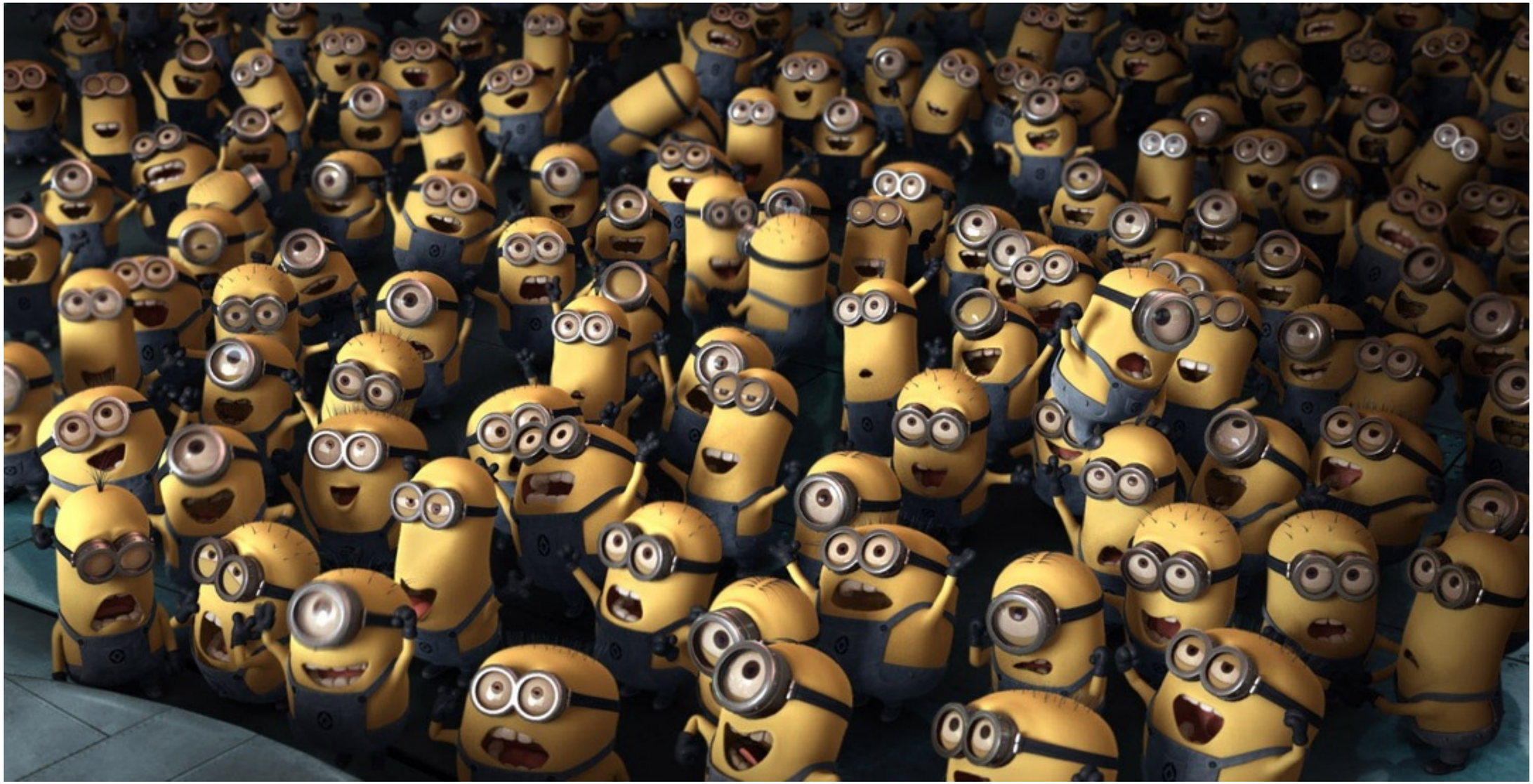
Distributed Systems Using Hazelcast

Peter Veentjer

HAZELCAST

Whoami

- Working for Hazelcast
  - Senior developer
  - Solution architect
- Author of ‘Mastering Hazelcast 3'
- 14 years Java experience
- Big love
  - Concurrency control
  - Distributed computing



What is Hazelcast?

What is Hazelcast?

