



# Exploring Terracotta JVM clustering

---

Geert Bevin, Senior Developer  
Terracotta, Inc.

<http://terracotta.org>

[gbevin@terracottatech.com](mailto:gbevin@terracottatech.com)

# Who is Geert?

- developer at Terracotta - <http://terracotta.org>
- founder of Uwyn - <http://uwyn.com>
- founder of RIFE - <http://rifers.org>
- contributor to many open-source projects: RIFE, OpenLaszlo, Gentoo Linux, Bla-bla List, Drone
- Sun Java Champion
- creator of native Java continuations
- biker and gamer

# What is Terracotta?

# What is Terracotta?

Open Source Clustering For Java  
Scalability and Availability for the JVM

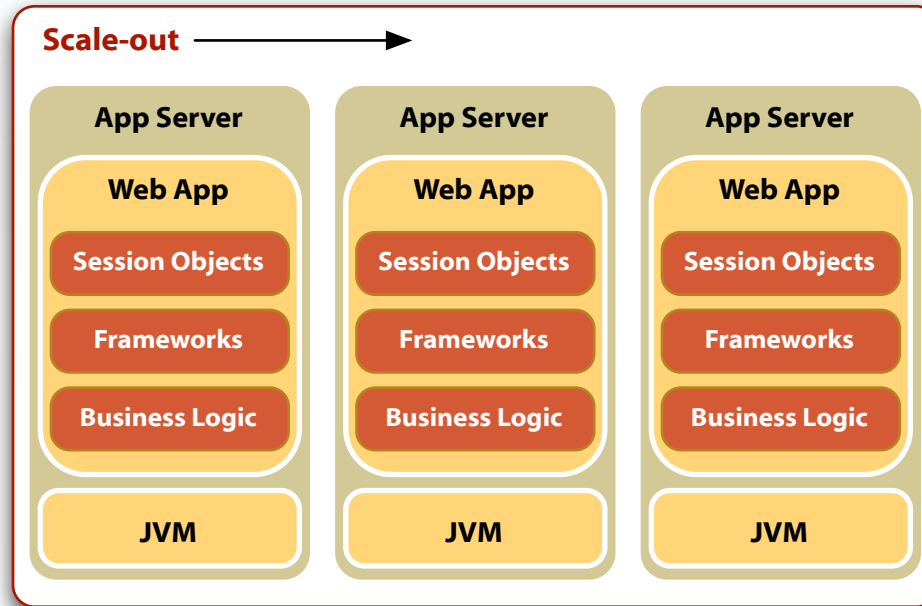
# Terracotta Approach

## ■ Today's Reality

- Scale out is complex
- Requires custom Java code

## ■ Our different approach

- Cluster the JVM
