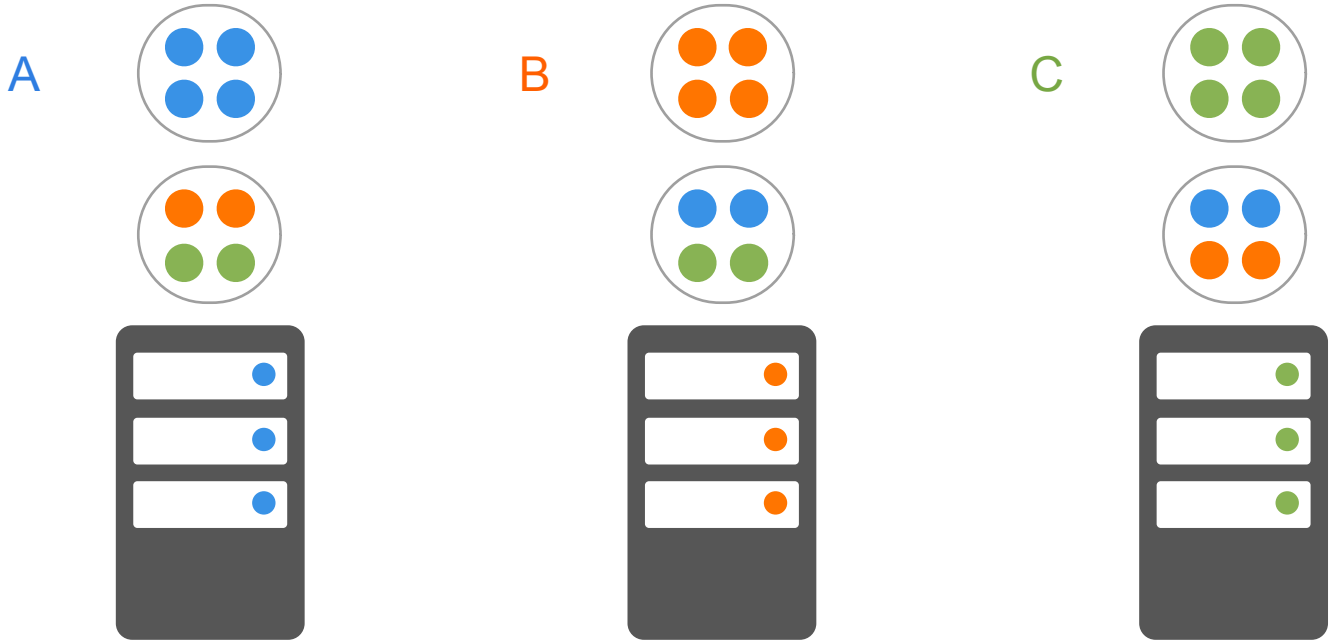
© 2015 Hazelcast Inc. Confidential & Proprietary

Rebalance Data on New Node

© 2015 Hazelcast Inc. Confidential & Proprietary

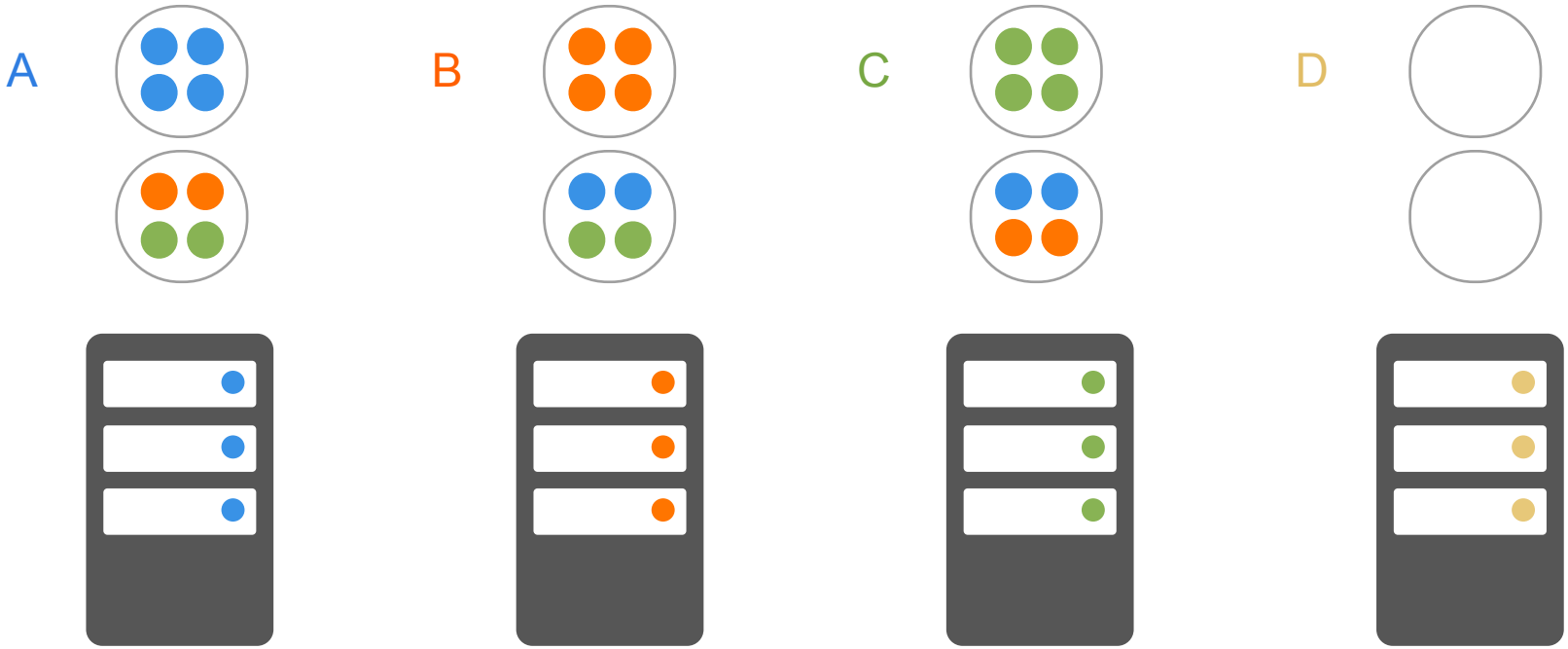
Distributed Maps

Fixed number of partitions (default 271)
Each key falls into a partition
partitionId = hash(keyData)%PARTITION_COUNT
Partition ownerships are reassigned upon membership change



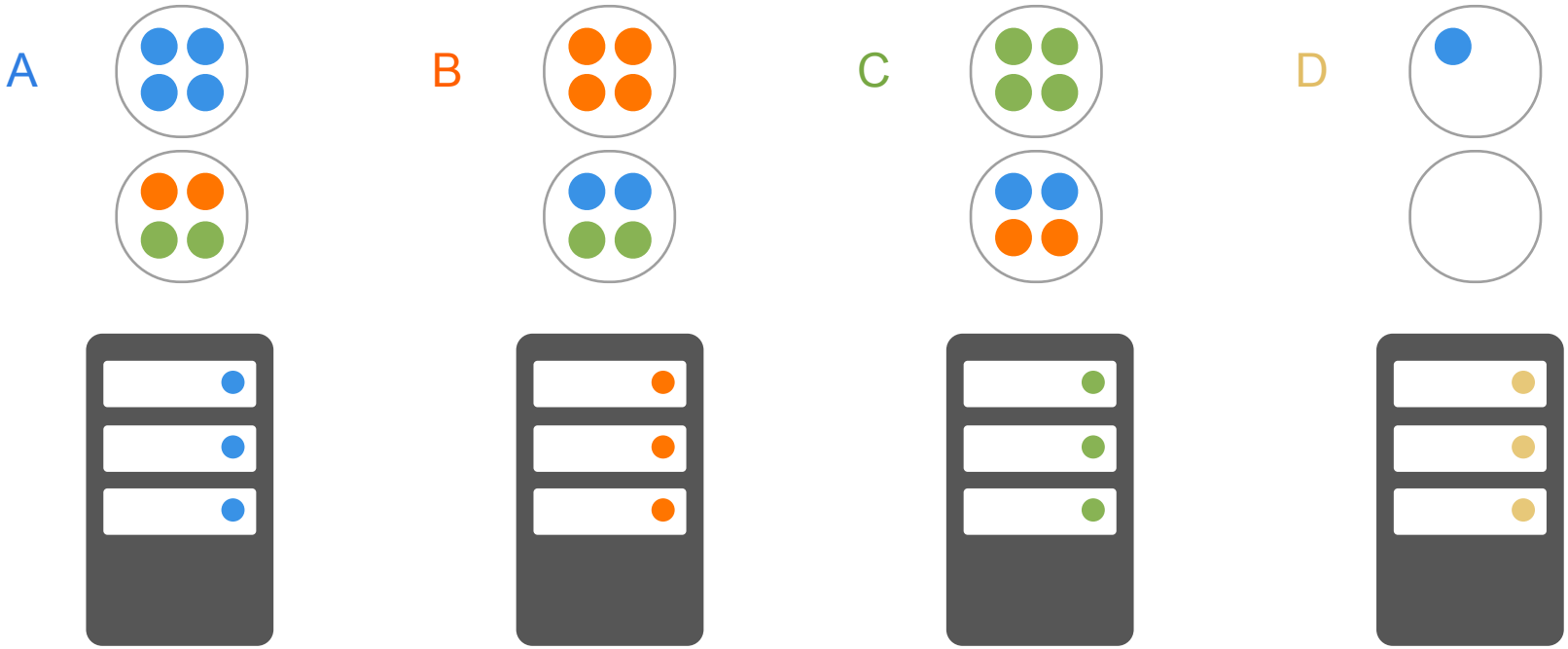
© 2015 Hazelcast Inc. Confidential & Proprietary