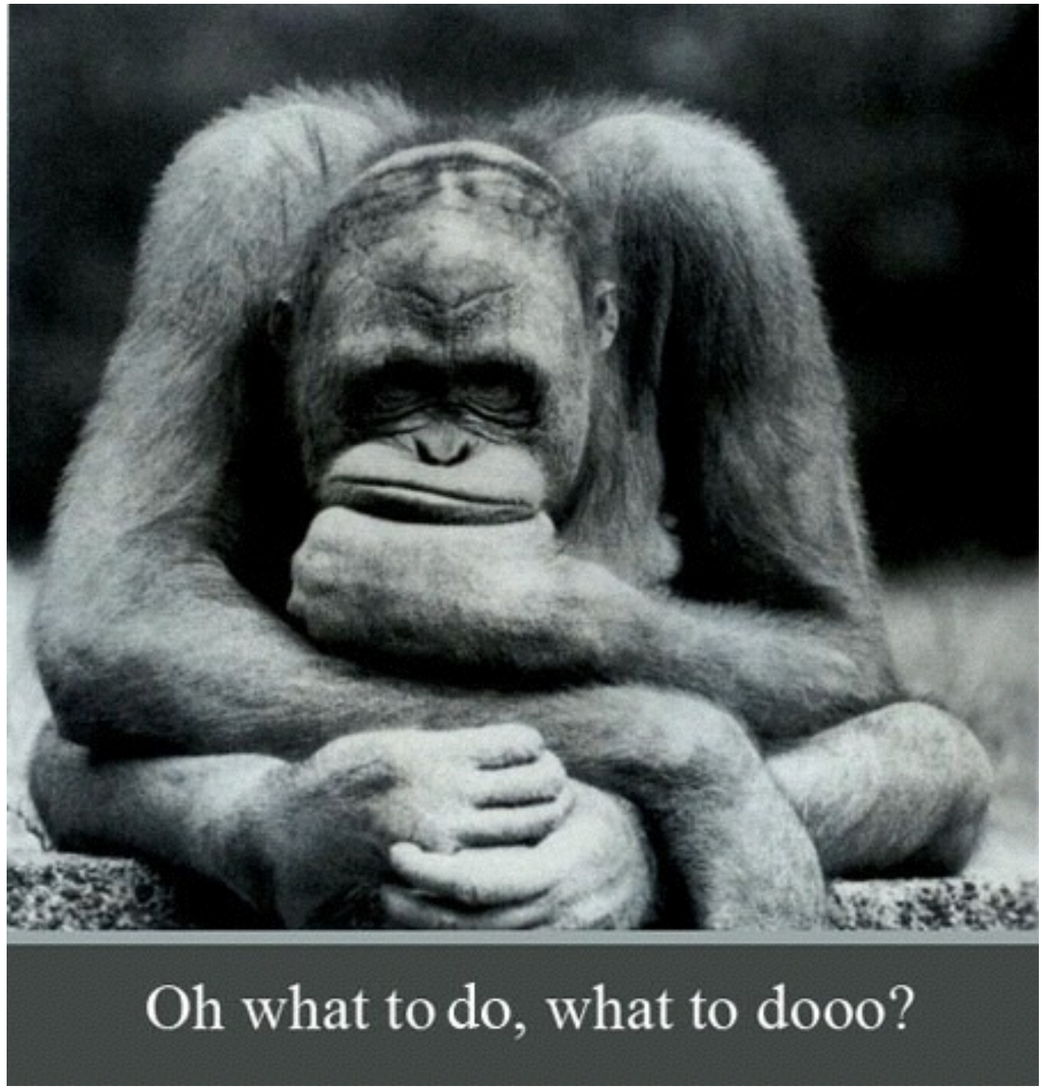
- Leading Open Source Java In Memory Data/Compute Grid
- Out main goal is to simplify development of:
  - Scalable systems
  - Highly available systems

Why Hazelcast?

- 2.5 MByte JAR
  - no other dependencies!
  - no need to install software!
- Its a library, not an application framework
- Apache 2 License
- Free to use
  - no limitations

Distributed Data-structures

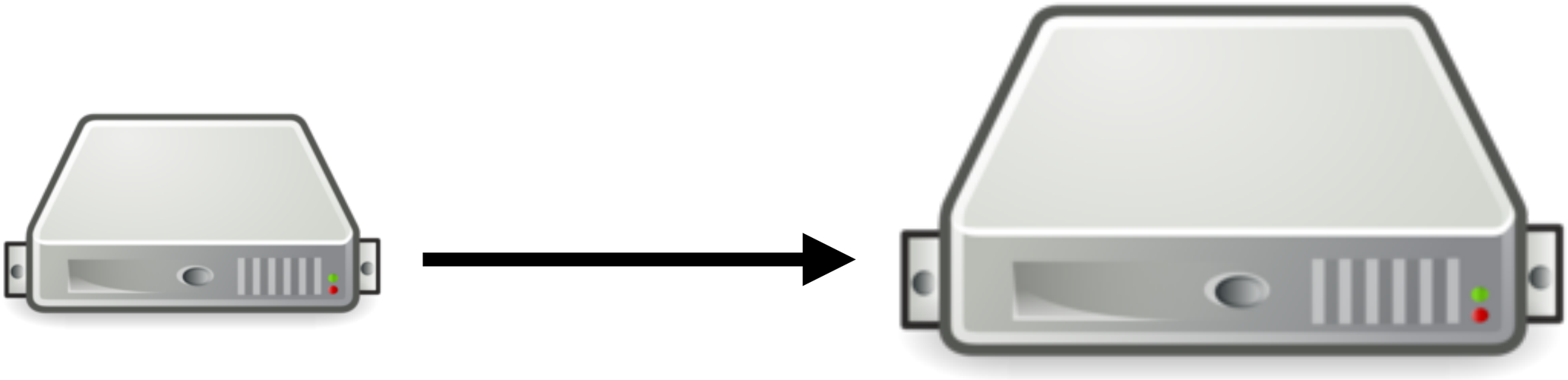
- IAtomicLong/Ref
- IdGenerator
- Lock/Condition
- CountDownLatch
- Semaphore
- Queue
- Map
- MultiMap
- Set
- List
- Topic
- Executor
- TransactionalMap/Q/S…
- Write your own!