- Eliminate need for custom code



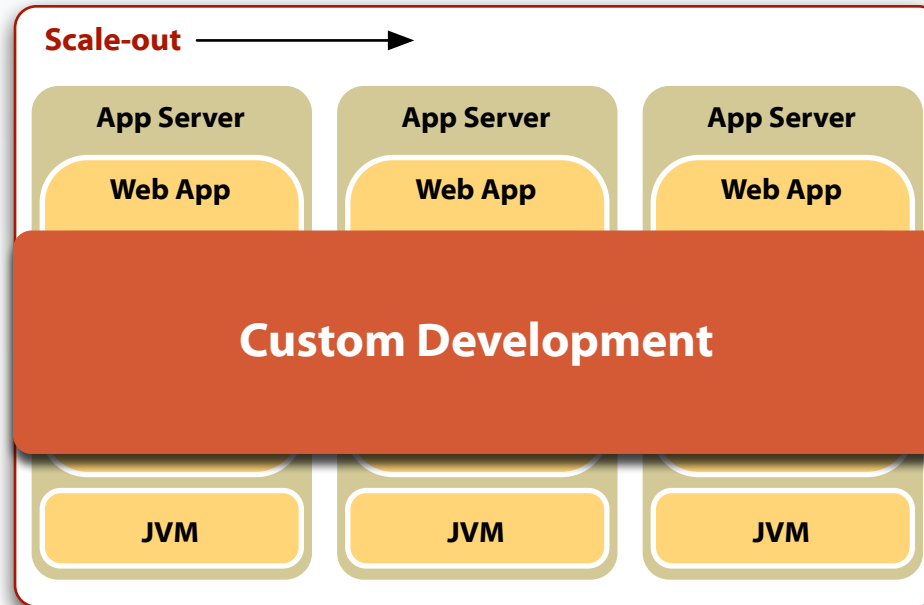
# Terracotta Approach

## ■ Today's Reality

- Scale out is complex
- Requires custom Java code

## ■ Our different approach

- Cluster the JVM
- Eliminate need for custom code



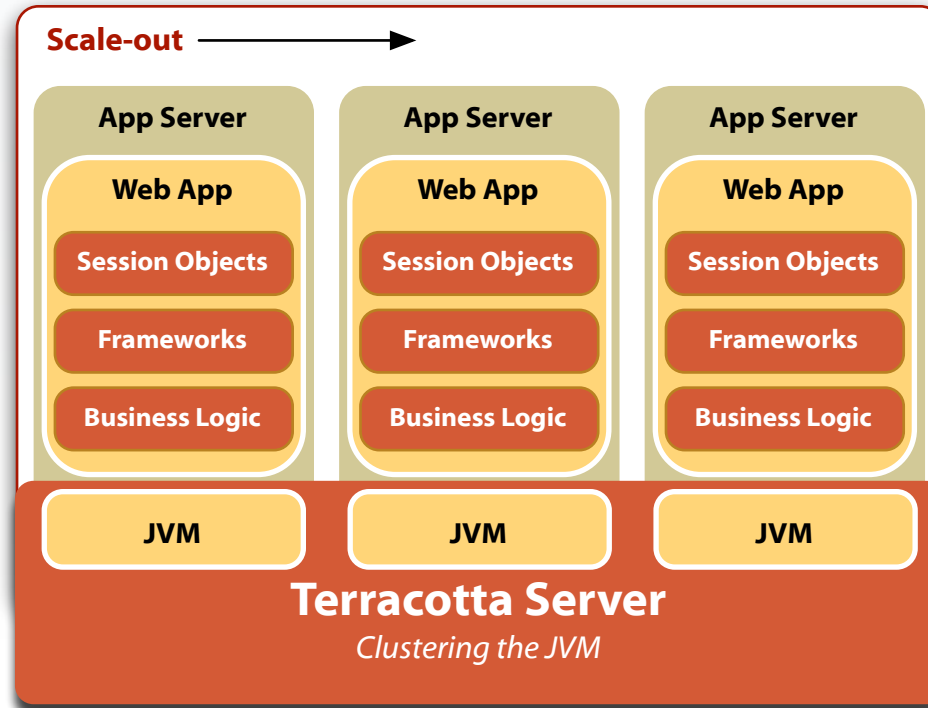
# Terracotta Approach

## ■ Today's Reality

- Scale out is complex
- Requires custom Java code

## ■ Our different approach

- Cluster the JVM
- Eliminate need for custom code



# Main features

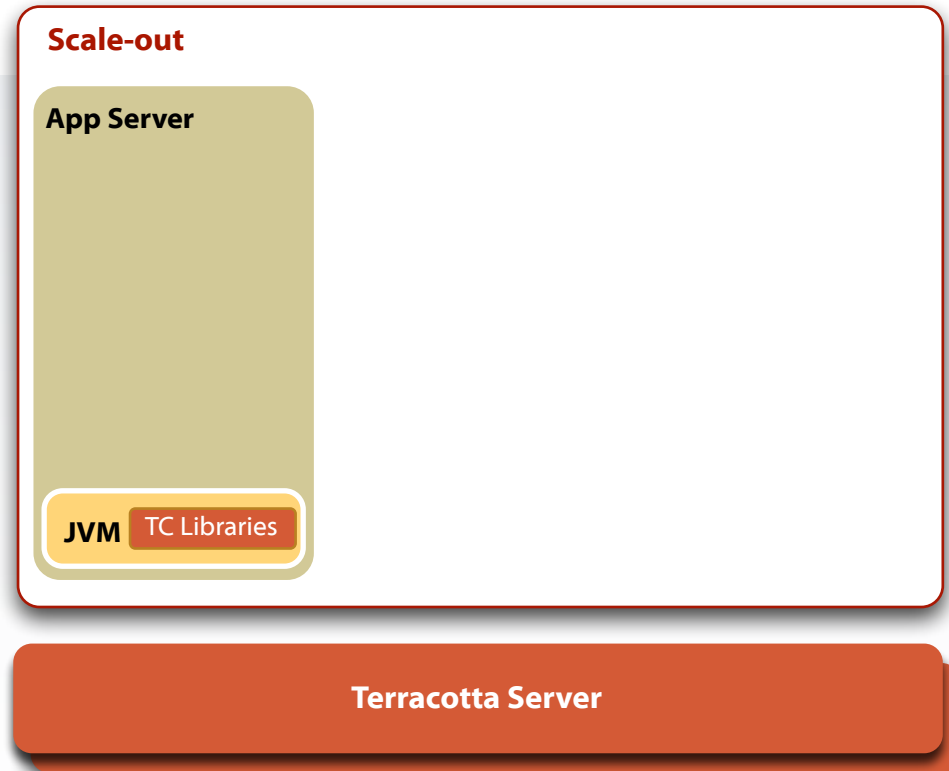
- **Heap Level Replication**
  - Declarative
  - No Serialization
  - Fine Grained / Field Level
  - Only Where Resident
- **JVM Coordination**
  - Distributed Synchronized
  - Distributed wait()/notify()
  - Fine Grained Locking
- **Large Virtual Heaps**
  - As large as available disk
  - Dynamic paging

