Clustering your application with Hazelcast

Talip Ozturk
talip@hazelcast.com

Who we are

- Founded in 2008
- Startup company founded by

  Talip Ozturk and Fuad Malikov
- Open source business model. Consulting and support on Hazelcast.
- Hundreds of users. Mostly in US and Europe.
- References available upon request.

Hazelcast Events

01/14/2010 Rhein JUG	Dusseldorf, Germany
09/10/2009 Java2Days	Sofia, Bulgaria
06/17/2009 NY JavaSIG	New York, USA
06/10/2009 Kansas JUG	Kansas, USA
06/09/2009 San Francisco JUG	San Francisco, USA
05/27/2009 Houston JUG	Houston, USA
05/26/2009 Tampa JUG	Tampa, USA
04/29/2009 Czech JUG	Prague, Czech Republic
04/23/2009 Luxembourg JUG	Luxembourg City, Luxembourg
04/21/2009 London JAVAWUG	London, UK
04/03/2009 Genoa University	Genoa, Italy
04/02/2009 Roma JUG	Roma, Italy
04/01/2009 Vienna JSUG	Vienna, Austria
03/19/2009 TSSJS - Project Report	Las Vegas, USA

Agenda

- Demo
- Introduction
- Code Samples
- Internals
- Q/A

Map

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

Map map = new HashMap();

map.put("1", "value");
map.get("1");
```

Concurrent Map

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

Map map = new ConcurrentHashMap();

map.put("1", "value");
map.get("1");
```

Distributed Map

```
import java.util.Map;
import com.hazelcast.core.Hazelcast;

Map map = Hazelcast.getMap("mymap");

map.put("1", "value");
map.get("1");
```

Why Hazelcast?

- Scale your application
- Share data across cluster
- Partition your data
- Send/receive messages
- Balance the load
- Process in parallel on many JVM

Solutions in the Market

- Oracle Coherence
- IBM WebSphere eXtreme Scale / ObjectGrid
- Terracotta
- Gigaspaces
- Gemstone
- JBossCache/JGroups/Infinispan

Difference

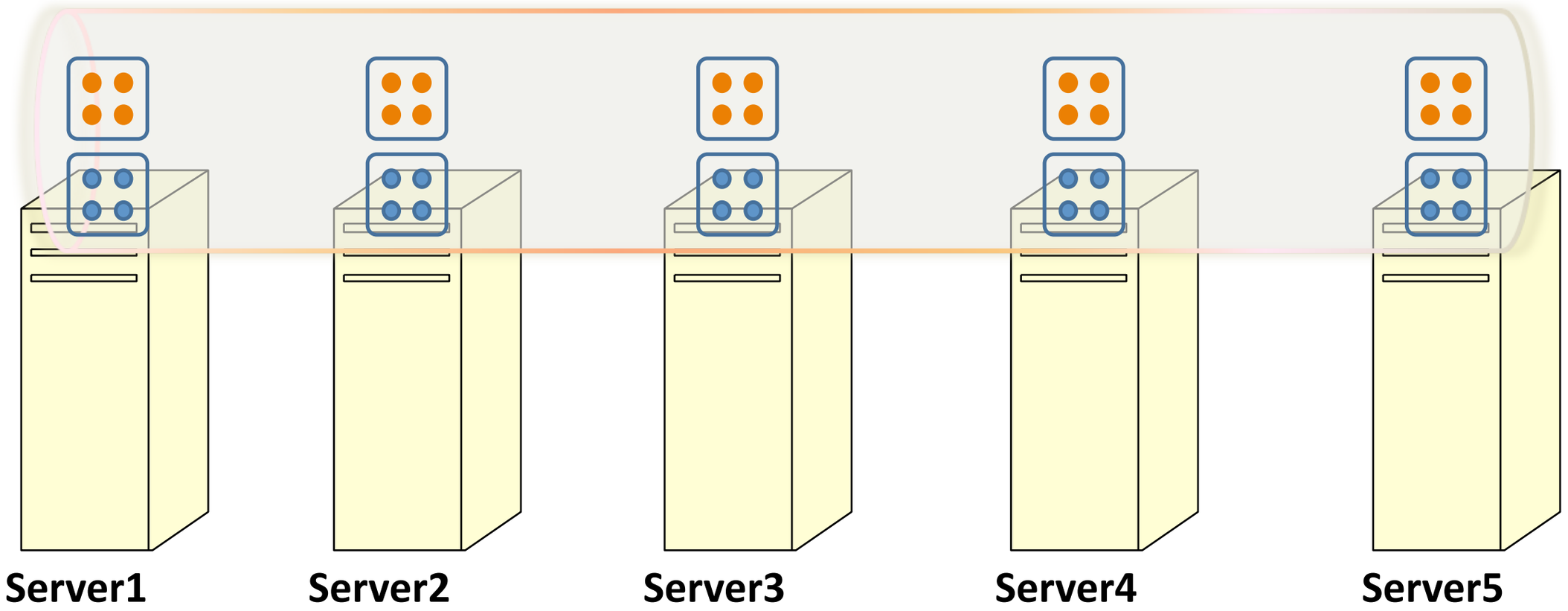
- License / Cost
- Feature-set
- Ease of use
- Main focus (distributed map, tuple space, cache, processing vs. data)
- Light/Heavy weight