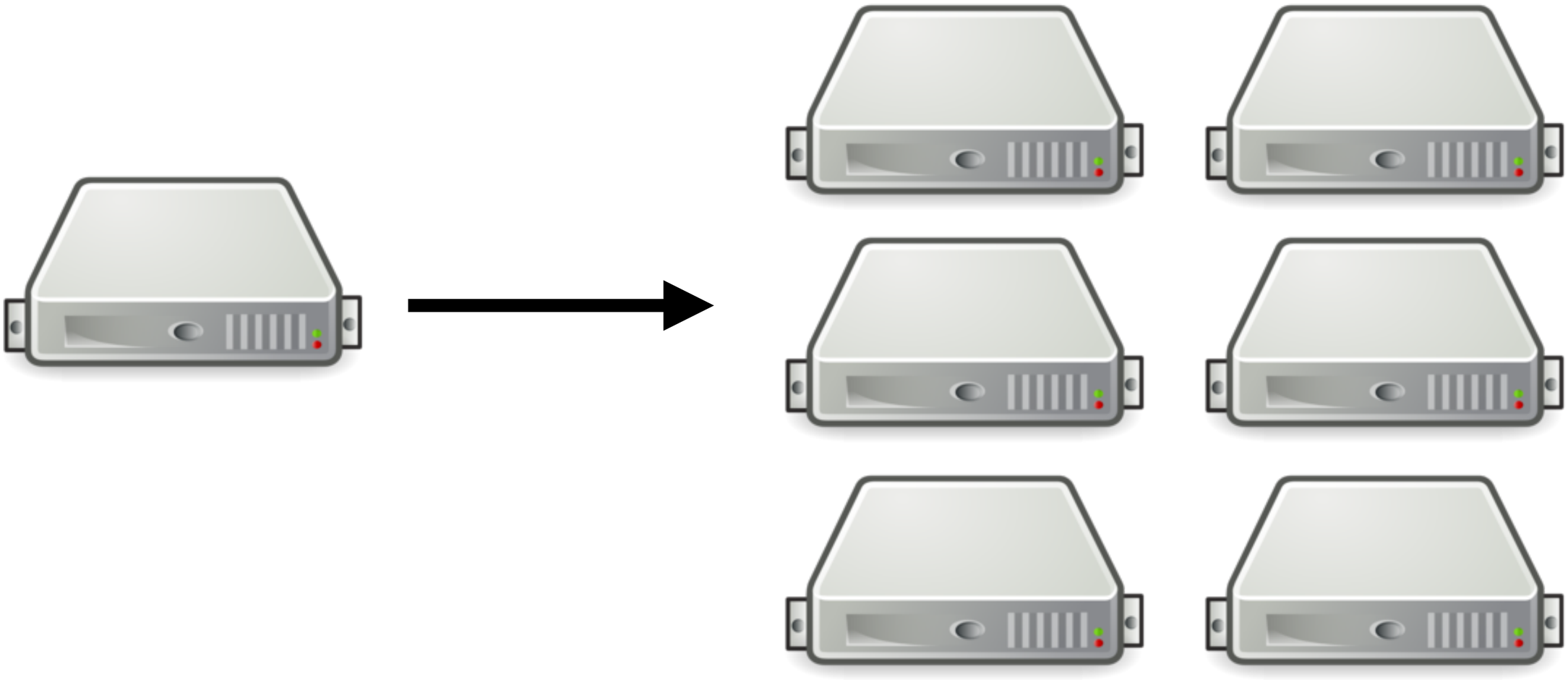
So what can I do with it?

Scaling

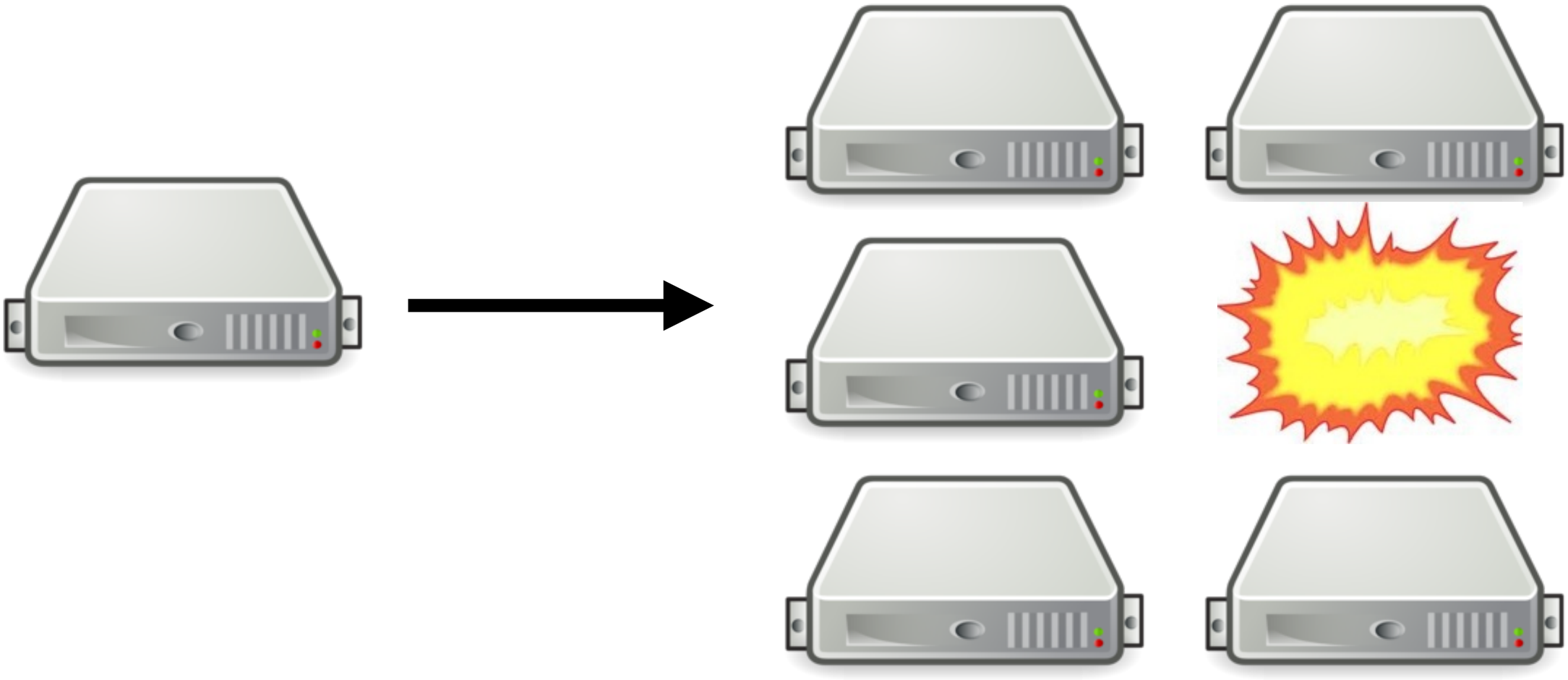
Scale up



Scale Out



High Availability



Raspberry Pi Cluster

When?