Scale-out

Terracotta Server

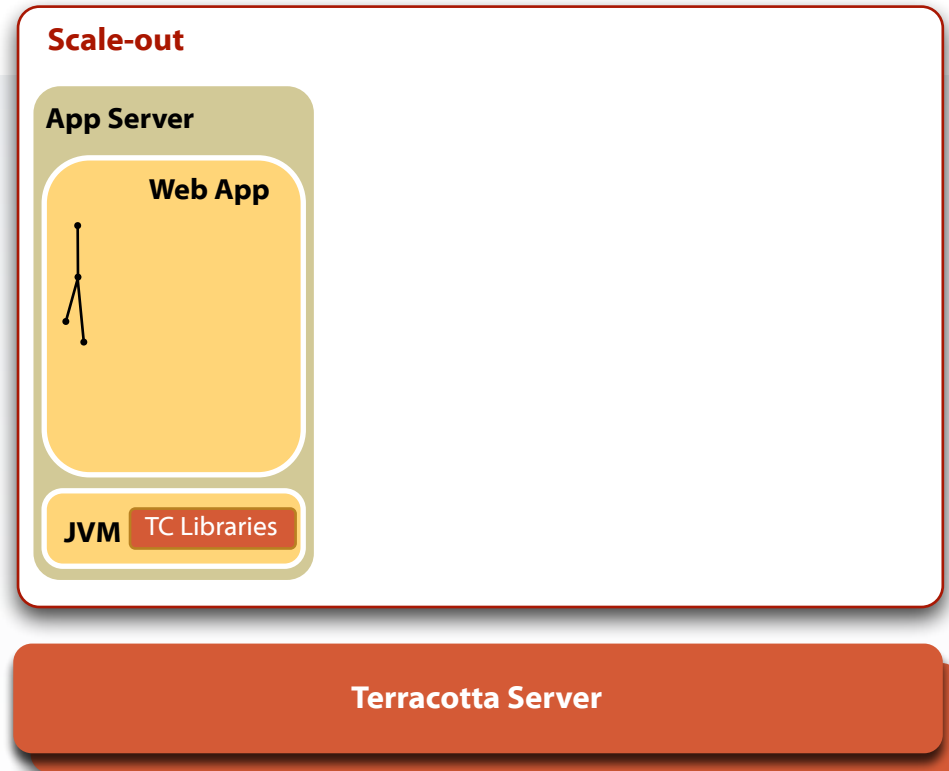
# Main features

- **Heap Level Replication**
  - Declarative
  - No Serialization
  - Fine Grained / Field Level
  - Only Where Resident
- **JVM Coordination**
  - Distributed Synchronized
  - Distributed wait()/notify()
  - Fine Grained Locking
- **Large Virtual Heaps**
  - As large as available disk
  - Dynamic paging



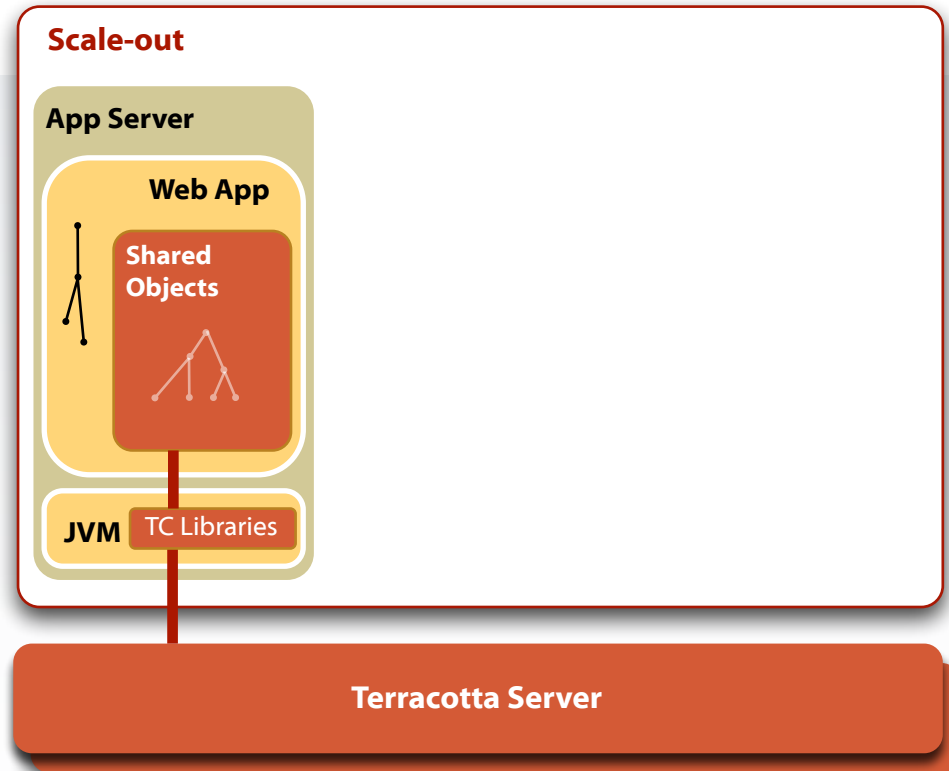
# Main features

- **Heap Level Replication**
  - Declarative
  - No Serialization
  - Fine Grained / Field Level
  - Only Where Resident
- **JVM Coordination**
  - Distributed Synchronized
  - Distributed wait()/notify()
  - Fine Grained Locking
- **Large Virtual Heaps**
  - As large as available disk
  - Dynamic paging



