introduction to

emin demirci

agenda

- introduction
- distributed computing
- in-memory data grids
- hazelcast
- code samples
- demo
- internals
- q/a

about me

- core developer at hazelcast
- holds bsc. computer engineering
- started programming some time ago, then turned into a career
- lives in beautiful istanbul
- interested in distributed systems

distributed computing

- use of bunch of computers to solve a computational problem
- problem is divided into multiple tasks and they are solved by one or more computers
- computers communicate each other by sending messages

in-memory data grids

- middleware software
- shared nothing architecture
- manages objects across distributed servers in the RAM
- ability to scale
- provides fault tolerance

why use an imdg?

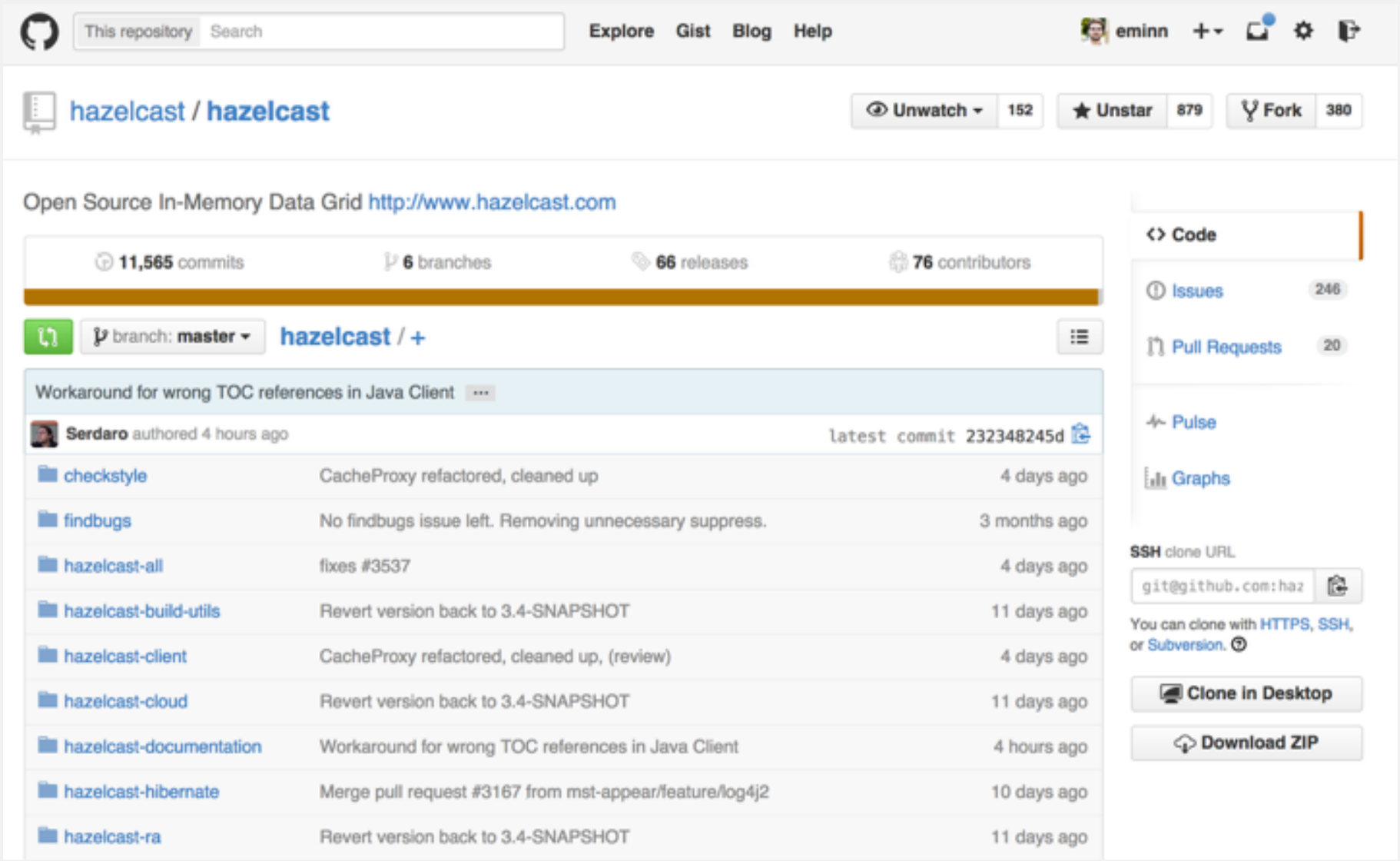
- performance - ram is faster
- flexibility - rich set of data structures
- operations - easy to scale/maintain

other imdg solutions

- oracle coherence
- ibm extremescale
- vmware gemfire
- gigaspaces
- redhat infinispan
- gridgain
- terracotta

what is hazelcast ?