- Caching
- Messaging Solution
- Event processing
- Clustered Scheduling
- Job Processing
- Cluster management
- HTTP session clustering

Which companies

- E-Commerce
  - Apple, eBay
- Financials
  - JP Morgan, Morgan Stanley, HSBC, Deutsche Bank
- Telco's
  - AT&T, Ericsson
- Gaming
  - Ubisoft/Blue Byte

Which Open Source Projects

- WSO2 Carbon
- Mule ESB
- Vert.x
- Apache Camel
- Apache Shiro
- OrientDB
- Alfresco
- Karaf

How Often

- In October 2013
  - 3M startups
  - 48K unique

Where is the code!

Creating Hazelcast Cluster

```
HazelcastInstance hz = Hazelcast.newHazelcastInstance();
```

XML Configuration

```
<hazelcast>
   <network>…</network>
   <map name="m">…</map>
   <queue name="q">…</queue>
   <…
</hazelcast>
```

Programmatic Configuration

```
Config config = new Config();
… make config modifications
HazelcastInstance hz = Hazelcast.newHazelcastInstance(config);
```

Demo

Map

```
Map<String,String> products = new HashMap();
map.put("1","IPhone");
```

Map

```
Map<String,String> products = new ConcurrentHashMap();
map.put("1","IPhone");
```

Map

```
HazelcastInstance hz = Hazelcast.newHazelcastInstance();
Map<String,String> products = hz.getMap("products");
cities.put("1","IPhone");
```

Demo

Map Configuration

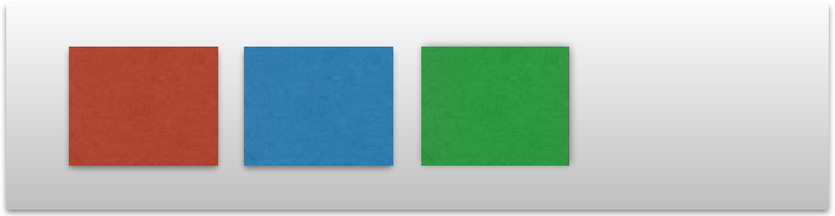
```
<map name="products">
  <time-to-live-seconds>4</time-to-live-seconds>
  <indexes>
     <index>name</index>
  </indexes>
  <..>
</map>
```

Map Scalability and High Availability

… 271 Partitions

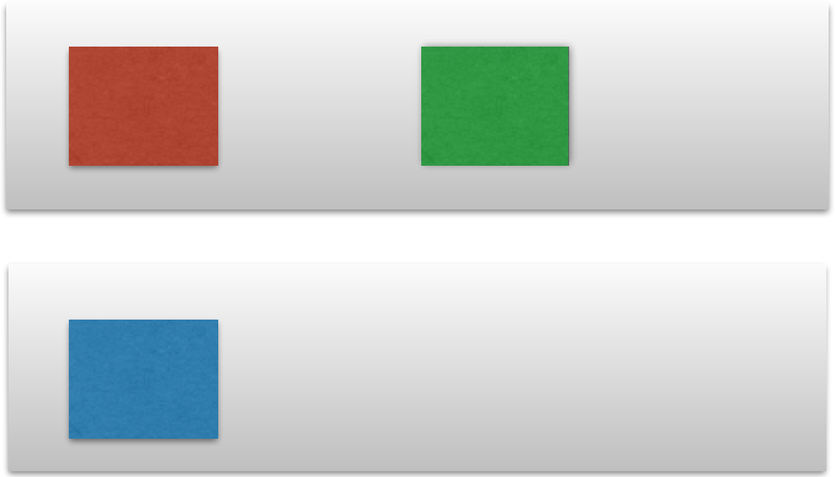
partitionid = hash(key) % partitioncount