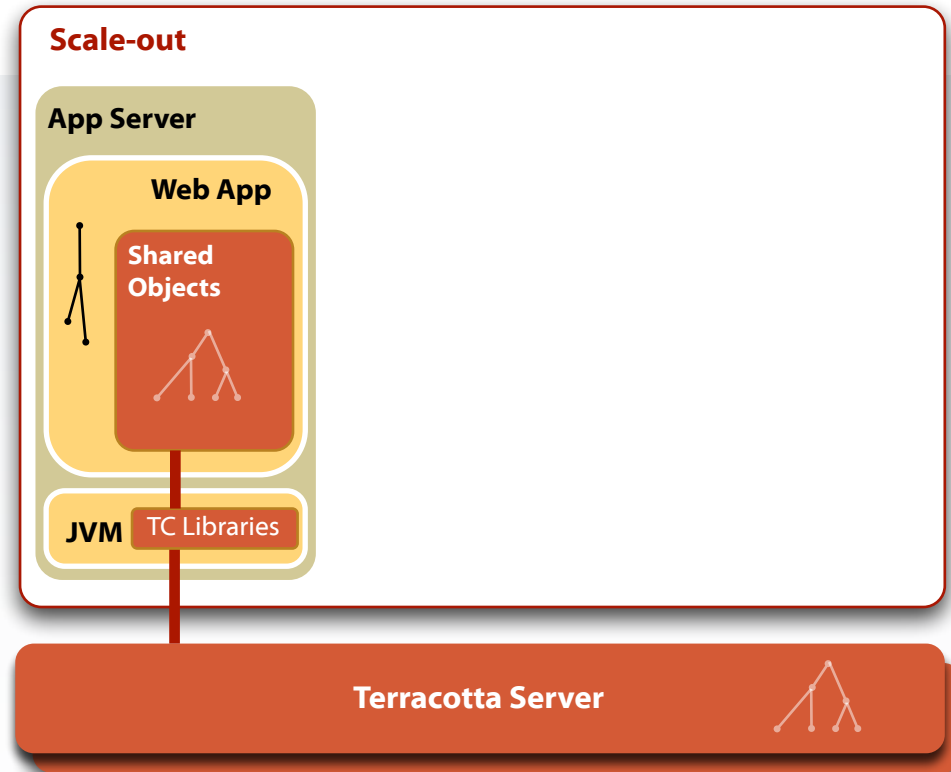
# Main features

- **Heap Level Replication**
  - Declarative
  - No Serialization
  - Fine Grained / Field Level
  - Only Where Resident
- **JVM Coordination**
  - Distributed Synchronized
  - Distributed wait()/notify()
  - Fine Grained Locking
- **Large Virtual Heaps**
  - As large as available disk
  - Dynamic paging



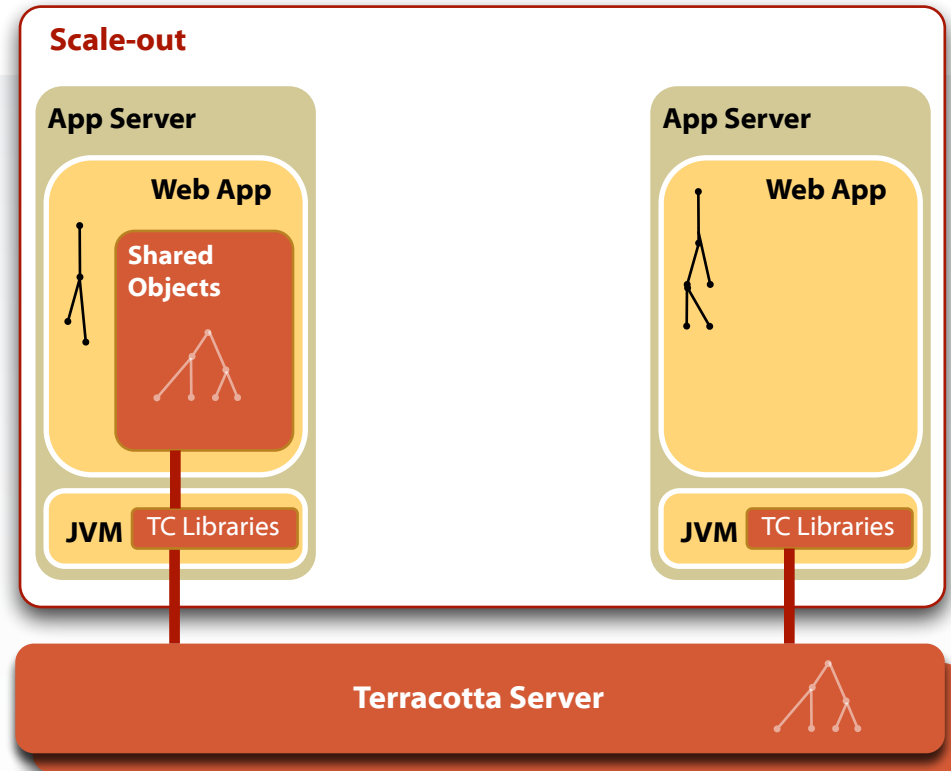
# Main features

- **Heap Level Replication**
  - Declarative
  - No Serialization
  - Fine Grained / Field Level
  - Only Where Resident
- **JVM Coordination**
  - Distributed Synchronized
  - Distributed wait()/notify()
  - Fine Grained Locking
- **Large Virtual Heaps**
  - As large as available disk
  - Dynamic paging