New Node Added



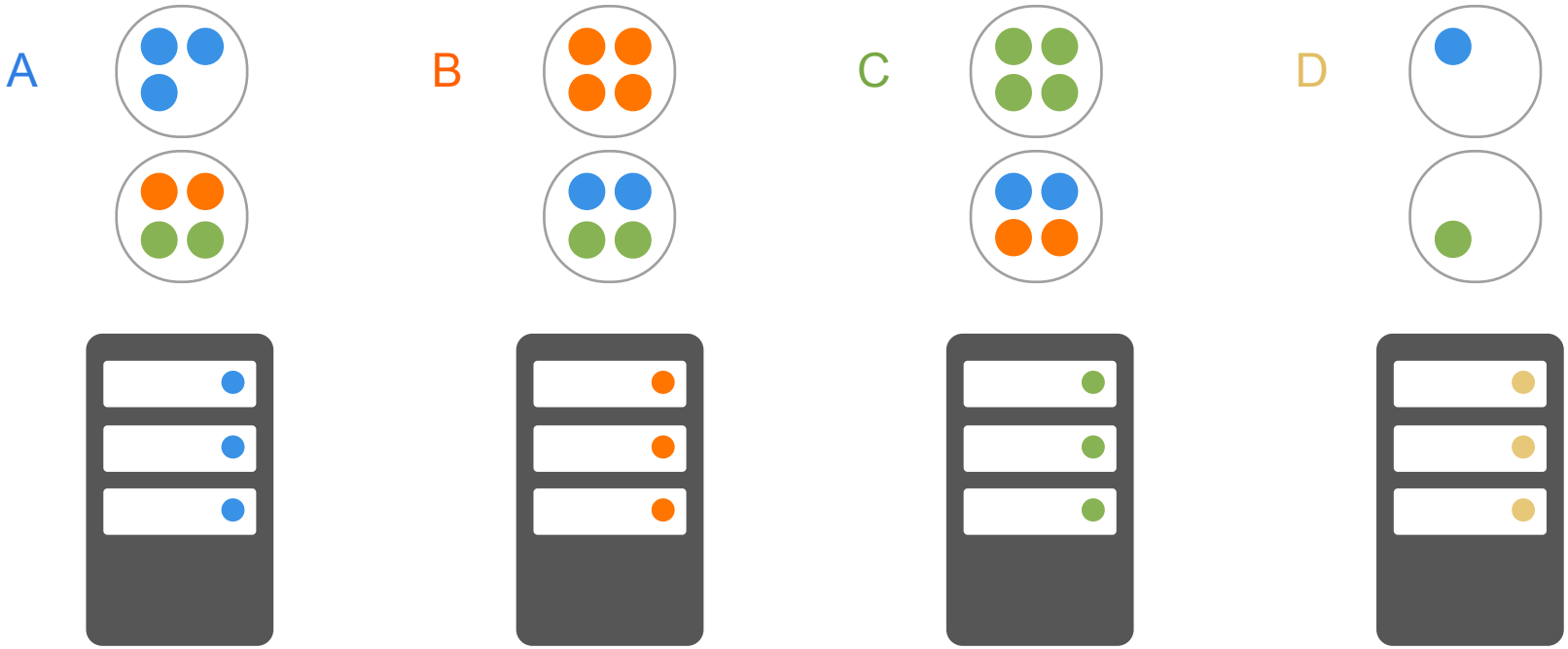
© 2015 Hazelcast Inc. Confidential & Proprietary

Migration



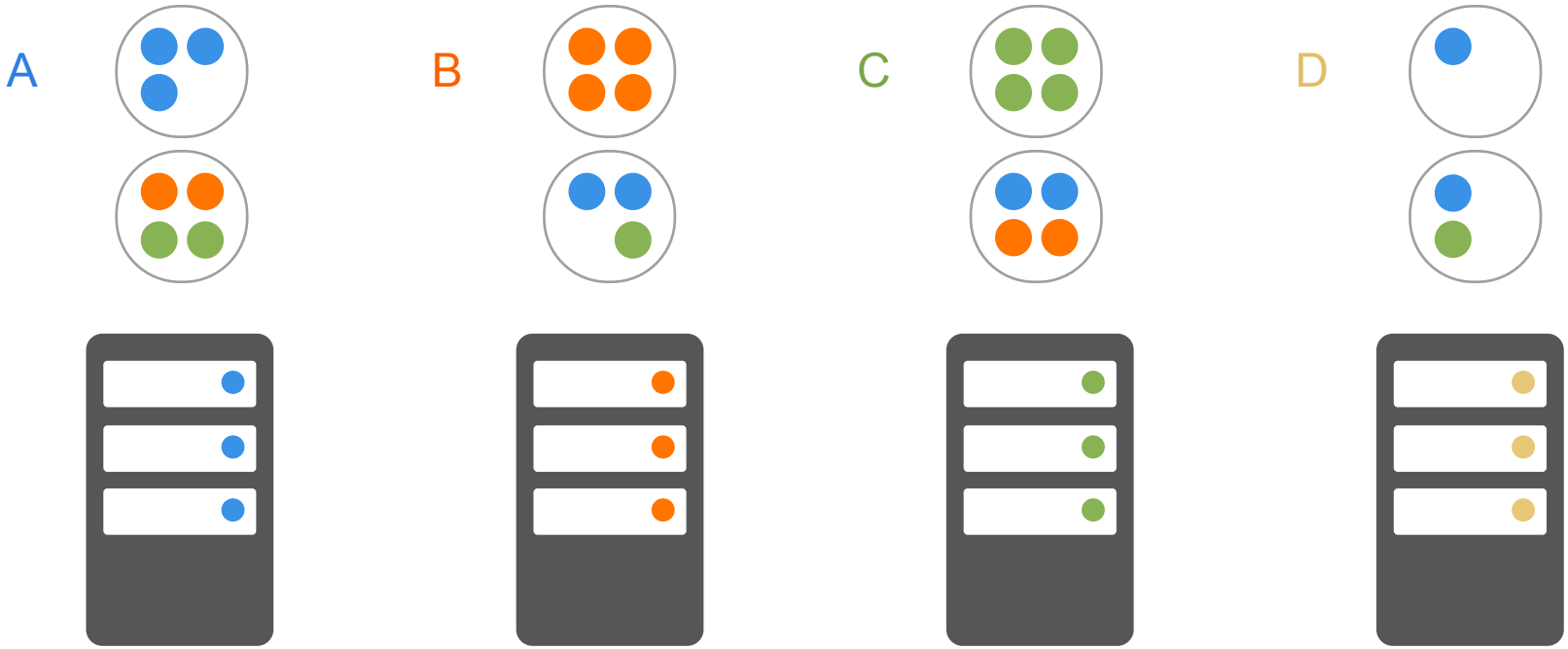
© 2015 Hazelcast Inc. Confidential & Proprietary

Migration



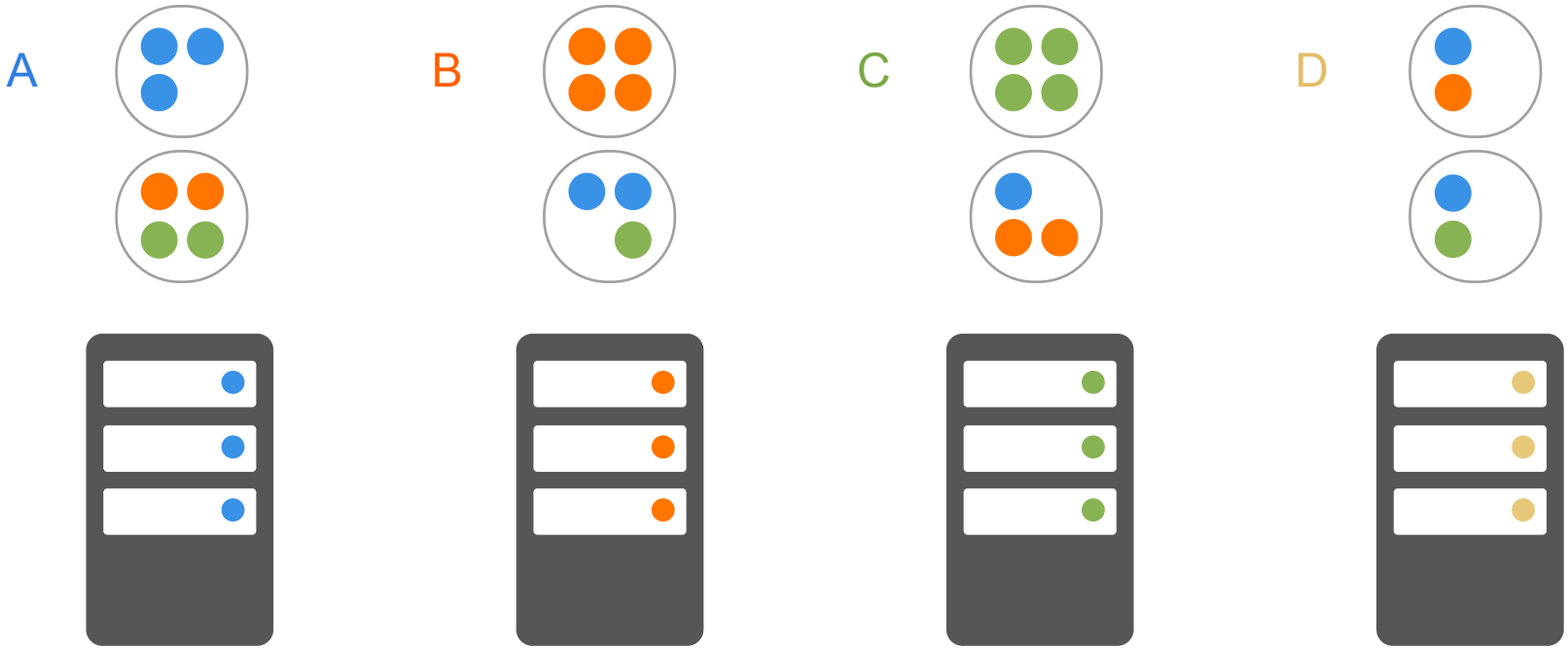
© 2015 Hazelcast Inc. Confidential & Proprietary

Migration



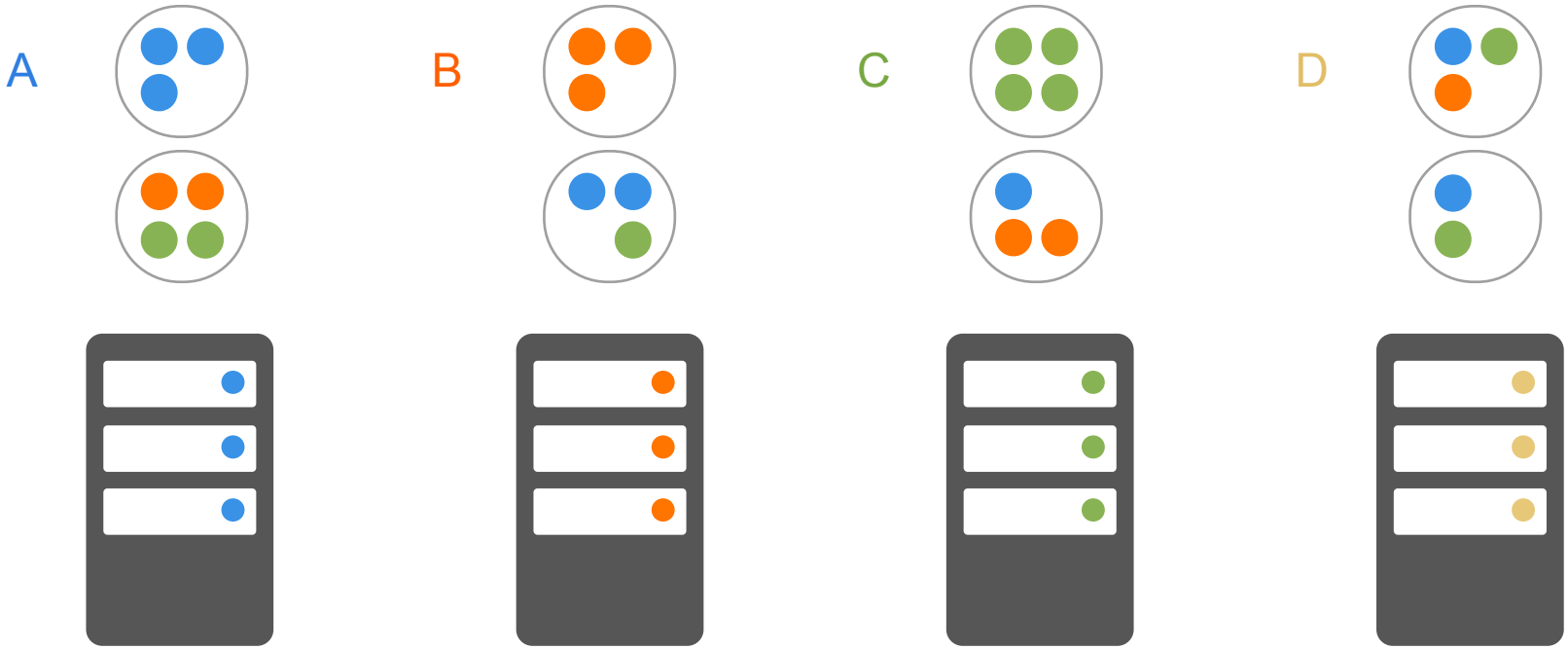
© 2015 Hazelcast Inc. Confidential & Proprietary