Map Scalability and High Availability

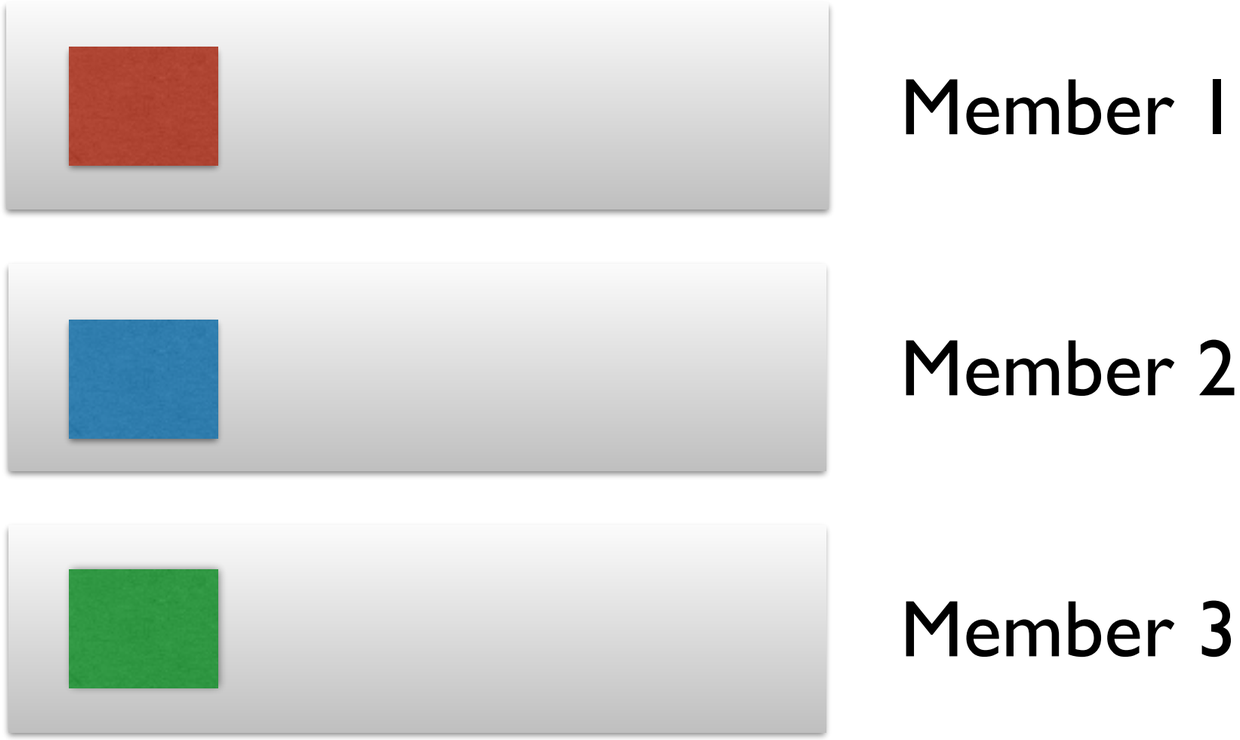


Member 1

Map Scalability and High Availability

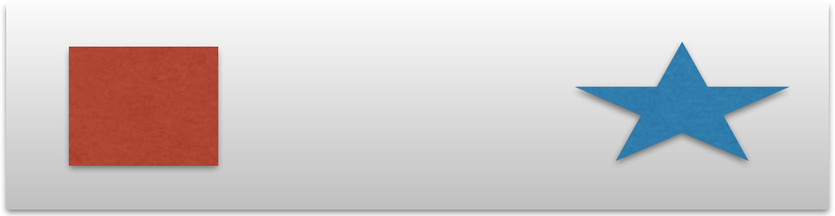


Map Scalability and High Availability

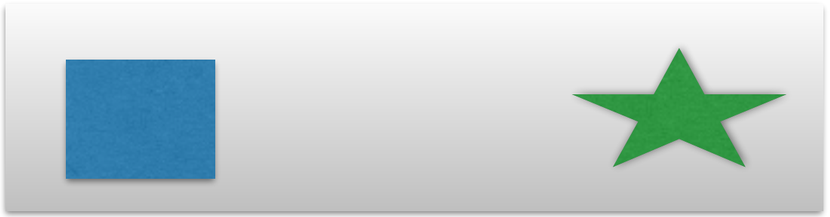


Map Scalability and High Availability

Map Scalability and High Availability



Member 1



Member 2

Member 3

Map Scalability and High Availability

Demo

Multimap

- Collection as value
  - Serialization Overhead
  - Lost update

Demo

Queue

```
BlockingQueue queue = new LinkedBlockingQueue();
queue.offer("1");
Object item = queue.take();
```

Queue

```
HazelcastInstance hz = Hazelcast.newHazelcastInstance();
BlockingQueue queue = hz.getQueue("queue");
queue.offer("1");
Object item = queue.take();
```

Demo

Topic

```
ITopic topic = hz.getTopic("topic");
//publishing
topic.publish(msg);

//subscribing
topic.addMessageListener(new TopicSubscriber());

public class TopicSubscriber implements MessageListener<String> {
    public void onMessage(Message<String> m) {
        System.out.println(m.getMessageObject());
    }
}
```