Introducing Hazelcast

- Open source (Apache License)
- Super light, simple, no-dependency
- Distributed/partitioned implementation of map, queue, set, list, lock and executor service
- Transactional (JCA support)
- Secure
- Topic for pub/sub messaging
- Cluster info and membership events
- Dynamic clustering, backup, fail-over

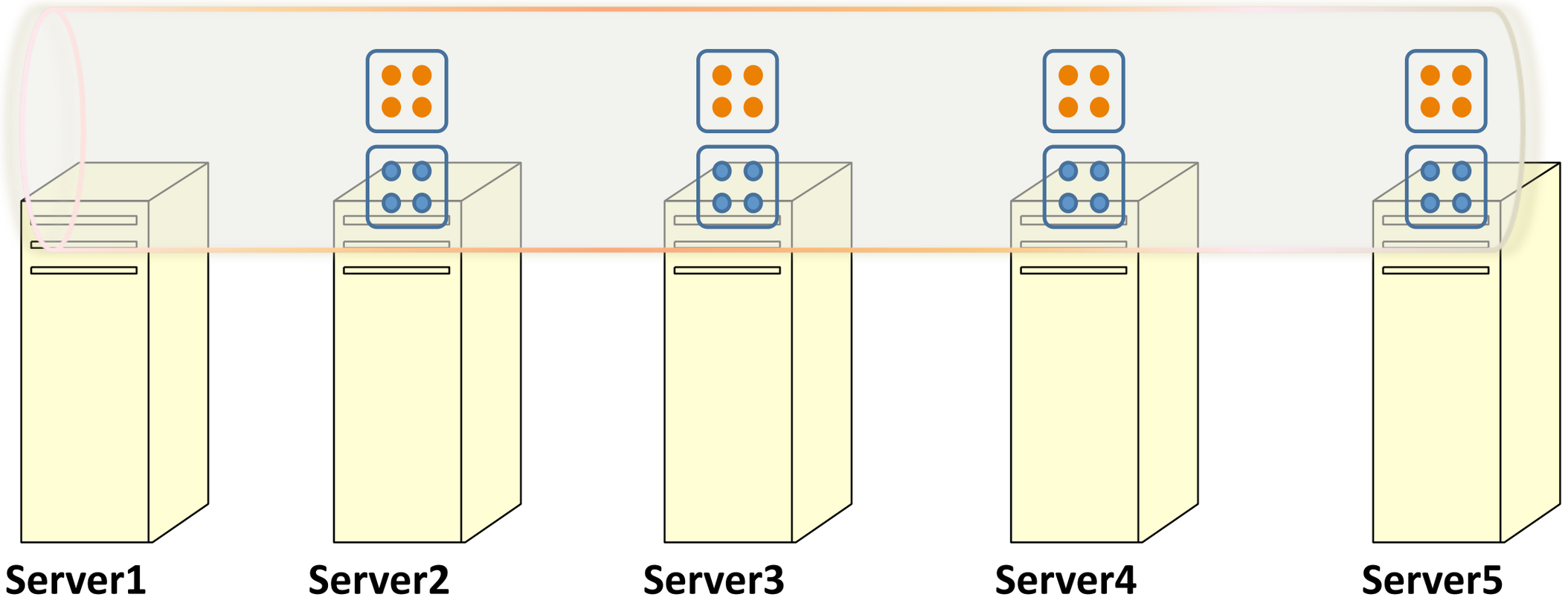
Data Partitioning in a Cluster

If you have 5 million objects in your 5-node cluster,
then each node will carry
1 million objects and 1 million backup objects.