Migration



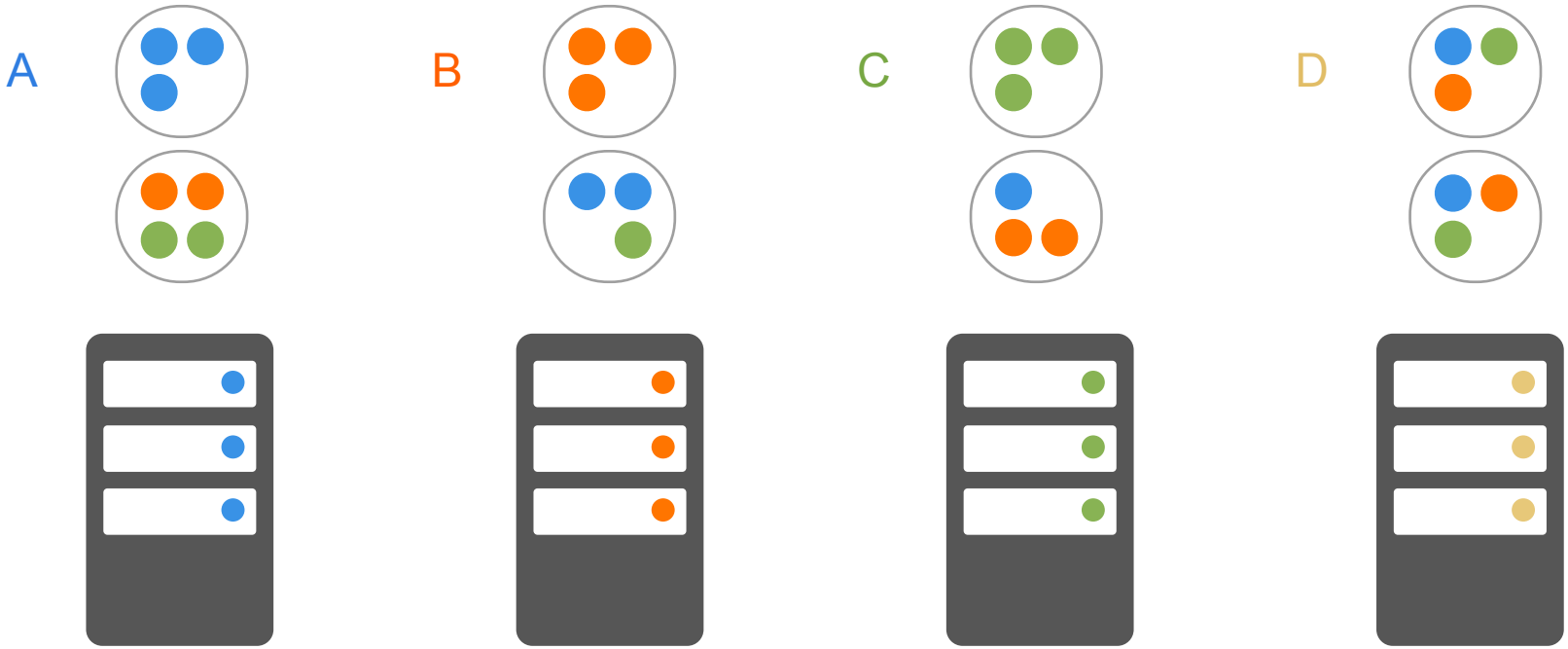
© 2015 Hazelcast Inc. Confidential & Proprietary

Migration



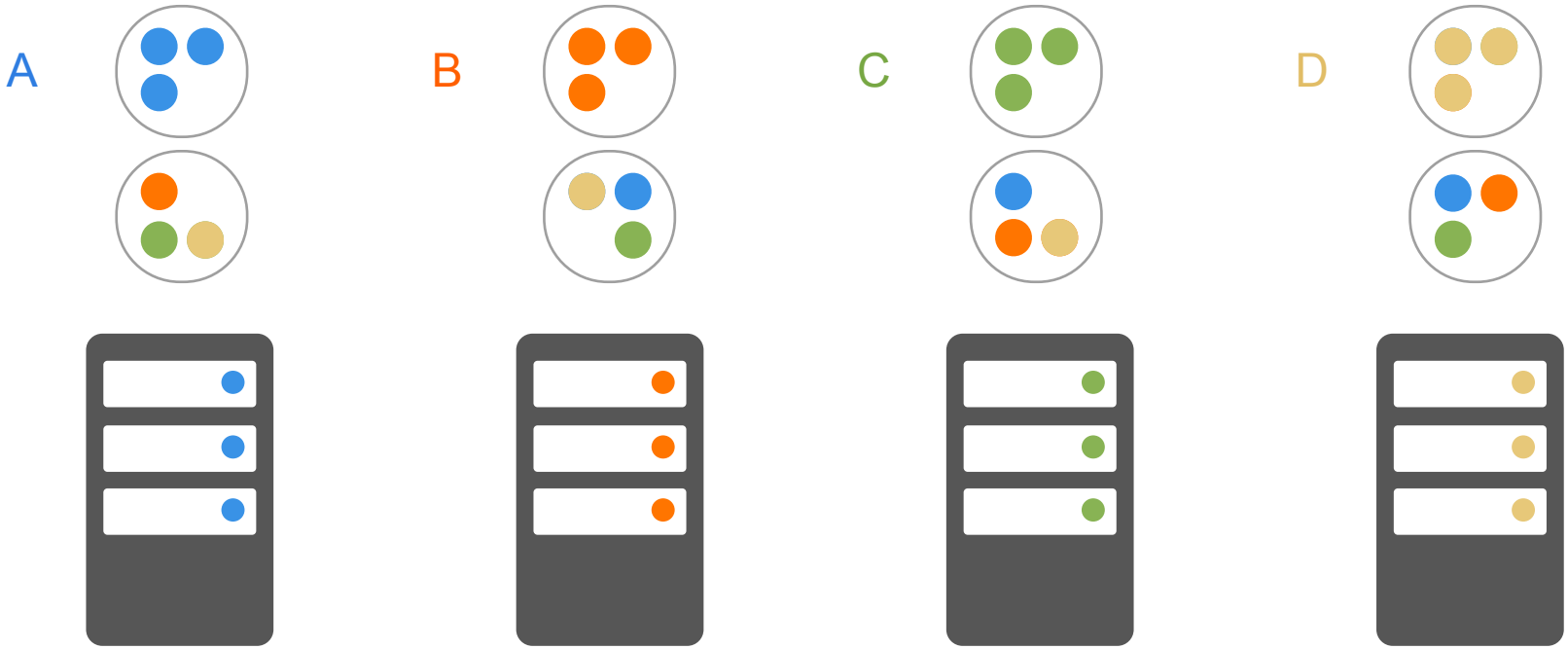
© 2015 Hazelcast Inc. Confidential & Proprietary