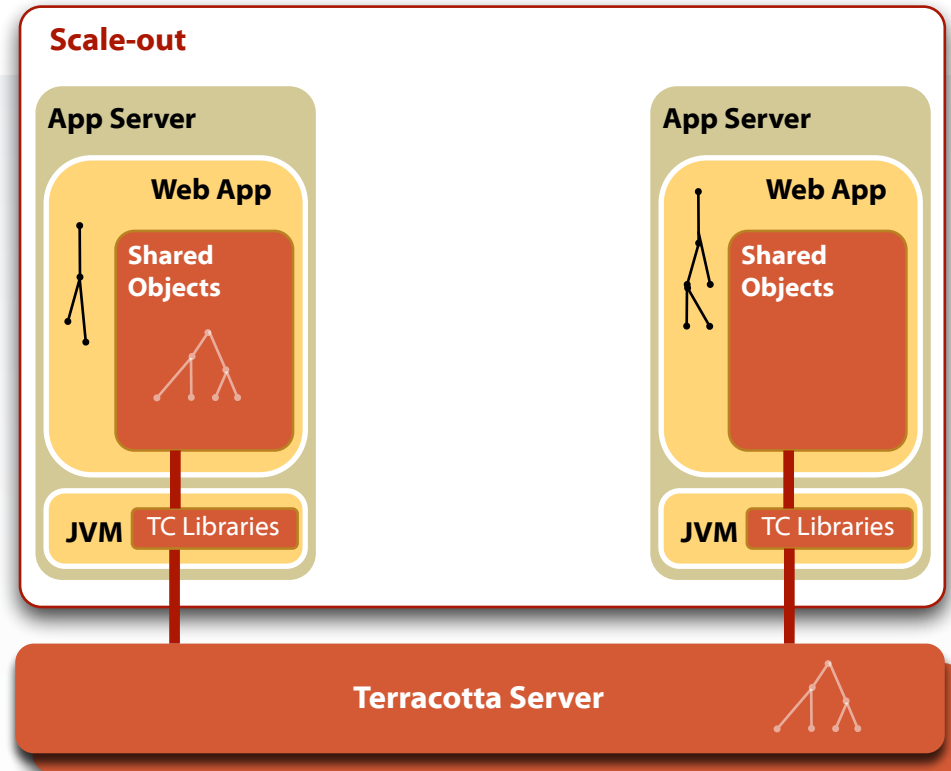
# Main features

- **Heap Level Replication**
  - Declarative
  - No Serialization
  - Fine Grained / Field Level
  - Only Where Resident
- **JVM Coordination**
  - Distributed Synchronized
  - Distributed wait()/notify()
  - Fine Grained Locking
- **Large Virtual Heaps**
  - As large as available disk
  - Dynamic paging



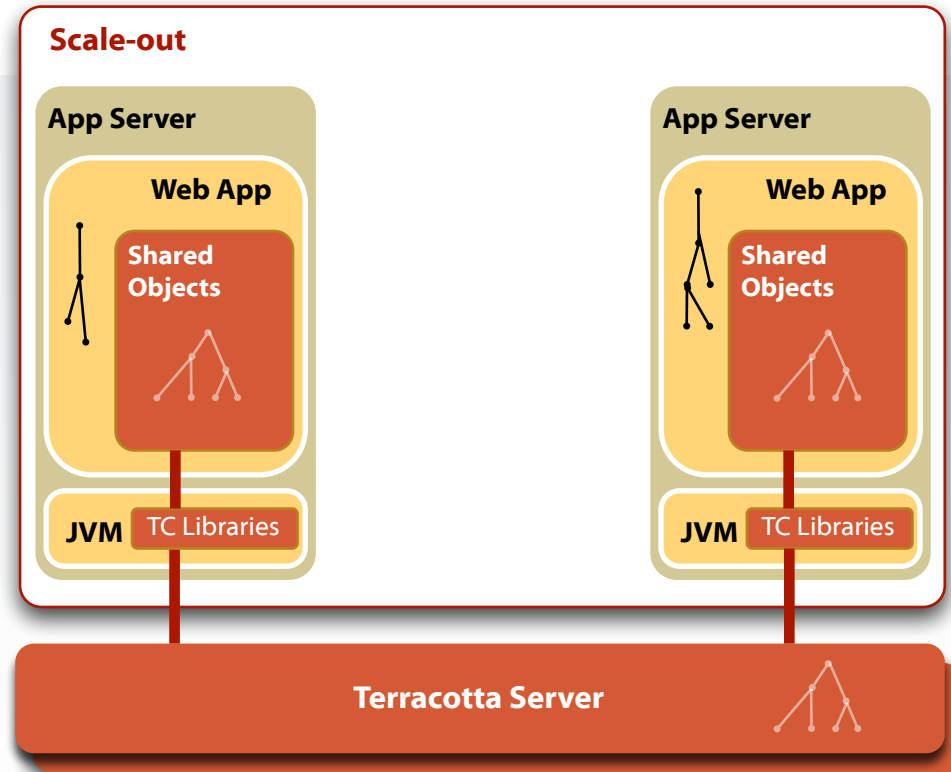
# Main features

- **Heap Level Replication**
  - Declarative
  - No Serialization
  - Fine Grained / Field Level
  - Only Where Resident
- **JVM Coordination**
  - Distributed Synchronized
  - Distributed wait()/notify()
  - Fine Grained Locking
- **Large Virtual Heaps**
  - As large as available disk
  - Dynamic paging