an open-source project



hazelcast

a company

a company

- hazelcast enterprise edition
- management center
- enterprise support
- training / consulting
- offices in istanbul (r&d), palo alto(hq) and london

an open-source project

- leading open-source in-memory data grid.
- dead simple distributed programming
- easy way to scale applications
- simple api
- built with ❤ in Istanbul

use cases

- scaling your application
- sharing data across cluster
- partitioning data
- sending/receiving messages
- load balancing
- session replication
- parallel task processing on multiple machines
- ...

how hazelcast differs ?

- apache licensed open source
- lightweight w/o any dependency
- ease of use and more fun !

who uses hazelcast ?

a lot of developers :)

fact : every ~0.4 second a hazelcast node is started around the world

who uses hazelcast ?

what is  hazelcast?

- distributed impl. of Java Collections
- dynamic clustering, backup and failover
- transaction support (two phase, XA)
- distributed execution framework
- map/reduce api
- distributed queries
- native Java, C#, C++ clients

starting a hazelcast instance

```java
public class StartHazelcastInstance {
    public static void main(String[] args) {
        HazelcastInstance hazelcastInstance = Hazelcast.newHazelcastInstance();
        System.out.println("Cluster Members = " + hazelcastInstance.getCluster().getMembers());
    }
}
```

hazelcast instance api

```java
public interface HazelcastInstance {

    String getName();

    <E> IQueue<E> getQueue(String name);

    <E> ITopic<E> getTopic(String name);

    <E> ISet<E> getSet(String name);

    <E> IList<E> getList(String name);

    <K, V> IMap<K, V> getMap(String name);

    <K, V> ReplicatedMap<K, V> getReplicatedMap(String name);

    JobTracker getJobTracker(String name);

    <K, V> MultiMap<K, V> getMultiMap(String name);

    ILock getLock(String key);

    Cluster getCluster();

};
```