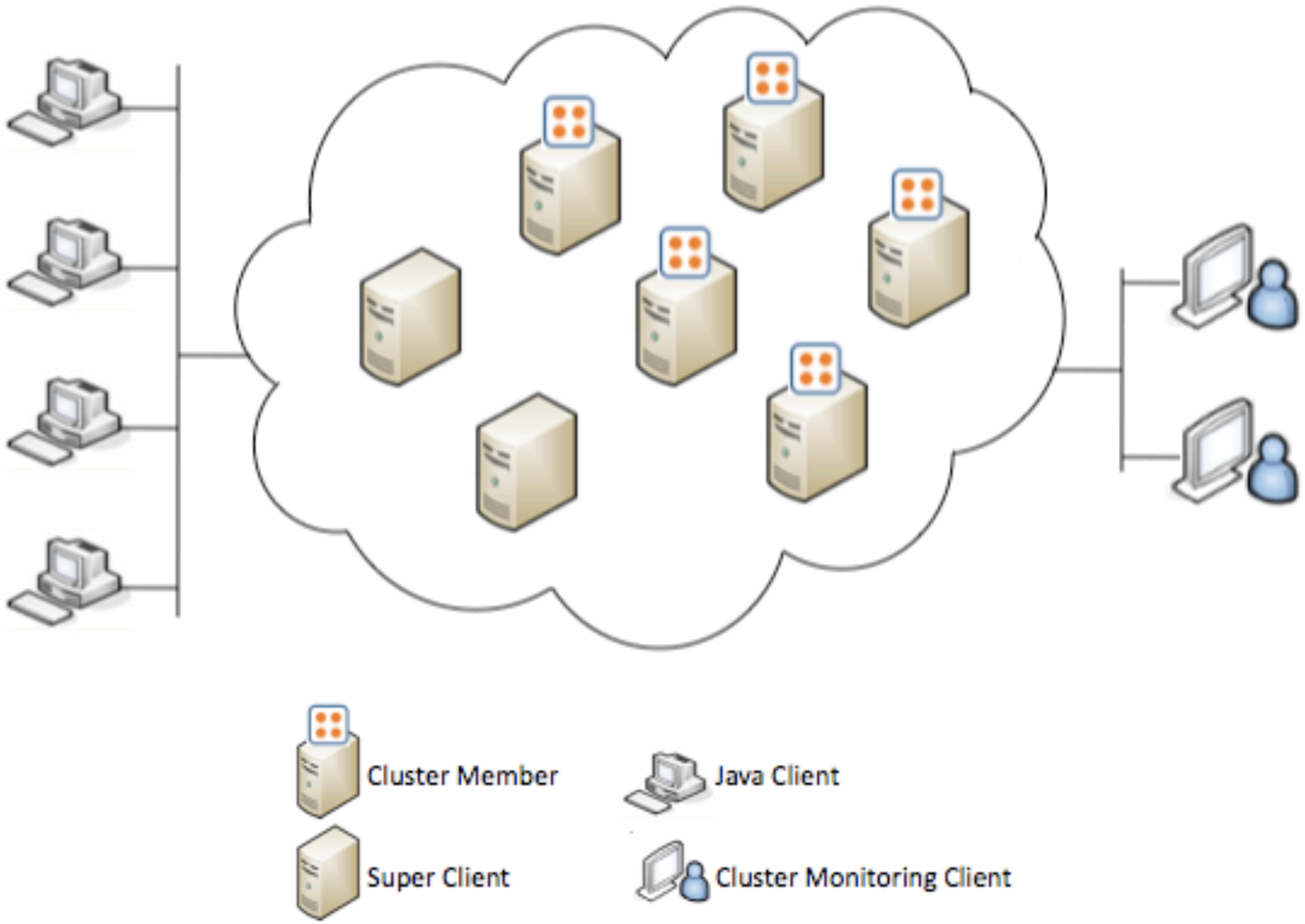


SuperClient in a Cluster

- `-Dhazelcast.super.client=true`
- As fast as any member in the cluster
- Holds no-data