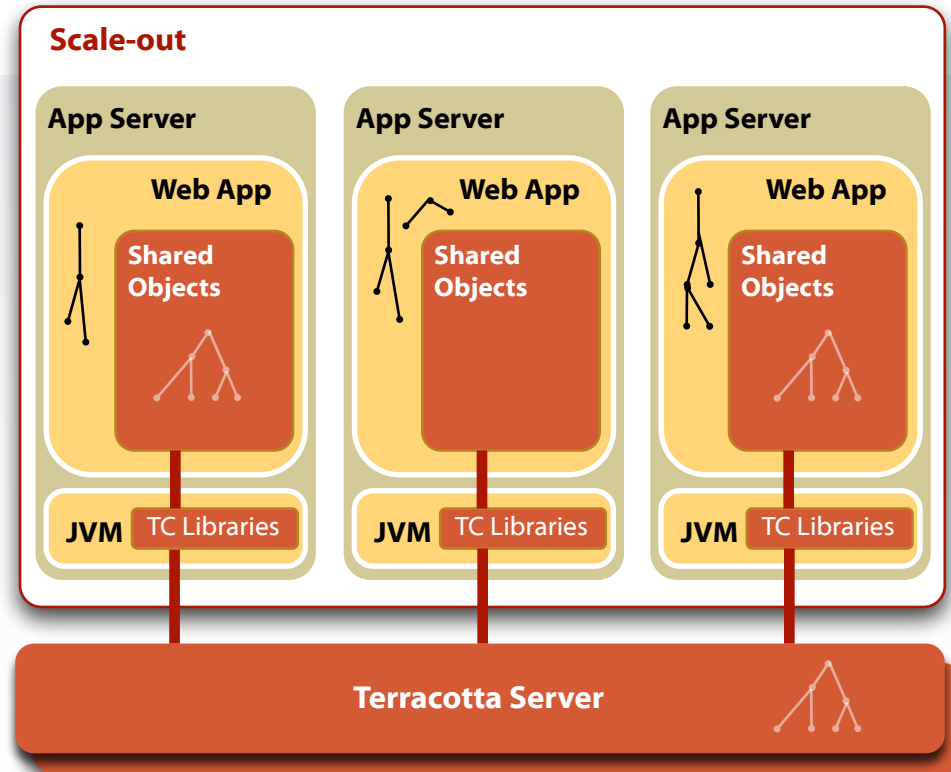
# Main features

- **Heap Level Replication**
  - Declarative
  - No Serialization
  - Fine Grained / Field Level
  - Only Where Resident
- **JVM Coordination**
  - Distributed Synchronized
  - Distributed wait()/notify()
  - Fine Grained Locking
- **Large Virtual Heaps**
  - As large as available disk
  - Dynamic paging



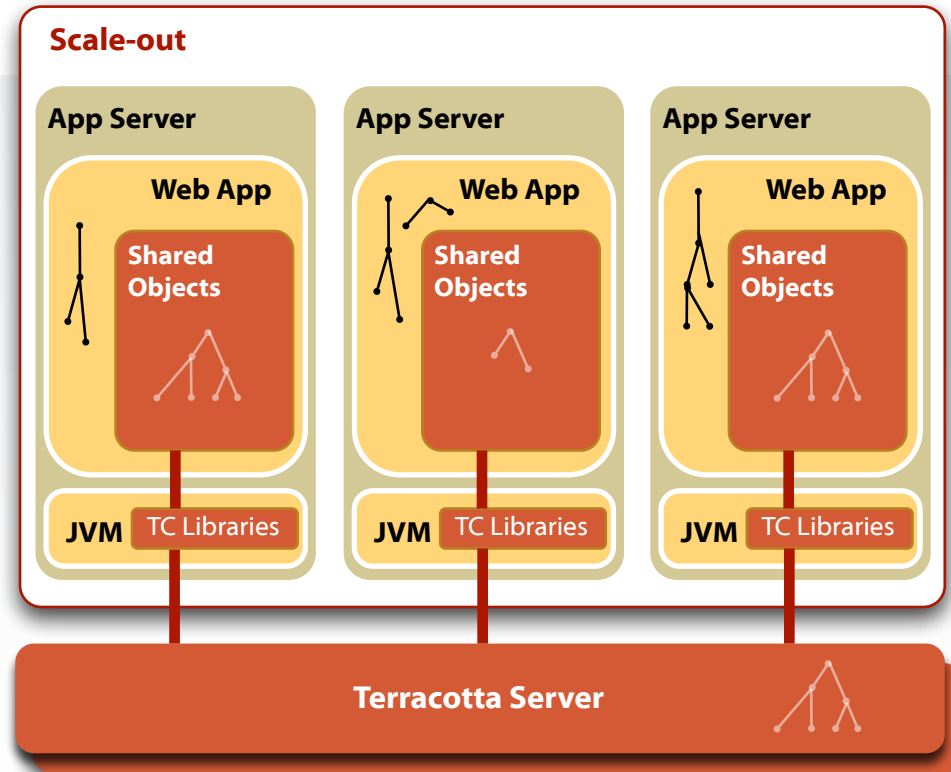
# Main features

- **Heap Level Replication**
  - Declarative
  - No Serialization
  - Fine Grained / Field Level
  - Only Where Resident
- **JVM Coordination**
  - Distributed Synchronized
  - Distributed wait()/notify()
  - Fine Grained Locking
- **Large Virtual Heaps**
  - As large as available disk
  - Dynamic paging