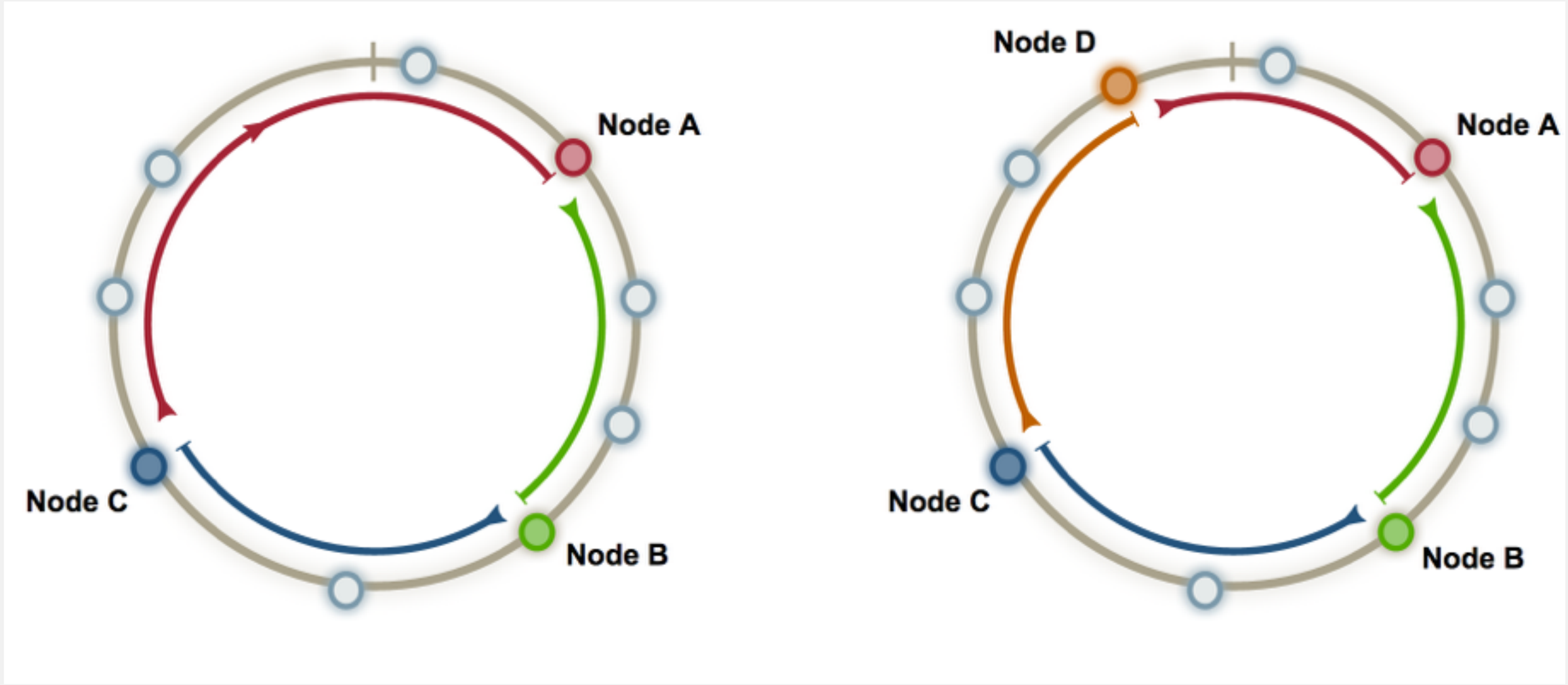
quick demo

```
Nov 16, 2014 4:37:11 PM com.hazelcast.config.FileSystemXmlConfig
INFO: Configuring Hazelcast from '/Users/emindemirci/Development/hazelcast/hazelcast.xml'.
Nov 16, 2014 4:37:11 PM com.hazelcast.instance.DefaultAddressPicker
INFO: [LOCAL] [dev] [3.3.2] Prefer IPv4 stack is true.
Nov 16, 2014 4:37:12 PM com.hazelcast.instance.DefaultAddressPicker
INFO: [LOCAL] [dev] [3.3.2] Picked Address[192.168.2.7]:5701, using socket ServerSocket[addr=/0:0:0:0:0:0:0:0%0,localport=5701], bind any local is true
Nov 16, 2014 4:37:12 PM com.hazelcast.spi.impl.BasicOperationScheduler
INFO: [192.168.2.7]:5701 [dev] [3.3.2] Starting with 8 generic operation threads and 8 partition operation threads.
Nov 16, 2014 4:37:12 PM com.hazelcast.system
INFO: [192.168.2.7]:5701 [dev] [3.3.2] Hazelcast 3.3.2 (20141112) starting at Address[192.168.2.7]:5701
Nov 16, 2014 4:37:12 PM com.hazelcast.system
INFO: [192.168.2.7]:5701 [dev] [3.3.2] Copyright (C) 2008-2014 Hazelcast.com
Nov 16, 2014 4:37:12 PM com.hazelcast.instance.Node
INFO: [192.168.2.7]:5701 [dev] [3.3.2] Creating MulticastJoiner
Nov 16, 2014 4:37:12 PM com.hazelcast.core.LifecycleService
INFO: [192.168.2.7]:5701 [dev] [3.3.2] Address[192.168.2.7]:5701 is STARTING
Nov 16, 2014 4:37:15 PM com.hazelcast.cluster.MulticastJoiner
INFO: [192.168.2.7]:5701 [dev] [3.3.2]


Members [1] {
    Member [192.168.2.7]:5701 this
}

Nov 16, 2014 4:37:15 PM com.hazelcast.core.LifecycleService
INFO: [192.168.2.7]:5701 [dev] [3.3.2] Address[192.168.2.7]:5701 is STARTED
Nov 16, 2014 4:37:15 PM com.hazelcast.partition.InternalPartitionService
INFO: [192.168.2.7]:5701 [dev] [3.3.2] Initializing cluster partition table first arrangement...
```

data partitioning



Drawing by Benjamin Erb http://berb.github.io/diploma-thesis/original/resources/cons_hash.svg

distributed map

```java
public class DistributedMap {
    public static void main(String[] args) {
        HazelcastInstance h = Hazelcast.newHazelcastInstance();
        ConcurrentMap<String, String> map = h.getMap("my-distributed-map");
        map.put("key", "value");
        String value = map.get("key");

        System.out.println("value = " + value);  // will print "value"

        //Concurrent Map methods
        map.putIfAbsent("somekey", "somevalue");
        map.replace("key", "value", "newvalue");
    }
}
```