Client

- Simplified
- Encryption
- Load Balance Policy
- C++ and C# version

Client

```
HazelcastInstance client = HazelcastClient.newHazelcastClient();
```

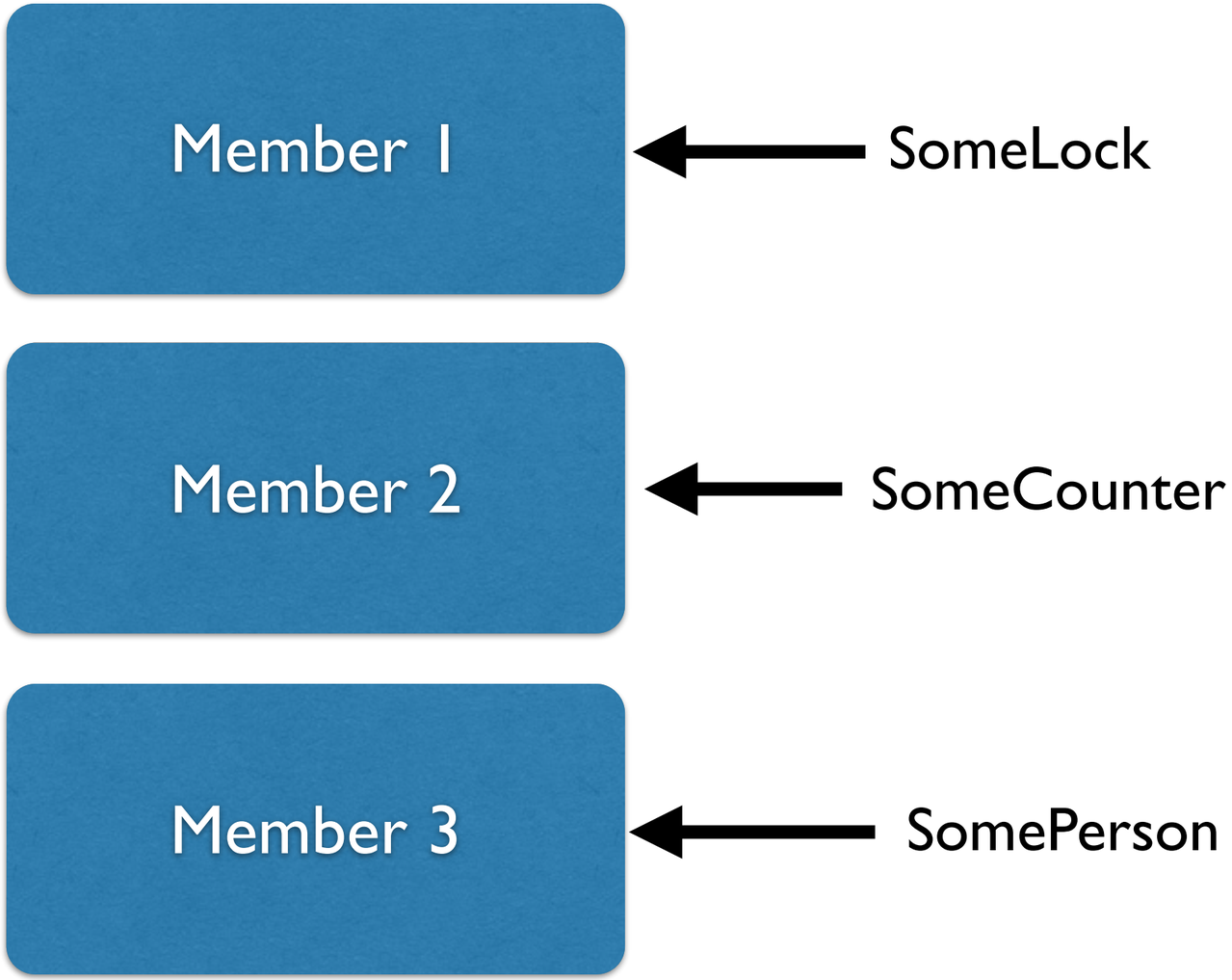
Network Communication

- Cluster Discovery
  - Multicast
  - TCP/IP Cluster
  - AWS Cluster
- Normal Network Communication
  - TCP/IP

Advanced Hazelcast

Data Locality: Problem

```
ILock lock = hz.getLock("someLock");
IAtomicLong counter = getAtomicLong("someCounter");
IMap map = hz.getMap("someMap");
map.put("somePerson",new Person());
map.get("somePerson");
```


Member 1
SomeLock
Member 2
SomeCounter
Member 3
SomePerson

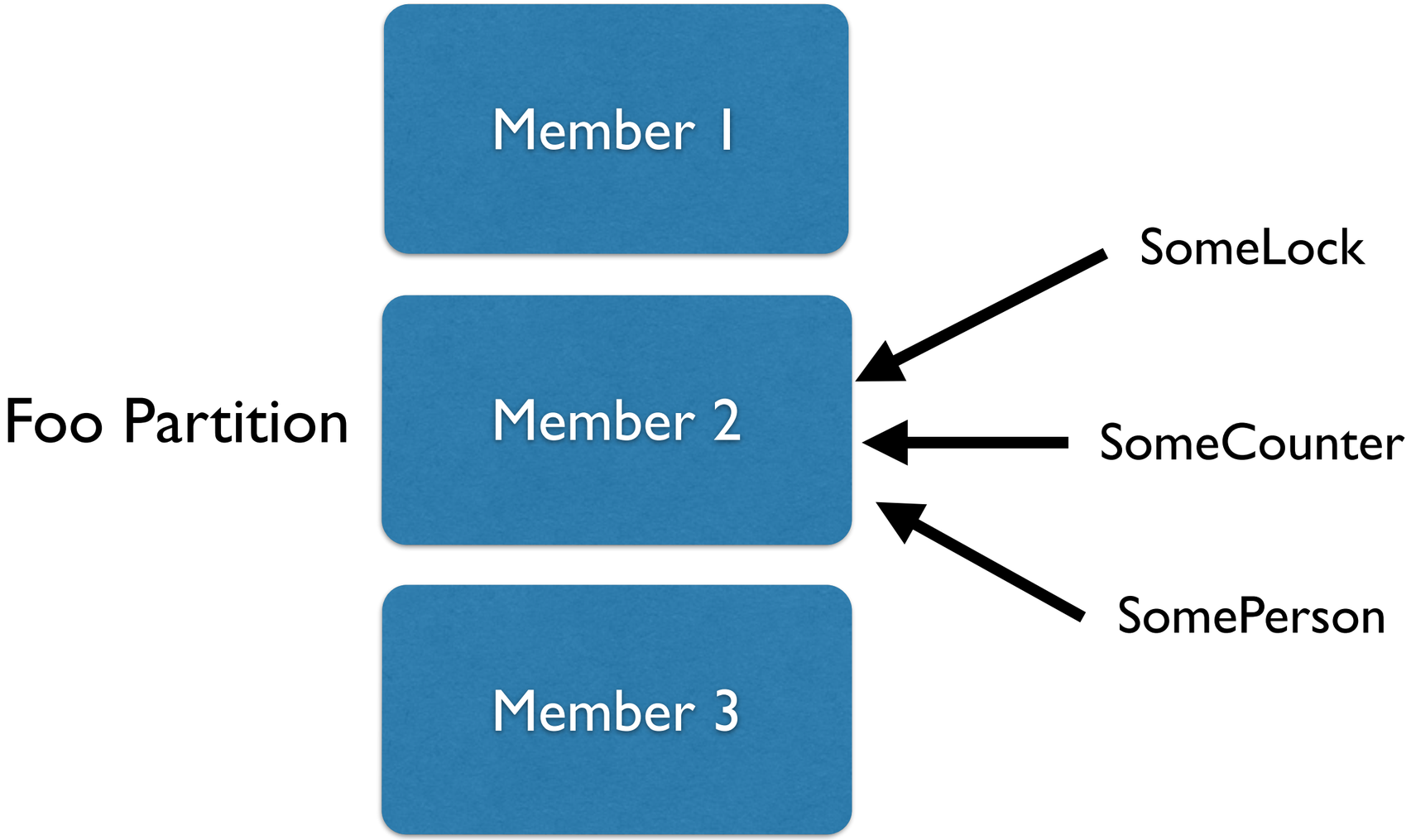
name

↓

name@partitionkey

Data Locality: Fixed

```
ILock lock = hz.getLock("someLock@foo");
IAtomicLong counter = hz.getAtomicLong("someCounter@foo");
IMap map = hz.getMap("someMap");
map.put("somePerson@foo", new Person());
Person p = map.get("somePerson@foo");
```