Migration



© 2015 Hazelcast Inc. Confidential & Proprietary

Migration Complete

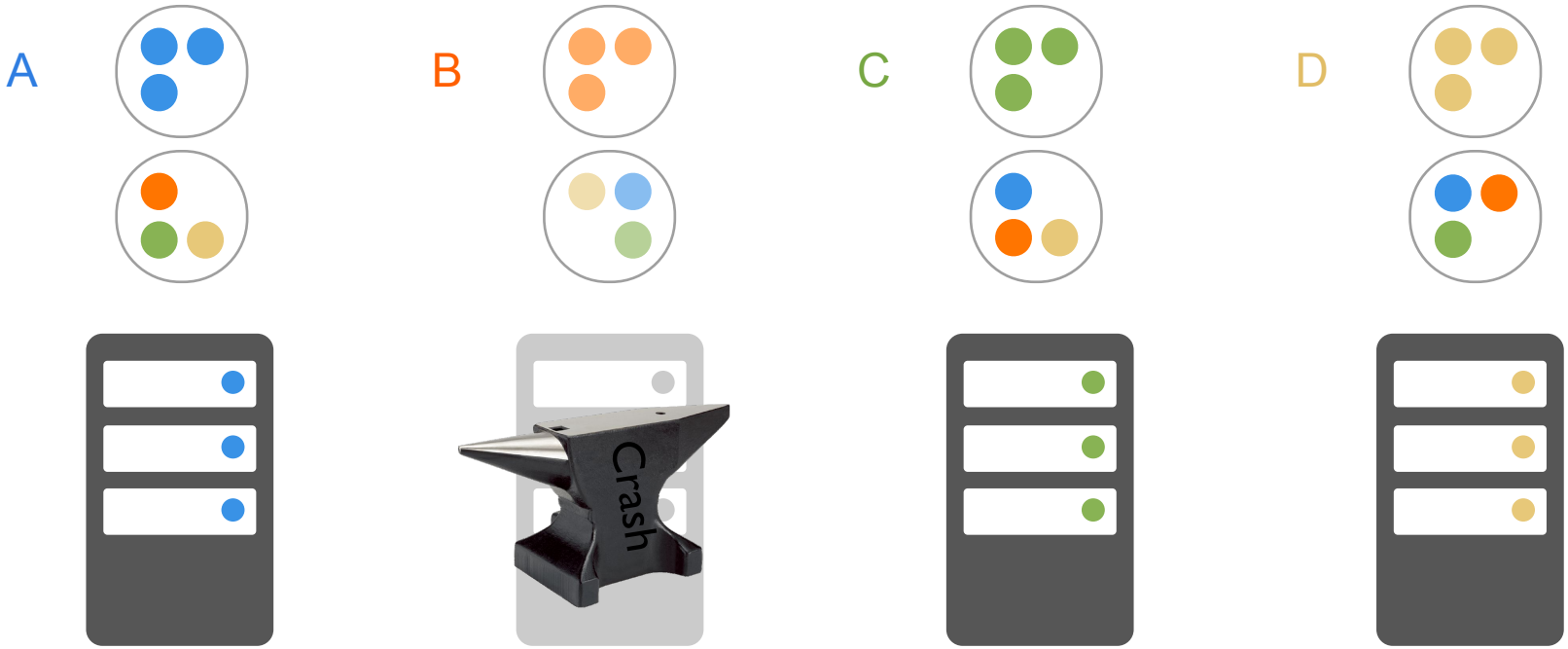


© 2015 Hazelcast Inc. Confidential & Proprietary

Data Safety when Node Dies

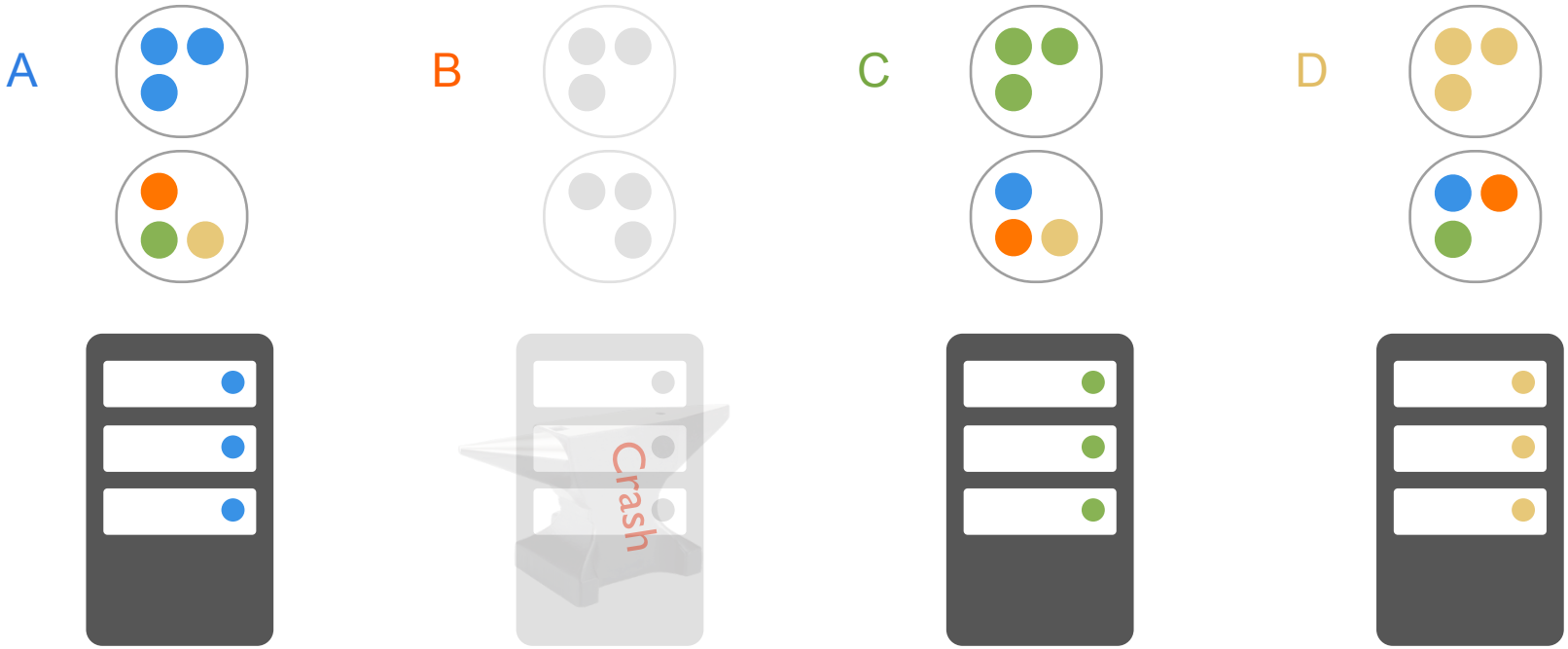
© 2015 Hazelcast Inc. Confidential & Proprietary

Node Crashes



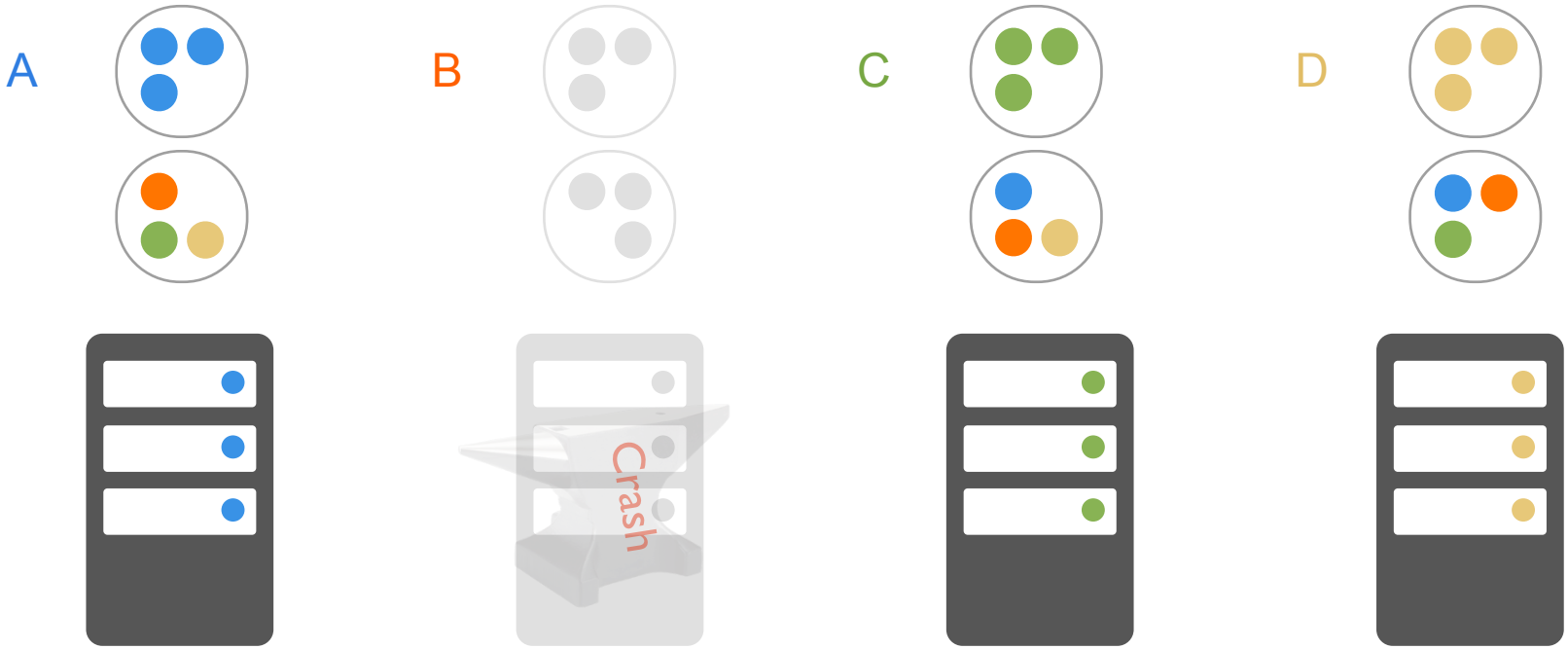
© 2015 Hazelcast Inc. Confidential & Proprietary

Backups Are Restored



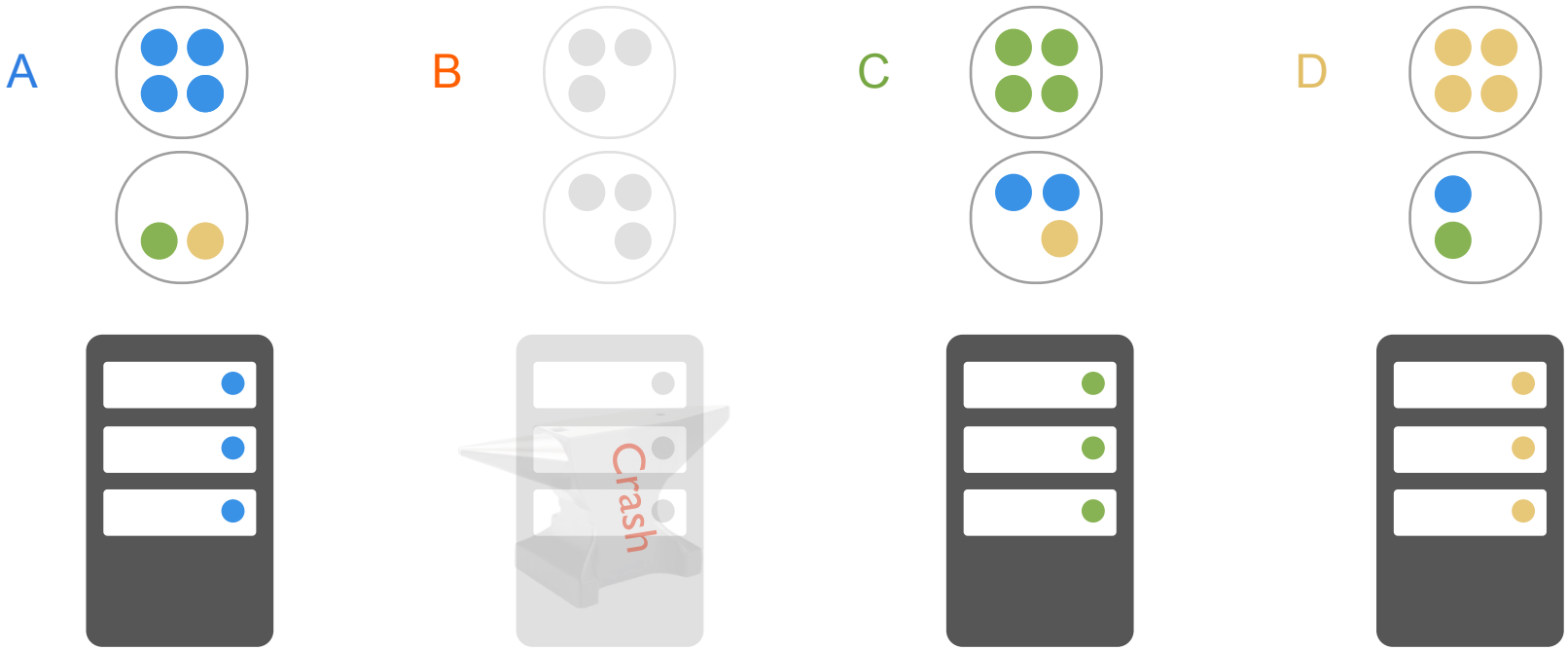
© 2015 Hazelcast Inc. Confidential & Proprietary

Backups Are Restored



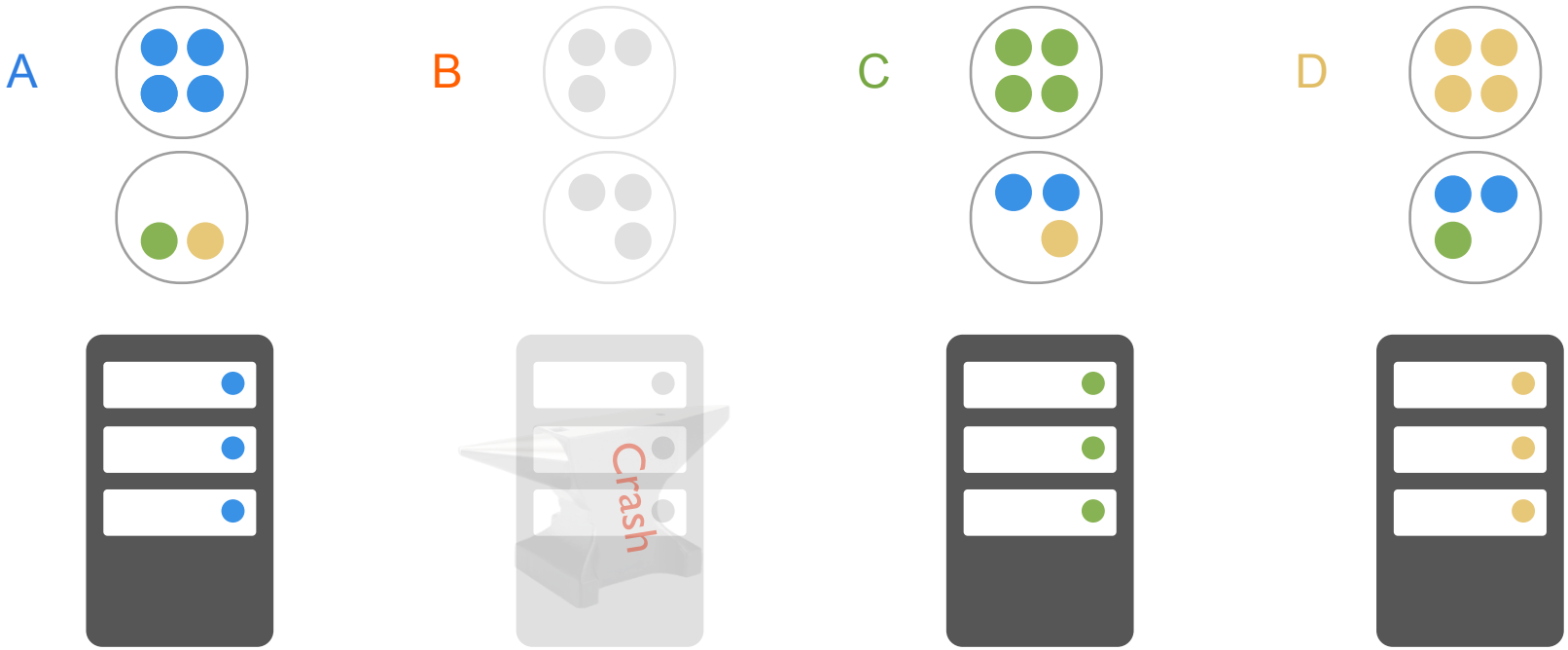
© 2015 Hazelcast Inc. Confidential & Proprietary