distributed queries

```java
public class SqlQueryMember {

    public static void main(String[] args) {
        HazelcastInstance hz = Hazelcast.newHazelcastInstance();
        IMap<String, Customer> map = hz.getMap("map");

        map.put("1", new Customer("peter", true, 36));
        map.put("2", new Customer("john", false, 40));
        map.put("3", new Customer("roger", true, 20));

        Set<Customer> employees = (Set<Customer>) map.values(new SqlPredicate("active AND age < 30"));
        System.out.println("Employees:" + employees);
    }
```

processing entries

```
public static void main(String[] args) {
    HazelcastInstance instance = Hazelcast.newHazelcastInstance();
    IMap<String, Employee> map = instance.getMap("employees");
    map.lock("john");
    Employee john = map.get("john");
    john.incSalary(10);
    map.put("john", john);
    map.unlock("john");
}
```

entry processors

```java
public class EntryProcessorMember {
    public static void main(String[] args) {
        HazelcastInstance hz = Hazelcast.newHazelcastInstance();
        IMap<String, Employee> employees = hz.getMap("employees");
        employees.put("John", new Employee(1000));
        employees.put("Mark", new Employee(1000));
        employees.put("Spencer", new Employee(1000));

        employees.executeOnEntries(new EmployeeRaiseEntryProcessor());

        for (Map.Entry<String, Employee> entry : employees.entrySet()) {
            System.out.println(entry.getKey() + " salary: " + entry.getValue().getSalary());
        }
        System.exit(0);
    }

    static class EmployeeRaiseEntryProcessor extends AbstractEntryProcessor<String, Employee> {
        @Override
        public Object process(Map.Entry<String, Employee> entry) {
            Employee value = entry.getValue();
            value.incSalary(10);
            entry.setValue(value);
            return null;
        }
    }
```

distributed queue

```java
public class DistributedQueue {
    public static void main(String[] args) throws InterruptedException {
        HazelcastInstance h = Hazelcast.newHazelcastInstance();
        BlockingQueue<String> queue = h.getQueue("my-distributed-queue");
        queue.offer("item");
        String item = queue.poll();

        //Timed blocking Operations
        queue.offer("anotheritem", 500, TimeUnit.MILLISECONDS);
        String anotherItem = queue.poll(5, TimeUnit.SECONDS);

        //Indefinitely blocking Operations
        queue.put("yetanotheritem");
        String yetanother = queue.take();
    }
}
```

distributed executor service

```java
public class DistributedExecutorService {
    public static void main(String[] args) {
        Config config = new Config();
        HazelcastInstance h = Hazelcast.newHazelcastInstance(config);
        IExecutorService ex = h.getExecutorService("my-distributed-executor");
        ex.submit(new MessagePrinter("message to any node"));
        Member firstMember = h.getCluster().getMembers().iterator().next();
        ex.executeOnMember(new MessagePrinter("message to very first member of the cluster"), firstMember);
        ex.executeOnAllMembers(new MessagePrinter("message to all members in the cluster"));
        ex.executeOnKeyOwner(new MessagePrinter("message to the member that owns the following key"), "key");
    }

    static class MessagePrinter implements Runnable, Serializable {
        final String message;

        MessagePrinter(String message) {
            this.message = message;
        }

        @Override
        public void run() {
            System.out.println(message);
        }
    }
}
```

distributed lock

```java
public class DistributedLock {
    public static void main(String[] args) {
        HazelcastInstance h = Hazelcast.newHazelcastInstance();
        Lock lock = h.getLock("my-distributed-lock");
        lock.lock();
        try {
            //do something here
        } finally {
            lock.unlock();
        }
    }
}
```

aggregations

```java
public class DistributedAggregation {
    public static void main(String[] args) {
        Config config = new Config();
        HazelcastInstance h = Hazelcast.newHazelcastInstance(config);

        IMap<String, Integer> salaries = h.getMap("salaries");

        Aggregation<String, Integer, Integer> integerSum = Aggregations.integerSum();
        Supplier<String, Integer, Integer> all = Supplier.all();

        Integer sum = salaries.aggregate(all, integerSum);
        System.out.println("Aggregated sum: " + sum);
    }
}
```