Member 1
Member 2
Member 3
Foo Partition
SomeLock
SomeCounter
SomePerson

Adding object in same partition

```
IAtomicLong c1 = …
IAtomicLong c2 = hz.getAtomicLong("c2@"+c1.getPartitionKey());
```

Executor

- Execute task anywhere
- Execute task on key owner
- Execute task
  - one/all/subset members
- Synchronisation
  - Future
  - ExecutionCallback
  - CompletableFuture Hazelcast 3.3

Demo

Locking

- Locks
- TransactionalMap.getForUpdate
- Map.Lock

Demo

Race problem

```
public void increment(int accountId,int amount){
    Account account = accounts.get(accountId);
    account.balance+=amount;
    accounts.put(accountId, account);
}
```

Pessimistic Increment

```
public void increment(int accountId,int amount){
    accounts.lock(accountId);
    try{
        Account account = accounts.get(accountId);
        account.balance+=amount;
        accounts.put(accountId, account);
    }finally{
        accounts.unlock(accountId);
    }
}
```

Pessimistic Increment

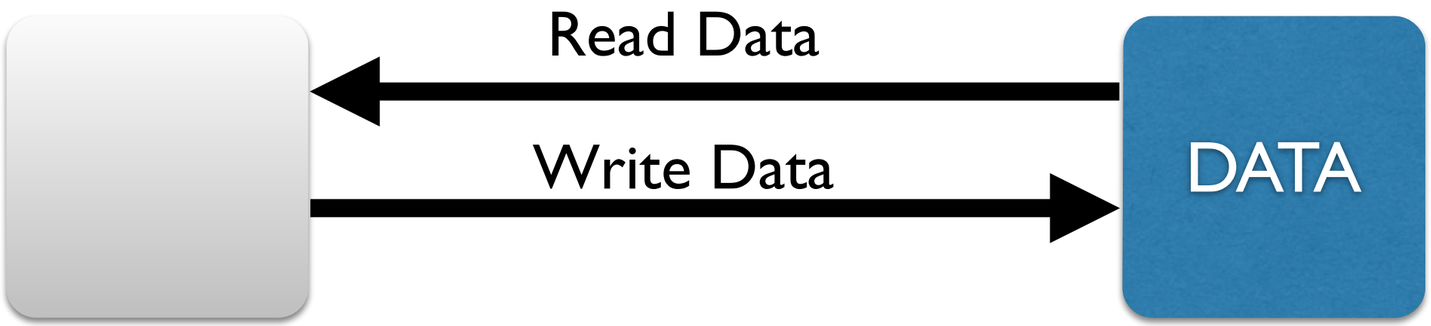
```
public void increment(int accountId,int amount){
    accounts.lock(accountId);
    try{
        Account account = accounts.get(accountId);
        account.balance+=amount;
        accounts.put(accountId, account);
    }finally{
        accounts.unlock(accountId);
    }
}
```

Optimistic Increment

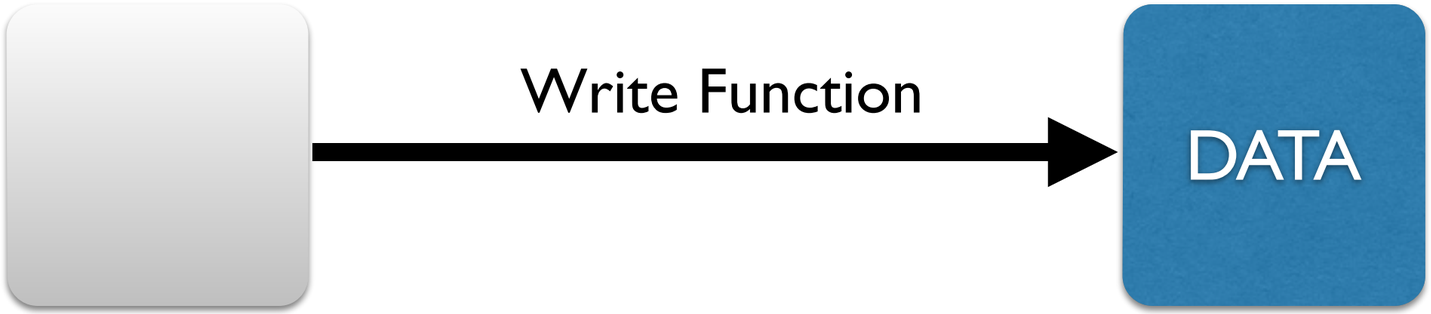
```
public void increment(int accountId,int amount){
    for(;;){
        Account oldAccount = accounts.get(accountId);
        Account newAccount = new Account(oldAccount);
        newAccount.balance+=amount;
        if(accounts.replace(accountId, oldAccount,newAccount)){
            return;
        }
    }
}
```

Optimistic Increment

```
public void increment(int accountId,int amount){
    for(;;){
        Account oldAccount = accounts.get(accountId);
        Account newAccount = new Account(oldAccount);
        newAccount.balance+=amount;
        if(accounts.replace(accountId, oldAccount,newAccount)){
            return;
        }
    }
}
```



Bad: Send Data to Function



Good: Send Function to Data

Increment with runnable

```
private class BalanceTask implements Runnable,Serializable{
    private int accountId, amount;

    public void run() {
        for(;;){
            Account oldAccount = accounts.get(accountId);
            Account newAccount = new Account(oldAccount);
            newAccount.balance+=amount;
            if(accounts.replace(accountId, oldAccount,newAccount)){
                return;
            }
        }
    }
}
```

Increment with runnable

```
public void increment(int accountId, int amount){
    BalanceTask task = new BalanceTask(accountId,amount);
    executorService.executeOnKeyOwner(task,accountId);
}
```