Backups Are Restored



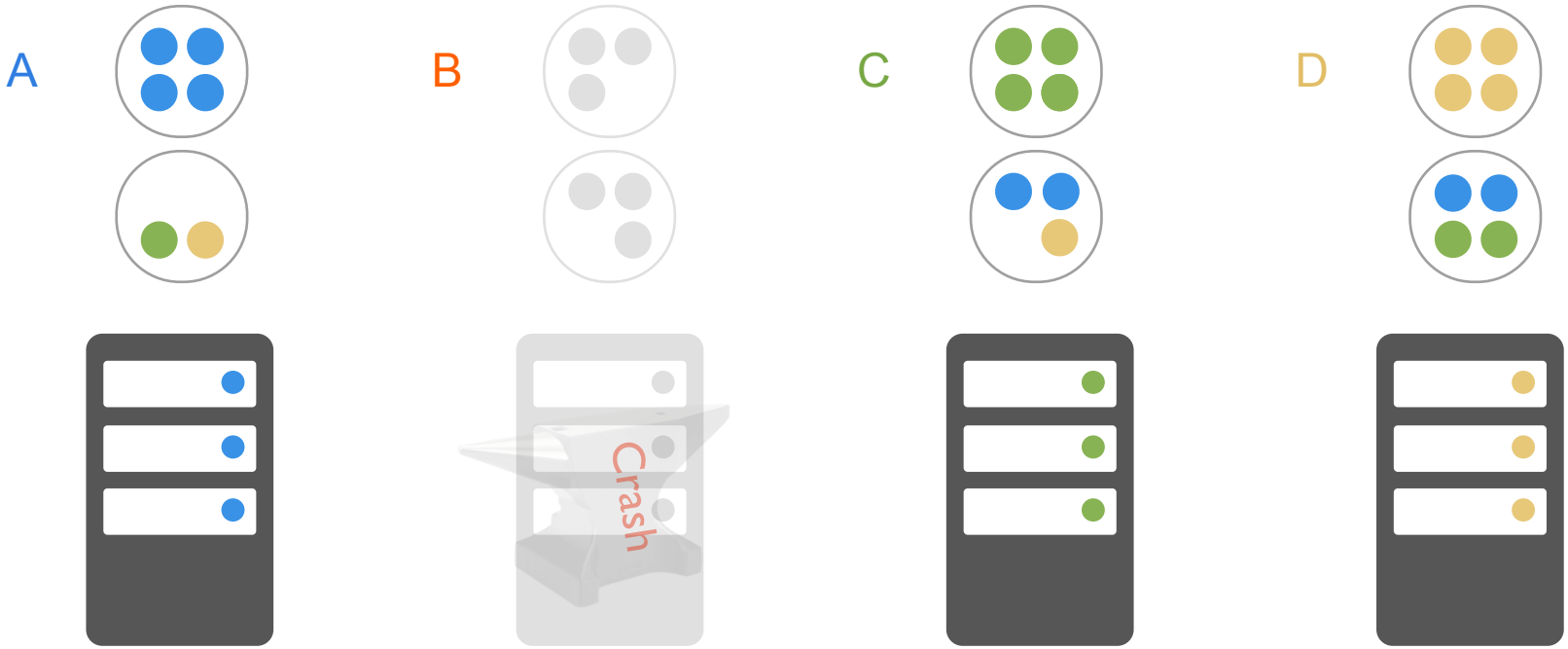
© 2015 Hazelcast Inc. Confidential & Proprietary

Backups Are Restored



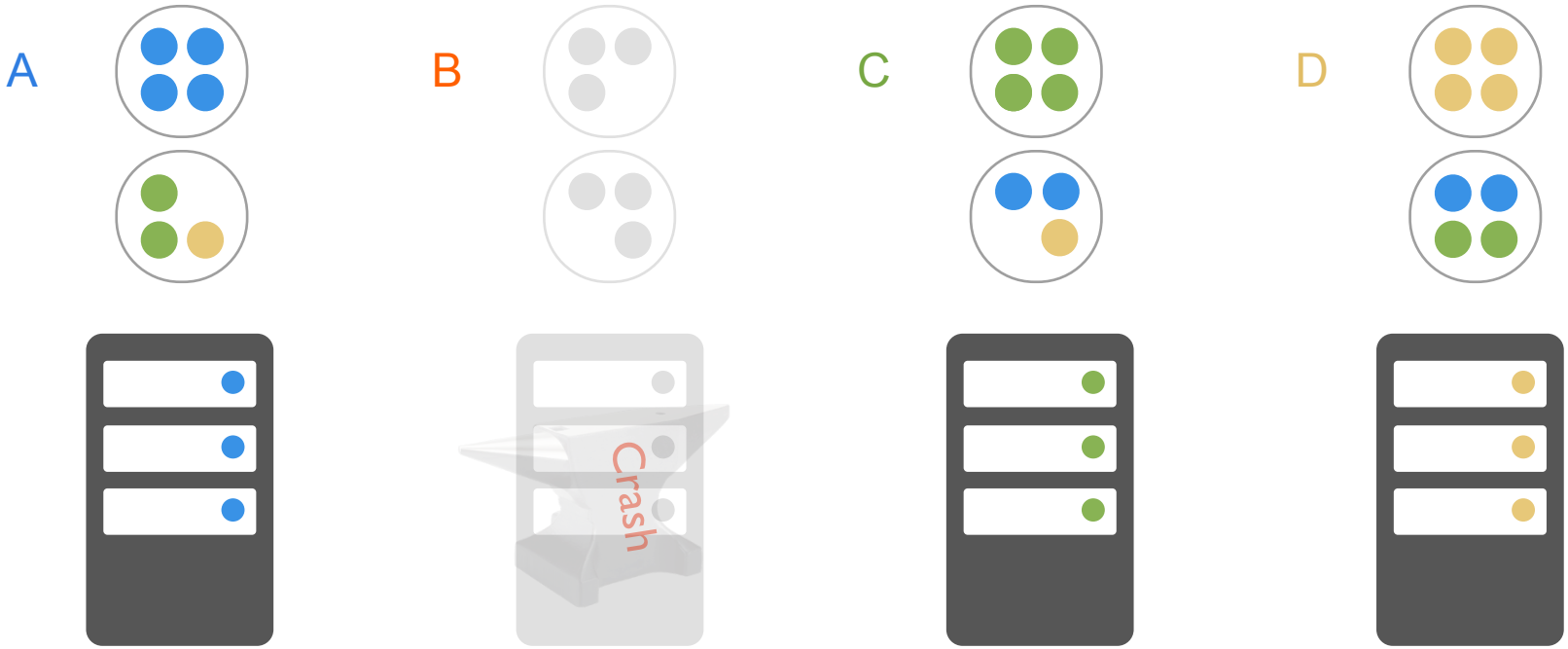
© 2015 Hazelcast Inc. Confidential & Proprietary

Backups Are Restored



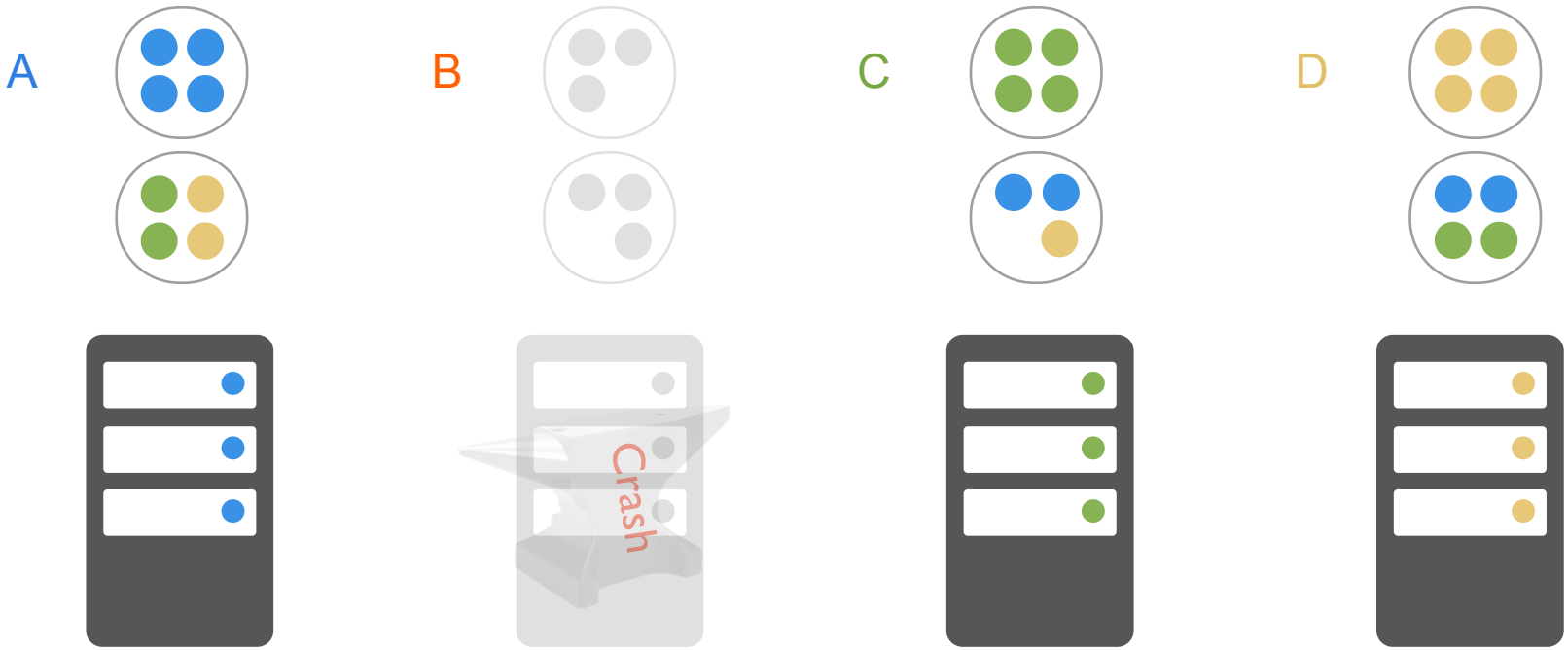
© 2015 Hazelcast Inc. Confidential & Proprietary

