



Terracotta DSO

Architecture Overview



TERRACOTTA

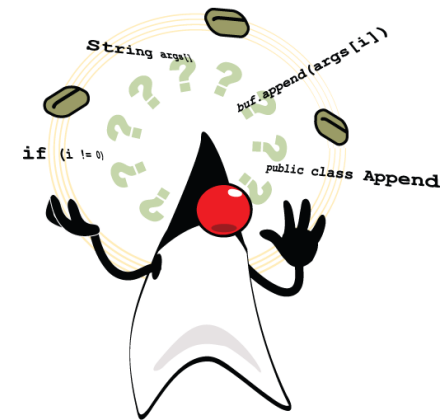
Simone Federici
Java Architect

s.federici@k-tech.it



Who I am...

- Simone Federici
 - Java Architect for K-Tech S.r.l.
 - APM Consultant for Wily/CA



Staff di Java Italian Portal
Coordinatore del JugRoma



Java Italian Portal

www.javaportal.it

Community di professionisti appassionati di java e OpenSource

Sul portale puoi trovare:

- **Articoli tecnici**, **forum**,
eventi, **news**, **iniziative**

Collabora con noi! Come?

Hai una tesi o altra documentazione (articoli) su java?



Pubblicala su JavaPortal!

Java Portal è un progetto senza scopi di lucro di **keyTECH**



Argomenti trattati



Clustering

Silver bullets



Session Clustering

Spring and POJO Clustering

Distribuite Cache

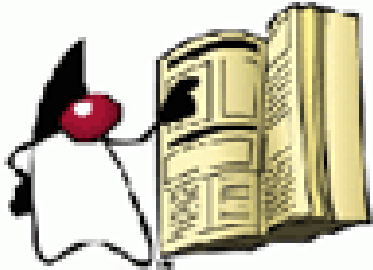


JEE Architecture





Obiettivo



Questo seminario ha come obiettivo, mettere in risalto l'uso di Terracotta DSO come soluzione per risolvere in modo semplice, trasparente e opensource, tutte le problematiche che riguardano il clustering.



Agenda

(1) Introduzione Terracotta DSO

(2) Clustering with Terracotta

(3) Architettura JEE

(4) Demo



Introduzione

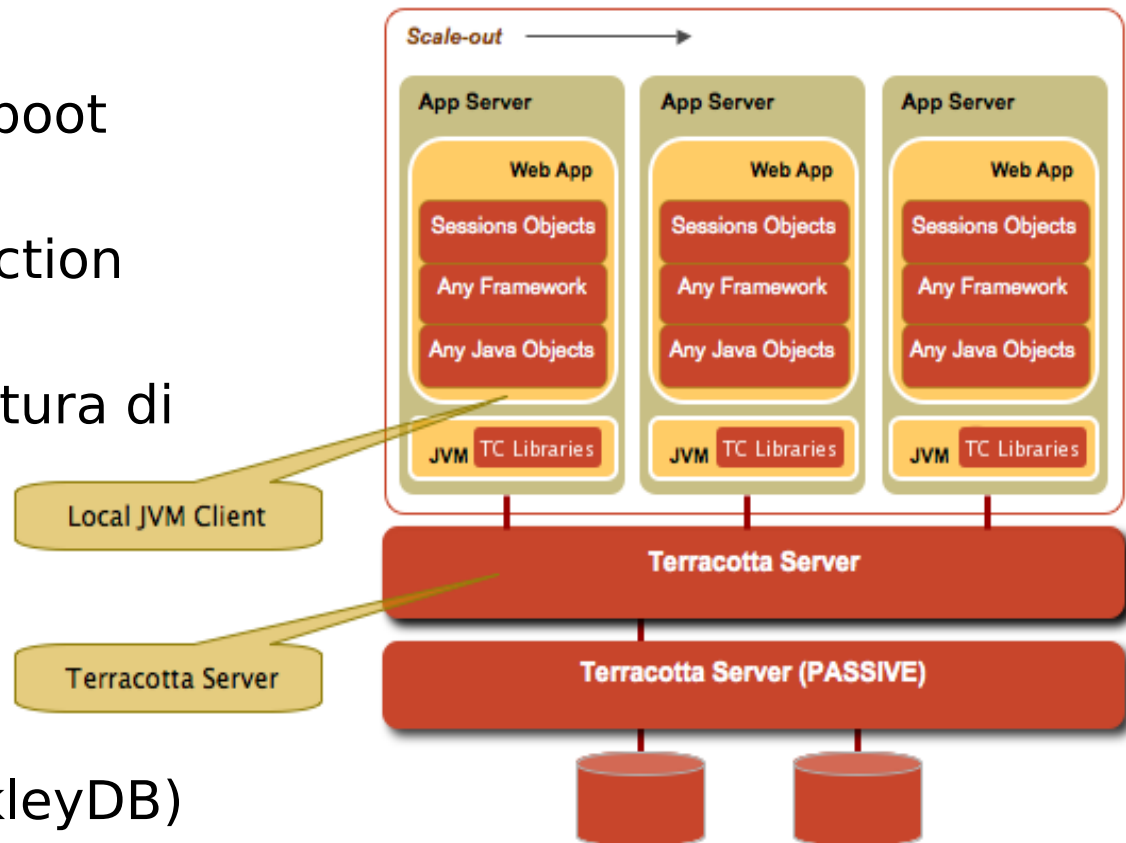
- Soluzione opensource per il clustering a livello di JVM.
- Soddisfa i requisiti di **scalabilità e affidabilità**.
- Clustering **trasparente a livello applicativo**
- Fa interagire le applicazioni distribuite come se fossero su una unica JVM.
- In una JVM i threads interagiscono gli uni con gli altri attraverso il cambiamento degli oggetti residenti nell'HEAP e attraverso le primitive concorrenti. ('synchronized' keyword, wait(), notify() e notifyAll()).
- Terracotta estende il loro significato per una sincronizzazione distribuita.