Hazelcast Network



Code Samples – Cluster Interface

```
import com.hazelcast.core.*;
import java.util.Set;

Cluster cluster = Hazelcast.getCluster();
cluster.addMembershipListener(listener);

Member localMember  = cluster.getLocalMember();
System.out.println (localMember.getInetAddress());

Set setMembers  = cluster.getMembers();
```

Code Samples – Distributed Map

```
import com.hazelcast.core.Hazelcast;
import java.util.Map;

Map<String, Customer> map = Hazelcast.getMap("customers");


map.put ("1", customer);

Customer c = map.get("1");
```

Code Samples – Distributed MultiMap

```
import com.hazelcast.core.Hazelcast;
İmport com.hazelcast.core.MultiMap;

MultiMap<String, Order> map = Hazelcast.getMultiMap(”orders");


map.put ("1”, new Order ("iPhone", 340));
map.put ("1”, new Order (”MacBook", 1200));
map.put ("1”, new Order (”iPod", 79));
map.put (“2”, new Order (”iMac", 1500));

Collection<Order> colOrders = map.get ("1”);
for (Order order : colOrders) {
    // process order
}


boolean removed = map.remove("1”, new Order(”iPod", 79));
```

Code Samples – Distributed Queue

```
import com.hazelcast.core.Hazelcast;
import java.util.concurrent.BlockingQueue;
import java.util.concurrent.TimeUnit;

BlockingQueue<Task> queue = Hazelcast.getQueue("tasks");
```

```
queue.offer(task);

Task t = queue.poll();
Task t = queue.poll(5, TimeUnit.SECONDS);
```

Code Samples – Distributed Set

```
import com.hazelcast.core.Hazelcast;
import java.util.Set;

Set<Price> set = Hazelcast.getSet("IBM-Quote-History");

set.add (new Price (10, time1));
set.add (new Price (11, time2));
set.add (new Price (13, time3));

for (Price price : set) {
    // process price
}
```

Code Samples – Distributed Lock

```
import com.hazelcast.core.Hazelcast;
import java.util.concurrent.locks.Lock;

Lock mylock = Hazelcast.getLock(mylockobject);
mylock.lock();
try {
    // do something
} finally {
    mylock.unlock();
}
```

Code Samples – Distributed Topic

```
import com.hazelcast.core.*;

public class Sample implements MessageListener {

    public static void main(String[] args) {
        Sample sample = new Sample();
        ITopic<String> topic = Hazelcast.getTopic ("default");
        topic.addMessageListener(sample);
        topic.publish ("my-message-object");
    }

    public void onMessage(Object msg) {
        System.out.println("Got msg :" + msg);
    }
}
```

Code Samples – Distributed Events

```
import com.hazelcast.core.IMap;
import com.hazelcast.core.Hazelcast;
import com.hazelcast.core.EntryListener;
import com.hazelcast.core.EntryEvent;

public class Sample implements EntryListener {
       public static void main(String[] args) {
              Sample sample = new Sample();
              IMap    map   = Hazelcast.getMap ("default");
              map.addEntryListener (sample, true);
              map.addEntryListener (sample, "key");
        }

       public void entryAdded(EntryEvent event) {
              System.out.println("Added " + event.getKey() + ":" + event.getValue());
       }

       public void entryRemoved(EntryEvent event) {
              System.out.println("Removed " + event.getKey() + ":" + event.getValue());
       }

       public void entryUpdated(EntryEvent event) {
              System.out.println("Updated " + event.getKey() + ":" + event.getValue());
       }
}
```

Code Samples – Transactions

```
import com.hazelcast.core.Hazelcast;
import com.hazelcast.core.Transaction;
import java.util.Map;
import java.util.Queue;

Map   map   = Hazelcast.getMap   ("default");
Transaction txn = Hazelcast.getTransaction();
txn.begin();
try {
    //process obj
    map.put (key, obj);
    txn.commit();
} catch (Exception e) {
    txn.rollback();
}
```