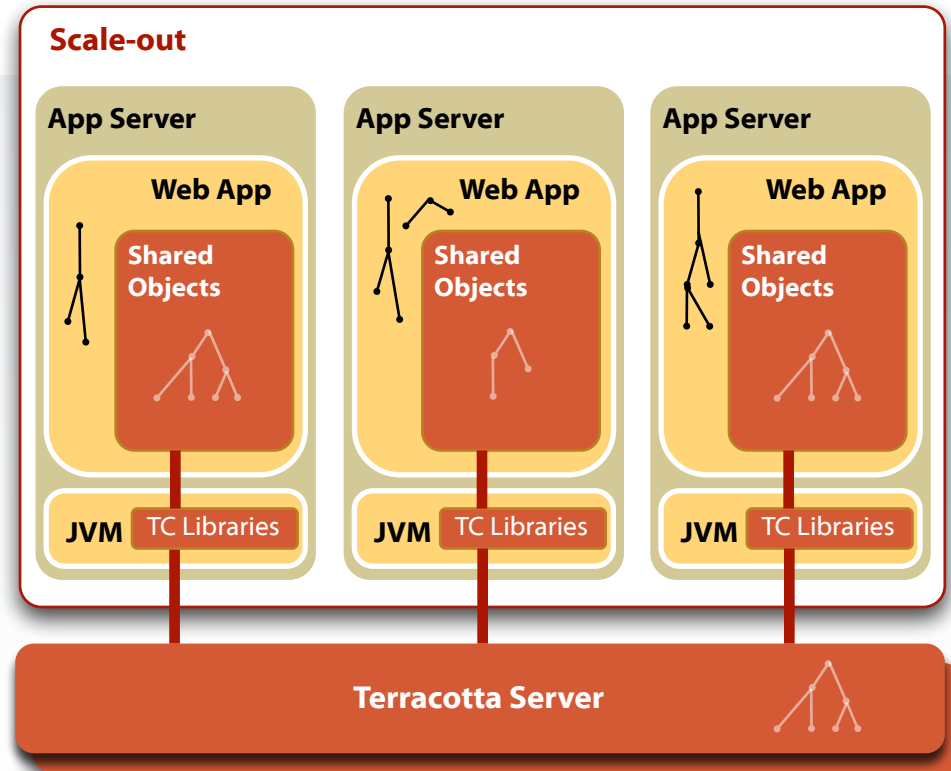
# Main features

- **Heap Level Replication**
  - Declarative
  - No Serialization
  - Fine Grained / Field Level
  - Only Where Resident
- **JVM Coordination**
  - Distributed Synchronized
  - Distributed wait()/notify()
  - Fine Grained Locking
- **Large Virtual Heaps**
  - As large as available disk
  - Dynamic paging