Backups Are Restored



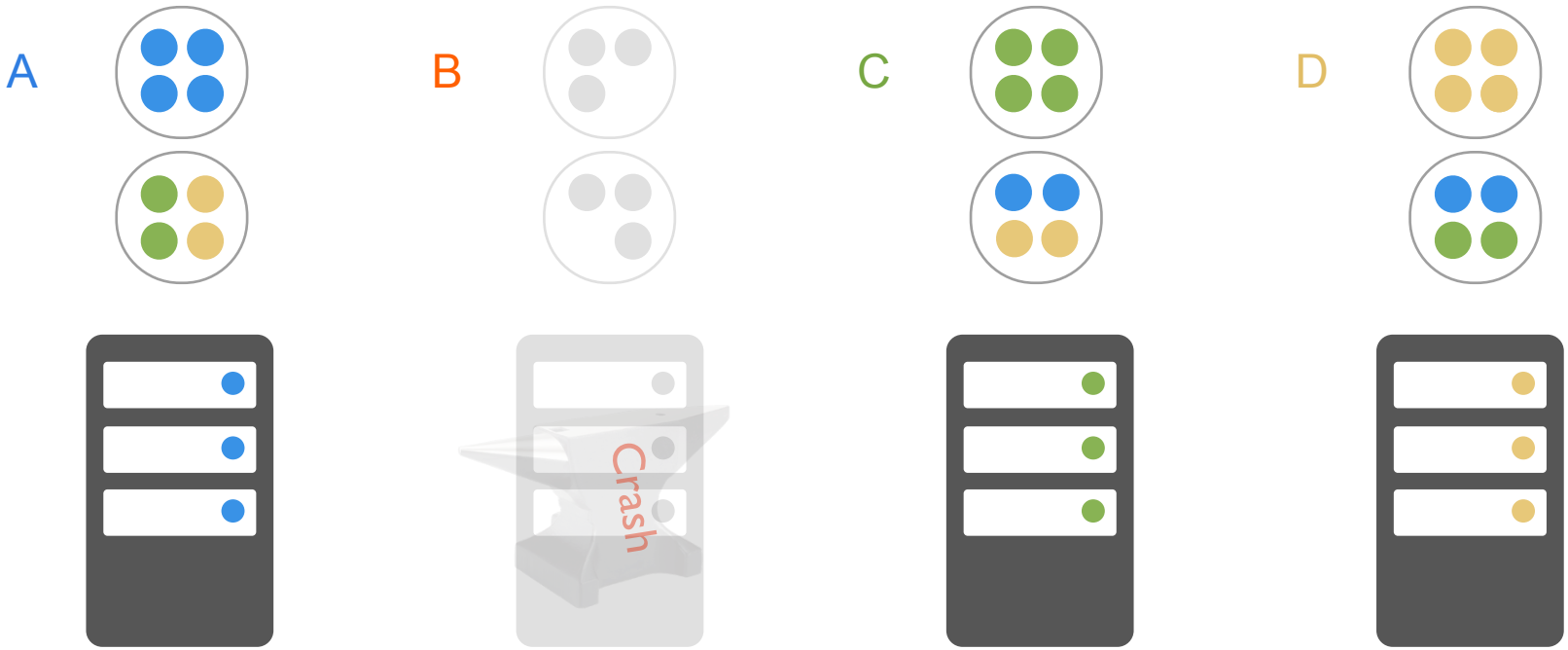
© 2015 Hazelcast Inc. Confidential & Proprietary

Backups Are Restored



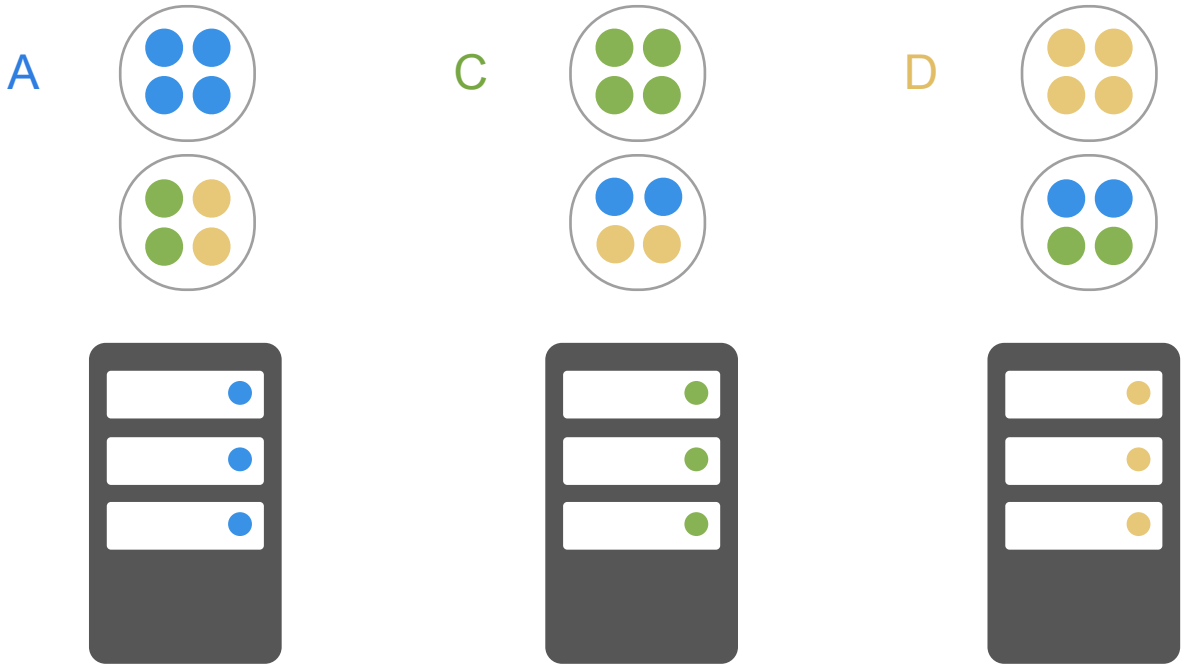
© 2015 Hazelcast Inc. Confidential & Proprietary

Backups Are Restored



© 2015 Hazelcast Inc. Confidential & Proprietary

Recovery Is Complete



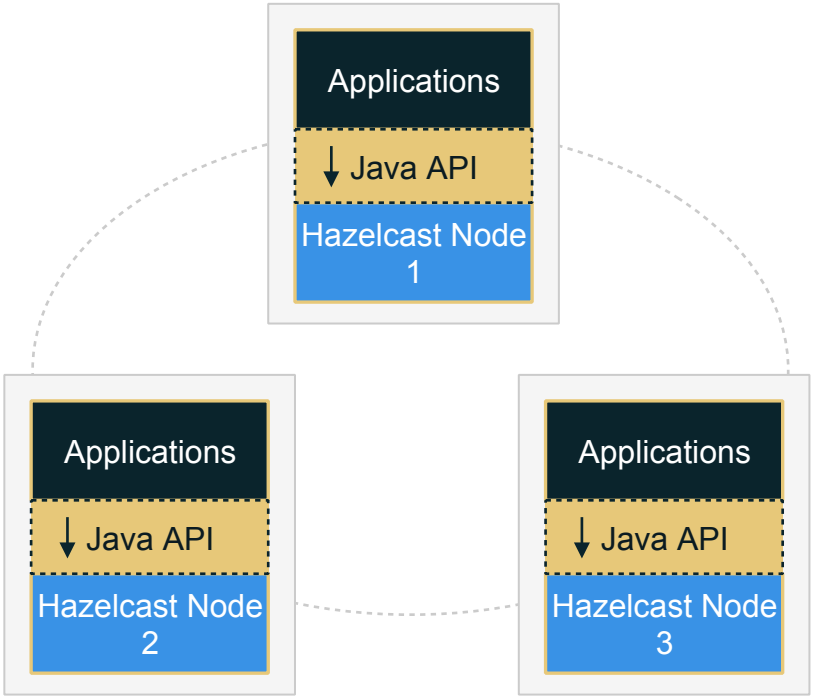
© 2015 Hazelcast Inc. Confidential & Proprietary

Deployment Strategies

© 2015 Hazelcast Inc. Confidential & Proprietary

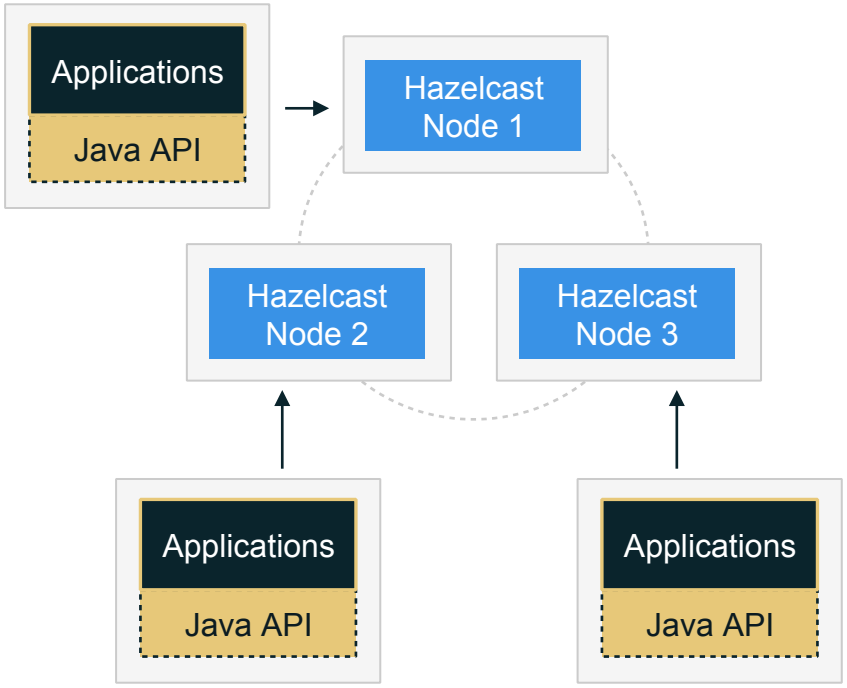
Deployment Options

Embedded Hazelcast



Great for early stages of rapid application development and iteration

Client-Server Mode



Necessary for scale up or scale out deployments – decouples upgrading of clients and cluster for long term TCO

© 2015 Hazelcast Inc. Confidential & Proprietary

Easy API

```
// Creating a new Hazelcast node
HazelcastInstance hz = Hazelcast.newHazelcastInstance();


// Getting a Map, Queue, Topic, ...
Map map = hz.getMap("my-map");
Queue queue = hz.getQueue("my-queue");
ITopic topic = hz.getTopic("my-topic");

//Creating a Hazelcast Client
HazelcastInstance client = Hazelcast.newHazelcastClient();

// Shutting down the node
hz.shutdown();
```