Code Samples – Persistence

```
import com.hazelcast.core.MapStore,
import com.hazelcast.core.MapLoader,

public class MyMapStore implements MapStore, MapLoader {

  public Object load (Object key) {
    return readFromDatabase(key);
  }

  public void store (Object key, Object value) {
     saveIntoDatabase(key, value);
  }

  public void remove(Object key) {
     removeFromDatabase(key);
  }

}
```

Persistence

- Write-Behind : asynchronously storing entries
- Write-Through : synchronous
- Read-Through : if get(key) is null, load it

Code Samples – Executor Service

```
FutureTask<String> futureTask = new
        DistributedTask<String>(new Echo(input), member);

ExecutorService es = Hazelcast.getExecutorService();

es.execute(futureTask);

String result = futureTask.get();
```

Executor Service Scenario

```
public int addBonus(long customerId, int extraBonus){

        IMap<Long, Customer> mapCustomers =
                              Hazelcast.getMap("customers");
        mapCustomers.lock (customerId);
        Customer customer = mapCustomers.get(customerId);
        int currentBonus = customer.addBonus(extraBonus);
        mapCustomers.put(customerId, customer);
        mapCustomers.unlock(customerId);
        return currentBonus;

}
```

Send computation over data

```
public class BonusAddTask implements Callable<Integer>, Serializable{


        private static final long serialVersionUID = 1L;
        private long customerId;
        private long extraBonus;

        public BonusAddTask () {
        }
        public BonusAddTask (long customerId, int extraBonus) {
              this.customerId = customerId;
              this.extraBonus = extraBonus;
        }
        public Integer call () {
              IMap<Long, Customer> mapCustomers = Hazelcast.getMap("customers");
              mapCustomers.lock (customerId);
              Customer customer = mapCustomers.get(customerId);
              int currentBonus = customer.addBonus(extraBonus);
              mapCustomers.put(customerId, customer);
              mapCustomers.unlock(customerId);
              return currentBonus;
        }
}
```

Send computation over data

```
public int addBonus(long customerId, int extraBonus){
        ExecutorService es = Hazelcast.getExecutorService();
        FutureTask<Integer> task =
                        new DistributedTask<Integer>(new
                        BonusAddTask(customerId, extraBonus),
                        customerId);
        es.execute(task);
        int currentBonus = task.get();
        return currentBonus;
}
```

Code Samples – Query

```
public class Employee {
    private boolean active;
    private String name;
    private int age;

   // getters

   // setters

}
```

Code Samples – Query

```
import com.hazelcast.core.Hazelcast;
import com.hazelcast.core.IMap;
import com.hazelcast.query.SqlPredicate;
import java.util.Collection;

IMap map = Hazelcast.getMap("employees");

map.addIndex("active" ,false);
map.addIndex("name" ,false);
map.addIndex("age" ,true);


Collection<Employee> employees =
   map.values(new SqlPredicate("active AND age <= 30"));
```

```
<hazelcast>
        <group>
                <name>dev</name>
                <password>dev-pass</password>
        </group>
        <network>
                <port auto-increment="true">5701</port>
                <join>
                        <multicast enabled="true">
                                <multicast-group>224.2.2.3</multicast-group>
                                <multicast-port>54327</multicast-port>
                        </multicast>
                        <tcp-ip enabled="false">
                                <interface>192.168.1.2-5</interface>
                                <hostname>istanbul.acme</hostname>
                        </tcp-ip>
                </join>
                <interfaces enabled="false">
                        <interface>10.3.17.*</interface>
                </interfaces>
</network>
 <queue name="default">
        <max-size-per-jvm>10000</max-size-per-jvm>
         <time-to-live-seconds>60</time-to-live-seconds>
</queue>
 <map name="default">
        <backup-count>1</backup-count>
        <time-to-live-seconds>60</time-to-live-seconds>
        <max-size>10000</max-size>
        <eviction-policy>LRU</eviction-policy>
        <eviction-percentage>25</eviction-percentage>
</map>
```

Questions?

- http://www.hazelcast.com
- http://code.google.com/p/hazelcast/
- hazelcast@googlegroups.com
- Twitter @oztalip
- http://www.linkedin.com/in/talipozturk