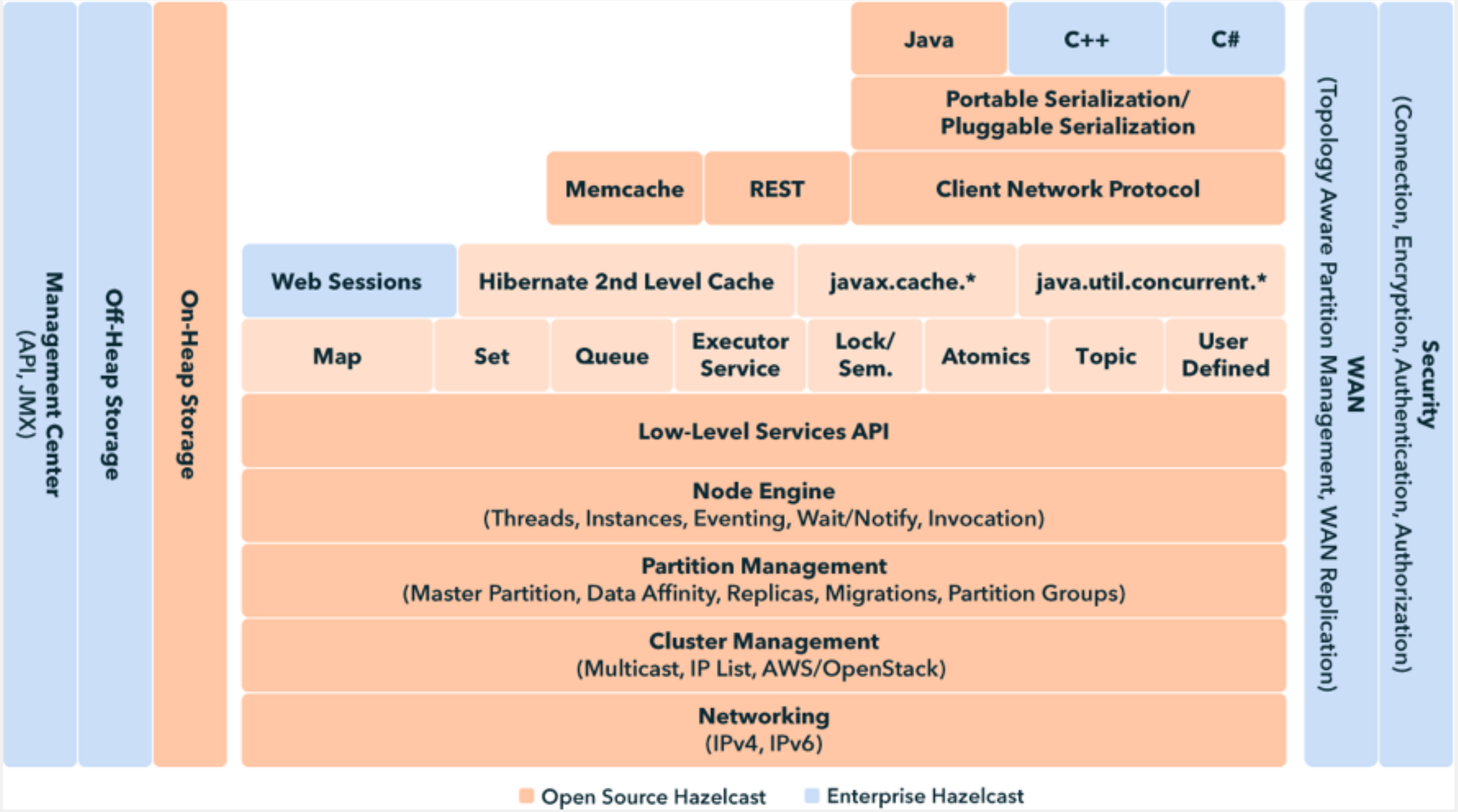
wan replication

- active/active
- active/passive

wan replication

```
<hazelcast>
<wan-replication name="my-wan-cluster">
    <target-cluster group-name="tokyo" group-password="tokyo-pass">
        <replication-impl>com.hazelcast.wan.impl.WanNoDelayReplication</replication-impl>
        <end-points>
            <address>10.2.1.1:5701</address>
            <address>10.2.1.2:5701</address>
        </end-points>
    </target-cluster>
    <target-cluster group-name="london" group-password="london-pass">
        <replication-impl>com.hazelcast.wan.impl.WanNoDelayReplication</replication-impl>
        <end-points>
            <address>10.3.5.1:5701</address>
            <address>10.3.5.2:5701</address>
        </end-points>
    </target-cluster>
</wan-replication>
...
</hazelcast>
```

hazelcast architecture



service provider interface (SPI)