© 2015 Hazelcast Inc. Confidential & Proprietary

Feature Overview

© 2015 Hazelcast Inc. Confidential & Proprietary

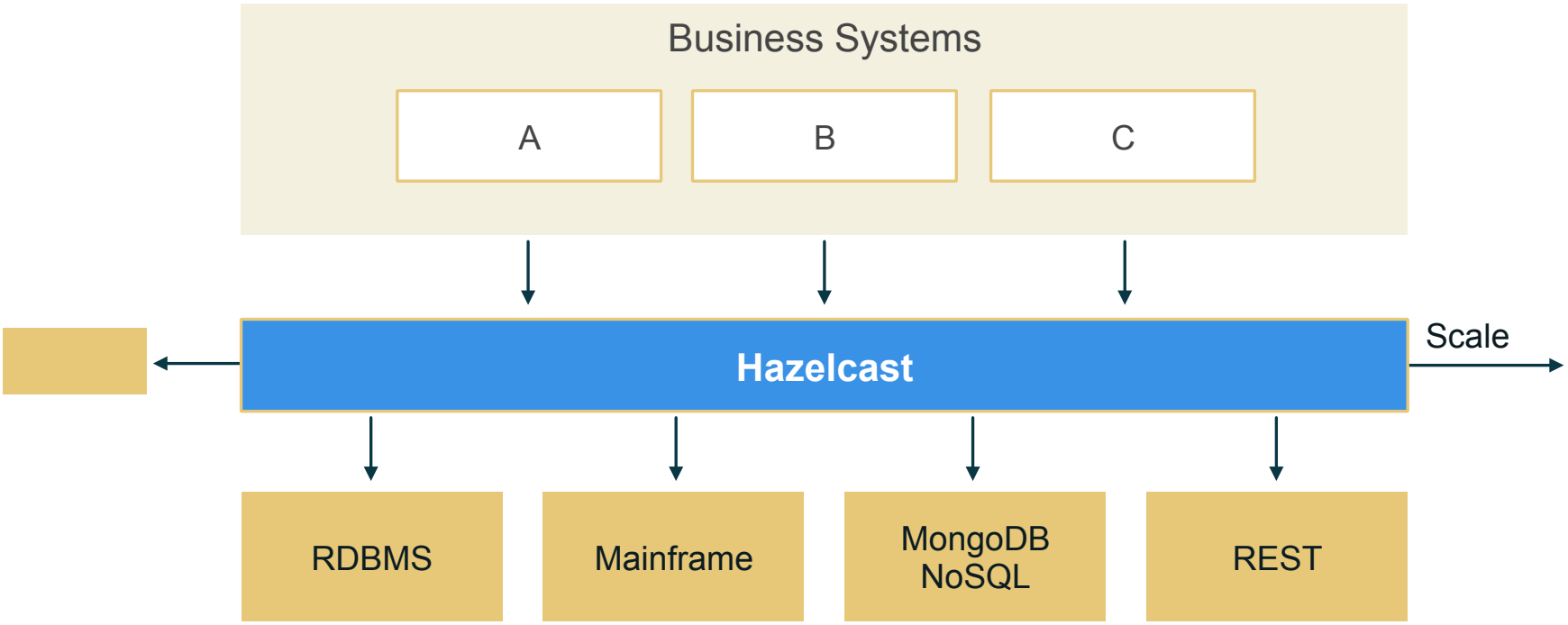
Easy to Unit Test

```
public class SomeTestCase {

  private HazelcastInstance[] instances;

  @Before
  public void before() throws Exception {
    // Multiple instances on the same JVM
    instances = new HazelcastInstance[2];
    instances[0] = Hazelcast.newHazelcastInstance();
    instances[1] = Hazelcast.newHazelcastInstance();
  }

  @After
  public void after() throws Exception {
    Hazelcast.shutdownAll();
  }
}
```

© 2015 Hazelcast Inc. Confidential & Proprietary

IM Data Store (Caching) Use Case

Database Caching Use-Case



© 2015 Hazelcast Inc. Confidential & Proprietary

IM Data Store (Caching) Features

HD Cache | Dist. Compute | Dist. Message

Java Collection API: Map, List, Set, Queue

JCache

High Density Memory Store

Hibernate 2nd Level Cache

Web Session Replication: Tomcat, Jetty

Predicate API: Indexes, SQL Query

Persistence: Map/Queue Store & Loader. Write Behind/Through

Eviction

Near Cache

Transactions: Local & XA

WAN & DR Replication

Memcached Interface

© 2015 Hazelcast Inc. Confidential & Proprietary

Map API

```
interface com.hazelcast.core.IMap<K, V>
            extends java.util.Map, java.util.ConcurrentMap

HazelcastInstance hz = getHazelcastInstance();

//java.util.concurrent.ConcurrentMap implementation
IMap<String, User> hzMap = hz.getMap("users");
hzMap.put("Peter", new User("Peter", "Veentjer"));

hzMap.putIfAbsent("Peter", new User("Peter", "Veentjer"));

//Distributed Lock
hzMap.lock("Peter");


User peter = map.get("Peter");
```

© 2015 Hazelcast Inc. Confidential & Proprietary

Persistence API

```
public class MapStorage
    implements MapStore<String, User>, MapLoader<String, User> {
  // Some methods missing ...
  @Override public User load(String key) { return loadValueDB(key); }
  @Override public Set<String> loadAllKeys() { return loadKeysDB(); }
  @Override public void delete(String key) { deleteDB(key); }
  @Override public void store(String key, User value) {
    storeToDatabase(key, value);
  }
}

<map name="users">
  <map-store enabled="true">
    <class-name>com.hazelcast.example.MapStorage</class-name>
    <write-delay-seconds>0</write-delay-seconds>
  </map-store>
</map>
```

© 2015 Hazelcast Inc. Confidential & Proprietary

JCache API

```
// Retrieve the CachingProvider which is automatically baced by
// the chosen Hazelcast server or client provider
CachingProvider cachingProvider = Caching.getCachingProvider();

// Create a CacheManager
CacheManager cacheManager = cachingProvider.getCacheManager();

// Cache<String, String> cache = cacheManager
//     .getCache( name, String.class, String.class );

// Create a simple but typesafe configuration for the cache

CompleteConfiguration<String, String> config =
    new MutableConfiguration<String, String>()
        .setTypes( String.class, String.class );
```

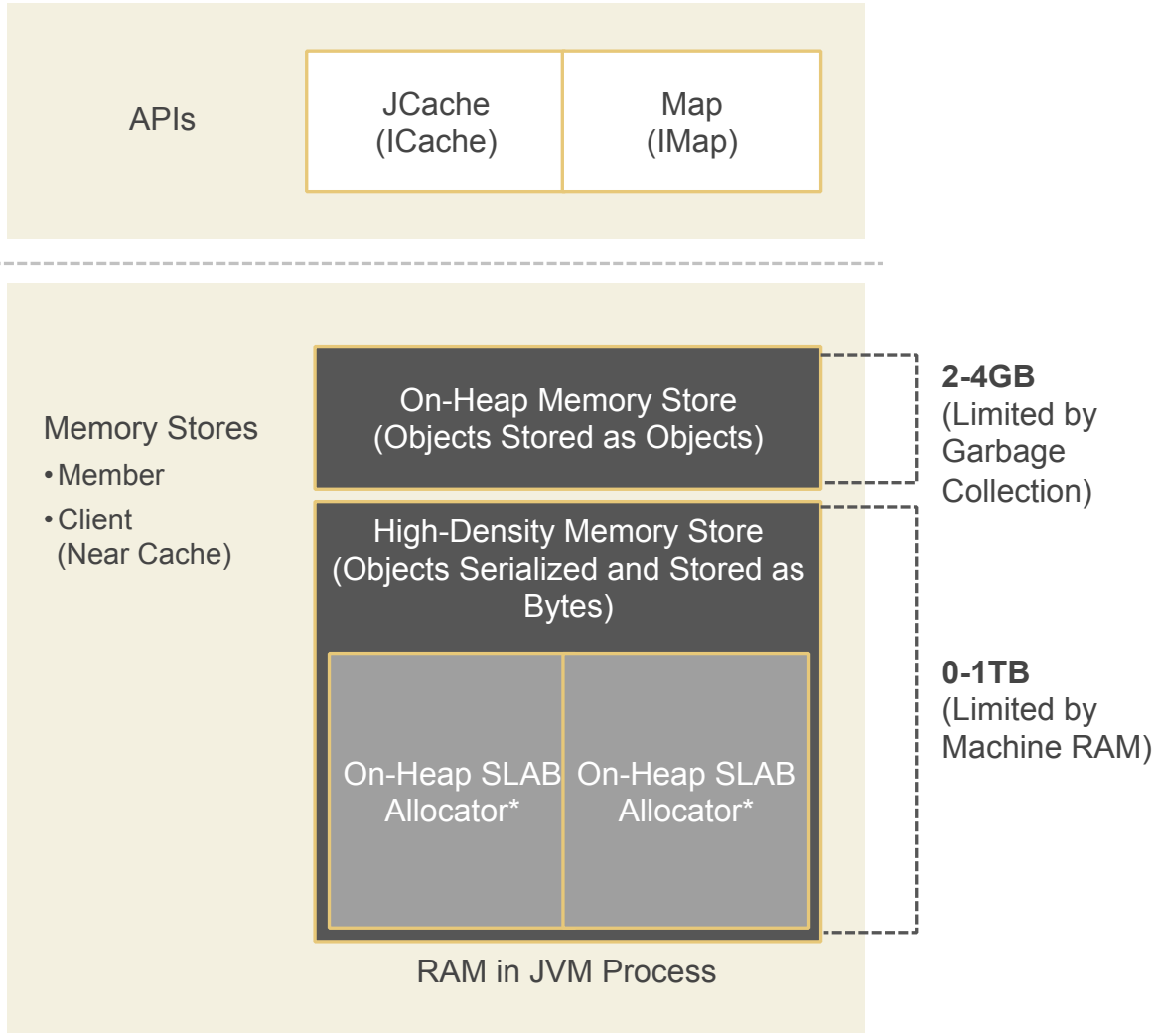
© 2015 Hazelcast Inc. Confidential & Proprietary

JCache API

```
// Create and get the cache
Cache<String, String> cache = cacheManager
        .createCache( "example", config );
// Alternatively to request an already existing cache
Cache<String, String> cache = cacheManager
     .getCache( name, String.class, String.class );

// Put a value into the cache
cache.put( "world", "Hello World" );

// Retrieve the value again from the cache
String value = cache.get( "world" );

System.out.println( value );
```

© 2015 Hazelcast Inc. Confidential & Proprietary

High Density Caching

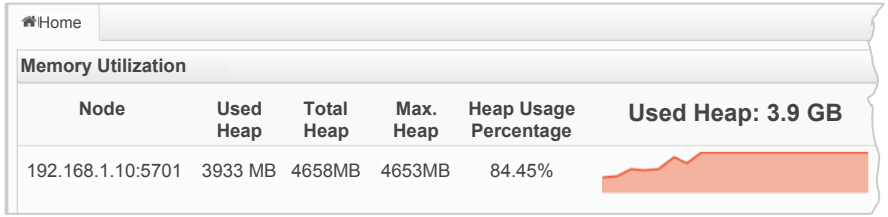


* coming in 3.6

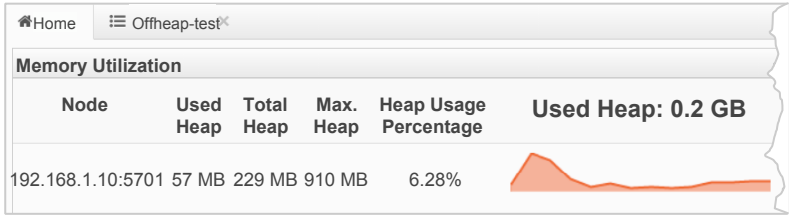
© 2015 Hazelcast Inc. Confidential & Proprietary