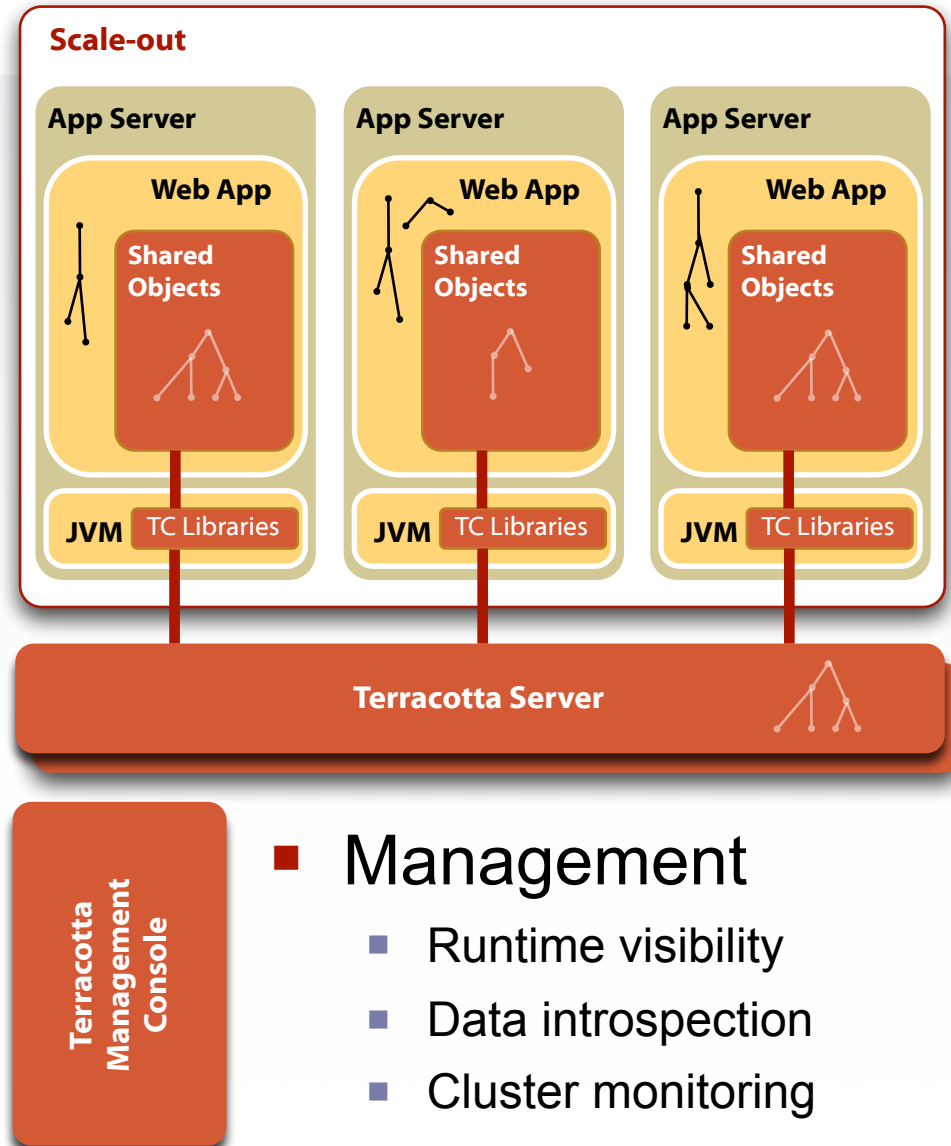
# Main features

- **Heap Level Replication**
  - Declarative
  - No Serialization
  - Fine Grained / Field Level
  - Only Where Resident
- **JVM Coordination**
  - Distributed Synchronized
  - Distributed wait()/notify()
  - Fine Grained Locking
- **Large Virtual Heaps**
  - As large as available disk
  - Dynamic paging



# Main features

- **Heap Level Replication**
  - Declarative
  - No Serialization
  - Fine Grained / Field Level
  - Only Where Resident
- **JVM Coordination**
  - Distributed Synchronized
  - Distributed wait()/notify()
  - Fine Grained Locking
- **Large Virtual Heaps**
  - As large as available disk
  - Dynamic paging





# Hello World

# Hello World - *tutorial/HelloWorld.java*

```
package tutorial;

import java.util.*;

public class HelloWorld {
    private List<Date> dates = new ArrayList<Date>();

    public void sayHello() {
        synchronized (dates) {
            dates.add(new Date());
            for (Date date : dates) {
                System.out.println("Hello " + date);
            }
        }
    }

    public static void main(String[] args) {
        new HelloWorld().sayHello();
    }
}
```

# Hello World - *tc-config.xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<tc:tc-config xmlns:tc="http://www.terracotta.org/config">
  <application>
    <dso>
      <roots>
        <root><field-name>tutorial.HelloWorld.dates</field-name></root>
      </roots>
      <locks>
        <autolock>
          <method-expression>* tutorial.HelloWorld*.*(..)</method-expression>
        </autolock>
      </locks>
    </dso>
  </application>
</tc:tc-config>
```

# Hello World - *trying it out*

- **Start the server:**

```
$ start-tc-server.sh
```

- **Start a client:**

```
$ dso-java.sh -Dtc.config=tc-config.xml  
tutorial.HelloWorld
```

- **Output:**

```
Hello Thu Dec 13 17:46:58 CET 2007
```

- **After starting the 2<sup>nd</sup> client:**

```
Hello Thu Dec 13 17:46:58 CET 2007  
Hello Thu Dec 13 17:47:06 CET 2007
```

- **The state is preserved in the server between executions**



# Enhanced Hello World

Using `java.util.concurrent`

# Coordination - *HelloWorldConcurrent.java*

```
package tutorial;

import java.util.*;
import java.util.concurrent.CyclicBarrier;

public class HelloWorldConcurrent {
    private List<Date> dates = new ArrayList<Date>();
    private CyclicBarrier barrier;

    public void sayHello() throws Exception {
        barrier = new CyclicBarrier(Integer.getInteger("nodes", 1));
        barrier.await();
        synchronized (dates) {
            dates.add(new Date());
            for (Date date : dates) {
                System.out.println("Hello " + date);
            }
        }
    }

    public static void main(String[] args) throws Exception {
        new HelloWorldConcurrent().sayHello();
    }
}
```

# Coordination - *tc-config-concurrent.xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<tc:tc-config xmlns:tc="http://www.terracotta.org/config">
  <application>
    <dso>
      <roots>
        <root><field-name>tutorial.HelloWorldConcurrent.dates</field-name></root>
        <root><field-name>tutorial.HelloWorldConcurrent.barrier</field-name></root>
      </roots>
      <locks>
        <autolock>
          <method-expression>* tutorial.HelloWorldConcurrent*.*(..)</method-expression>
        </autolock>
      </locks>
    </dso>
  </application>
</tc:tc-config>
```

## Coordination - *trying it out*

- Start the server:

```
$ start-tc-server.sh
```

- Start a client:

```
$ dso-java.sh -Dtc.config=tc-config-concurrent.xml  
-Dnodes=2 tutorial.HelloWorldConcurrent
```

- It just hangs there

- After starting the 2<sup>nd</sup> client:

```
Hello Thu Dec 13 19:12:38 CET 2007  
Hello Thu Dec 13 19:13:56 CET 2007
```

- The cyclic barrier is clustered across JVMs



# Wrapping up

# Wrapping up

- Terracotta works under the covers: follows the JMM
- Don't code for it, architect for it
- Endless clustering possibilities
- Be creative, imagine what **you** can do with this:
  - No serialization
  - Fine-grained field-level changes
  - Heap-level replication to selectively share object graphs across the cluster
  - Distributed wait/notify and synchronized
  - Large virtual heap
  - Distributed method invocations
  - Runtime monitoring and control

# Q&A

<http://terracotta.org>  
[gbevin@terracottatech.com](mailto:gbevin@terracottatech.com)

---