- roll your own services
- extend hazelcast based on your needs !
- hierarchical lock service
- priority queue
- scheduled executor service
- distributed actors
- anything you can think of !
- check out SPI section of the hazelcast documentation

session replication

- servlet filter based
- just put hazelcast filter to your web.xml
- native tomcat/jetty plugins (enterprise)

amazon ec2 support

```
<hazelcast>
...
<join>
    <aws enabled="true">
        <access-key>my-access-key</access-key>
        <secret-key>my-secret-key</secret-key>
        <region>us-west-1</region>
    </aws>
</join>
...
</hazelcast>
```

near cache

- client side
- node side

map/reduce

```java
public class BasicMapReduce {

    public static void main(String[] args) throws ExecutionException, InterruptedException {

        final HazelcastInstance hz1 = Hazelcast.newHazelcastInstance();
        final HazelcastInstance hz2 = Hazelcast.newHazelcastInstance();
        final HazelcastInstance hz3 = Hazelcast.newHazelcastInstance();

        //Create a default map.
        IMap<Integer, Integer> m1 = hz1.getMap("default");
        for (int i = 0; i < 10000; i++) {
            m1.put(i, i);
        }

        //Create a job tracker with default config.
        JobTracker tracker = hz1.getJobTracker("myJobTracker");

        //Using a built-in source from our IMap. This supplies key value pairs.
        KeyValueSource<Integer, Integer> kvs = KeyValueSource.fromMap(m1);

        //Create a new Job with our source.
        Job<Integer, Integer> job = tracker.newJob(kvs);

        //Configure the job.
        ICompletableFuture<Map<String, Integer>> myMapReduceFuture =
                job.mapper(new MyMapper()).reducer(new MyReducerFactory())
                        .submit();

        Map<String, Integer> result = myMapReduceFuture.get();

        System.out.println("The sum of the numbers 1 to 10,000 is: " + result.get("all_values"));
    }
}
```

map/reduce

```
public static class MyMapper implements Mapper<Integer, Integer, String, Integer> {

    @Override
    public void map(Integer key, Integer value, Context<String, Integer> context) {
        context.emit("all_values", value);
    }
}
public static class MyReducerFactory implements ReducerFactory<String, Integer, Integer> {
    @Override
    public Reducer<Integer, Integer> newReducer(String key) {
        return new MyReducer();
    }
}

public static class MyReducer extends Reducer<Integer, Integer> {

    private AtomicInteger sum = new AtomicInteger(0);

    @Override
    public void reduce(Integer value) {
        sum.addAndGet(value);
    }

    @Override
    public Integer finalizeReduce() {
        return sum.get();
    }
}
```

hazelcast stabilizer

```
#!/bin/sh

boxCount=25
members=25
workers=100
duration=48h


provisioner --scale $boxCount

coordinator --memberWorkerCount $members \
        --clientWorkerCount $workers \
        --duration $duration \
        --workerVmOptions "-XX:+HeapDumpOnOutOfMemoryError" \
        --parallel \
        ../../test.properties

provisioner --download

provisioner --terminate
```

hazelcast stabilizer

```
1
2  MapCasTest@class=com.hazelcast.stabilizer.tests.map.MapCasTest
3  MapCasTest@threadCount=3
4  MapCasTest@keyCount=1000
5  MapCasTest@basename=MapCasTest
6
7  MapLockTest@class=com.hazelcast.stabilizer.tests.map.MapLockTest
8  MapLockTest@threadCount=3
9  MapLockTest@keyCount=1000
10 MapLockTest@basename=MapLockTest
11
12 MapTransactionTest@class=com.hazelcast.stabilizer.tests.map.MapTransactionTest
13 MapTransactionTest@threadCount=3
14 MapTransactionTest@keyCount=1000
15 MapTransactionTest@reThrowTransactionException=false
16 MapTransactionTest@basename=MapTransactionTest
```

hazelcast stabilizer

```
CLOUD_PROVIDER=aws-ec2
CLOUD_IDENTITY=~/ec2.identity
CLOUD_CREDENTIAL=~/ec2.credential
MACHINE_SPEC=hardwareId=c3.2xlarge,locationId=us-east-1,imageId=us-east-1/ami-1b3b2472
JDK_FLAVOR=oracle
JDK_VERSION=7
PROFILER=none
HAZELCAST_VERSION_SPEC=maven=3.2.5
SECURITY_GROUP=stabilizer-final-xlarge
```

Thank you ! :)

any questions ?


emin@hazelcast.com


we are hiring, check out
hazelcast.com/careers