Increment with EntryProcessor

```
class BalanceProcessor
        extends AbstractEntryProcessor<Integer,Account>{
    int amount;
    BalanceProcessor(int amount) {
        this.amount = amount;
    }

    @Override
    public Object process(Map.Entry<Integer, Account> entry) {
        entry.getValue().balance+=amount;
        return null;
    }
}
```

Using entry processor

```
public void increment(int accountId, int amount){
    BalanceProcessor processor = new BalanceProcessor(amount);
    accounts.executeOnKey(accountId, processor);
}
```

Map: In Memory Format

- 2 Options
  - BINARY
  - OBJECT
- Predicates
- EntryProcessor

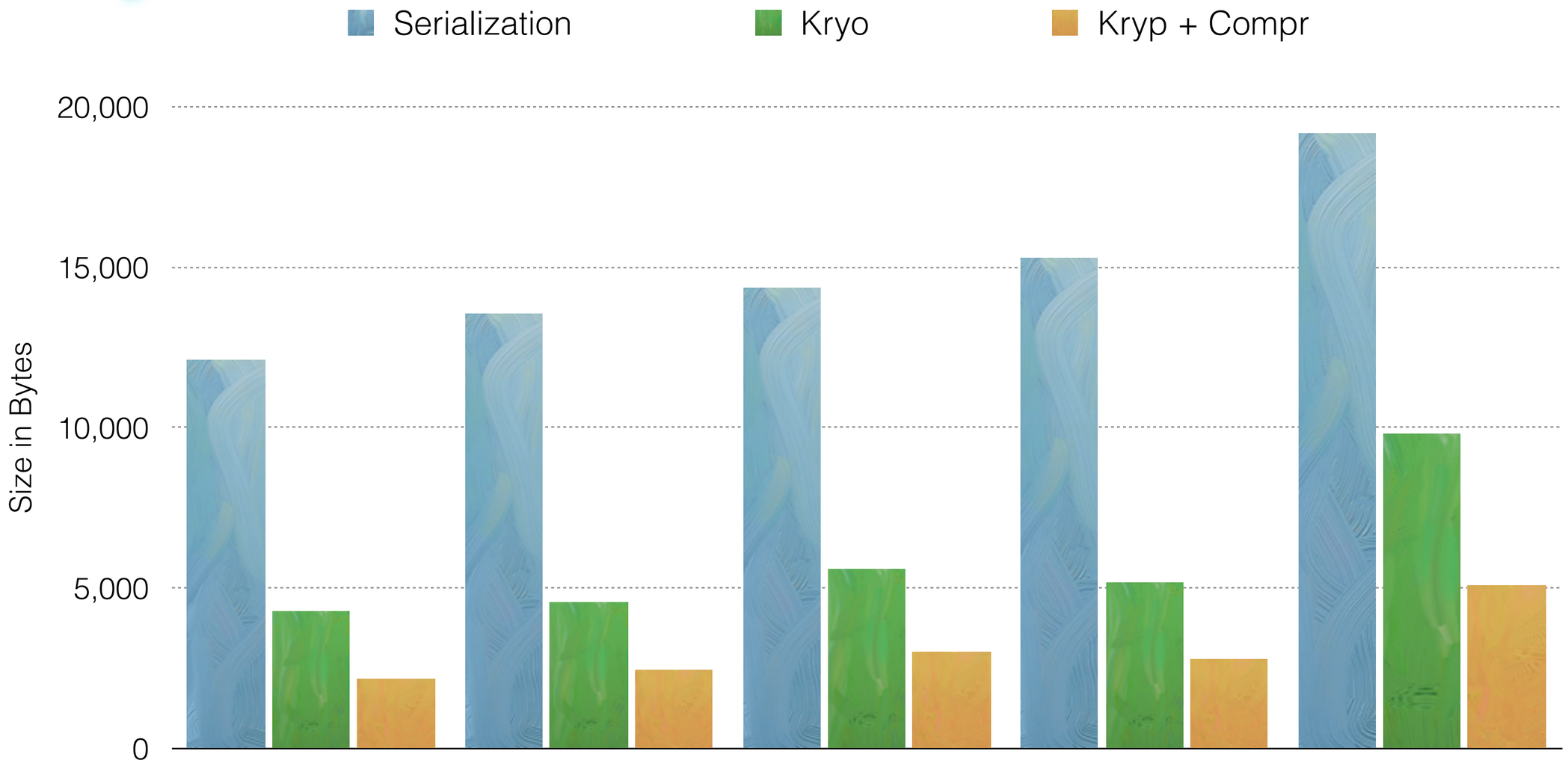
SPI

- You can write your own distributed data-structures
- Example
  - Distributed Actors implementation

Serialization API

- Serializable/Externalizable
- Portable
- ByteArraySerializable
- StreamSerializer
  - Kryo
  - Jackson Smile
  - Protobuf

Kryo Serialization



Pluggable Serialization

```
SerializerConfig productSerializer = new SerializerConfig()
    .setTypeClass(Product.class)
    .setImplementation(new ProductSerializer());

Config config = new Config();
config.getSerializationConfig().addSerializerConfig(productSerializer);
HazelcastInstance hz = Hazelcast.newHazelcastInstance(config);
```

Hazelcast 3.2 Map Reduce

```
IMap<Integer,Account> accounts = hz.getMap("accounts");
Map<Integer,Integer> result = accounts
    .map(new AccountSumMapper())
    .reduce(new AccountSumReducer())
    .submit()
```

Enterprise vs Community edition

- Support contracts
- Elastic Memory
- Security
- Management Center

The Future

- Topologies
- Hosted Management Center
- Dynamic Creation
- Map of Maps
- Tiered storage

Questions?

Good Question?
Get a Hazelcast
book!

You can find us at the
Hazelcast booth!