On Heap Vs. High-Density Memory Management

On Heap Memory		HD Memory v2
0 MB	*Native*	*3.3 GB*
3.9 GB	*Heap Storage*	*0.6 GB*
9 (4900 ms)	*Major GC*	*0 (0 ms)*
31 (4200 ms)	*Minor GC*	*356 (349 ms)*



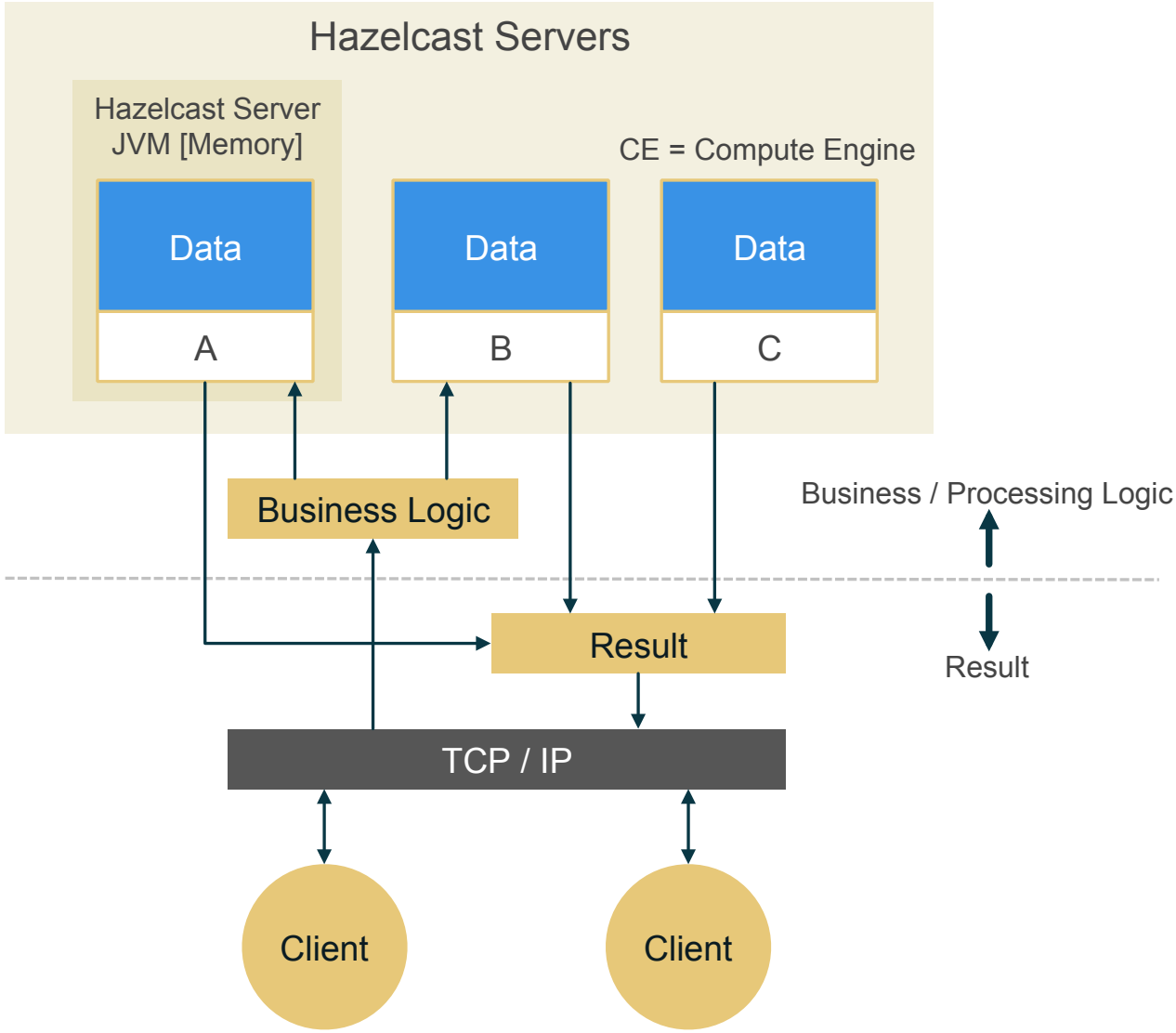
Example: On Heap Memory



Example: HD Memory v2

© 2015 Hazelcast Inc. Confidential & Proprietary

IM Distributed Computing Use Case



© 2015 Hazelcast Inc. Confidential & Proprietary

IM Distributed Computing Feature

Java Concurrency API
(Lock, Semaphore, AtomicLong, AtomicReference, Executor Service, Blocking Queue)

Entry and Item Listeners

Entry Processor

Aggregators

Map/Reduce

Data Affinity

Continues Query

Map Interceptors

Delta Update

© 2015 Hazelcast Inc. Confidential & Proprietary

Lock API

```
HazelcastInstance hz = getHazelcastInstance();

// Distributed Reentrant
Lock lock = hz.getLock("myLock");
lock.lock();
try {
  // Do something
} finally {
  lock.unlock();
}
```

© 2015 Hazelcast Inc. Confidential & Proprietary

Executor Service API

```
public interface com.hazelcast.core.IExecutorService
        extends java.util.concurrent.ExecutorService

HazelcastInstance hz = getHazelcastInstance();

//java.util.concurrent.ExecutorService implementation
IExecutorService es = hz.getExecutorService("name");
es.executeOnAllMembers(buildRunnable());
es.executeOnKeyOwner(buildRunnable(), "Peter");
es.execute(buildRunnable());

Map<..> futures = es.submitToAllMembers(buildCallable());
Future<..> future = es.submitToKeyOwner(buildCallable(), "Peter");

es.submitToAllMembers(buildCallable(), buildCallback());
es.submitToKeyOwner(buildCallable(), "Peter", buildCallback());
```

© 2015 Hazelcast Inc. Confidential & Proprietary

Map/Reduce API

```
HazelcastInstance hz = getHazelcastInstance();

Map users = hz.getMap("users");
JobTracker tracker = hz.getJobTracker("default");

KeyValueSource source = KeyValueSource.fromMap(users);
Job job = tracker.newJob(source);

ICompleteFuture future = job.mapper(new MyMapper())
                            .reducer(new MyReducer())
                            .submit();

Map result = future.get();
```

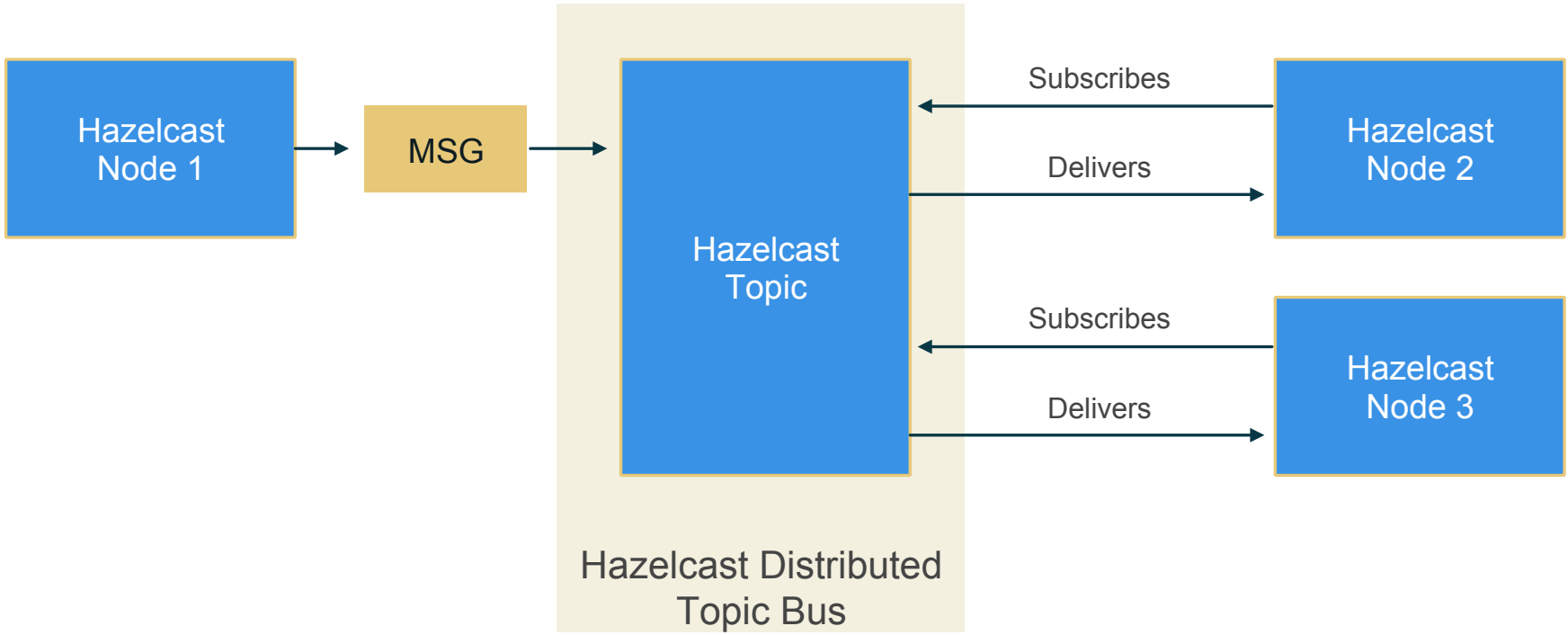
© 2015 Hazelcast Inc. Confidential & Proprietary

Aggregations API

```
HazelcastInstance hz = getHazelcastInstance();

Map users = hz.getMap("users");

int sum = users.aggregate(
    Supplier.all((user) -> user.getSalary()),
    Aggregations.longSum()
);
```

© 2015 Hazelcast Inc. Confidential & Proprietary

IM Distributed Messaging Use Case



© 2015 Hazelcast Inc. Confidential & Proprietary

IM Distributed Messaging Features

Queue

Topic (Pub/Sub)

Event Listeners

Ring Buffers

© 2015 Hazelcast Inc. Confidential & Proprietary

Queue API

```
interface com.hazelcast.core.IQueue<E>
        extends java.util.concurrent.BlockingQueue

HazelcastInstance hz = getHazelcastInstance();

//java.util.concurrent.BlockingQueue implementation
IQueue<Task> queue = hz.getQueue("tasks");

queue.offer(newTask());
queue.offer(newTask(), 500, TimeUnit.MILLISECONDS);

Task task = queue.poll();
Task task = queue.poll(100, TimeUnit.MILLISECONDS);
Task task = queue.take();
```

© 2015 Hazelcast Inc. Confidential & Proprietary

Topic API

```
public class Example implements MessageListener<String> {
  public void sendMessage {
    HazelcastInstance hz = getHazelcastInstance();
    ITopic<String> topic = hz.getTopic("topic");
    topic.addMessageListener(this);
    topic.publish("Hello World");
  }

  @Override
  public void onMessage(Message<String> message) {
    System.out.println("Got message: " + message.getMessageObject());
  }
}
```

© 2015 Hazelcast Inc. Confidential & Proprietary

Thank you

@fuadm, @hazelcast
hazelcast@googlegroups.com
http://www.hazelcast.com
http://github.com/hazelcast/hazelcast