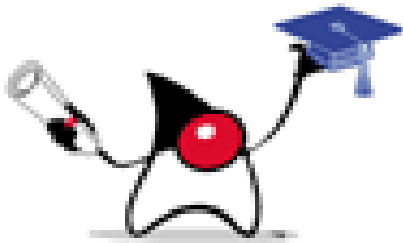
Terracotta DSO

- Client/Server p2p
- TC Library caricate nel boot classpath.
- Cluster capabilities injection
- Disaccoppiamento tra applicazione e infrastruttura di cluster DSO.
- Threads Coordination
- Scalability
- High availaility
- no Double master (berkleyDB)





Delevopment Benefit



- Cluster non applicativo, è un cluster di JVM.
- Separate of concerns tra Business Logic e Infrastructure Object.
- NESSUNA API Java NUOVA da imparare.
- Non c'è serializzazione.
- Non ci sono metodi CUSTOM da implementare per la replicazione.
- Un programmatore deve semplicemente conoscere la programmazione concorrente.



Features

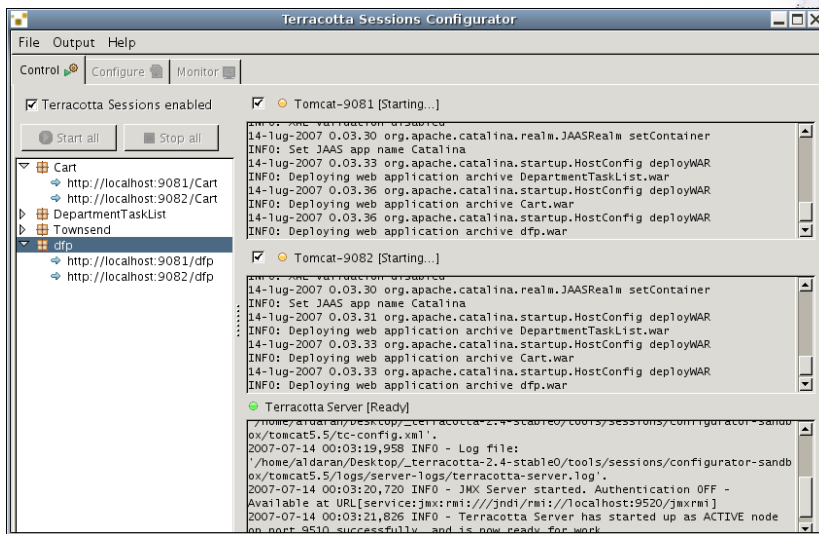
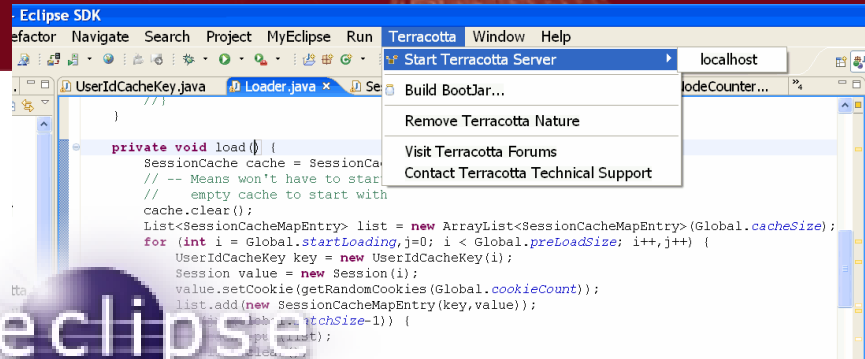
- Utilizzo della rete ottimizzato
- no full replications
- Update degli oggetti possibile
- JMX per il monitor del server
- Admin console
- Aumento della massima capacita di memoria della jvm.





Terracotta Tools

- Eclipse plugin
- Session Cofigurator
- Admin Conosle



Terracotta AdminClient

File	Help	Iteration	Start time	Elapsed time (ms.)	Begin object count	Candidate garbage count	Actual garbage count
DSO		46	Jan 17, 2006 4:25:57 PM	172	1,794	34	34
Roots		45	Jan 17, 2006 4:25:07 PM	437	1,736	20	20
Classes		44	Jan 17, 2006 4:24:17 PM	157	1,692	26	26
Locks		43	Jan 17, 2006 4:23:27 PM	157	1,645	32	32
Clerks		42	Jan 17, 2006 4:22:36 PM	140	1,691	125	125
Cache activity		41	Jan 17, 2006 4:21:46 PM	172	1,637	26	26
Transaction rate		40	Jan 17, 2006 4:20:56 PM	375	1,603	38	38
Cache hit ratio		39	Jan 17, 2006 4:20:06 PM	125	1,550	34	34
Garbage collection		38	Jan 17, 2006 4:19:16 PM	125	1,512	32	32
All statistics		37	Jan 17, 2006 4:18:25 PM	125	1,461	26	26
localhost:9520		36	Jan 17, 2006 4:17:35 PM	110	1,420	30	30
		35	Jan 17, 2006 4:16:45 PM	125	1,373	30	30
		34	Jan 17, 2006 4:15:55 PM	109	1,325	28	28
		33	Jan 17, 2006 4:15:05 PM	109	1,280	30	30
		32	Jan 17, 2006 4:14:15 PM	110	1,232	28	28
		31	Jan 17, 2006 4:13:25 PM	219	1,187	32	32
		30	Jan 17, 2006 4:12:34 PM	250	1,137	28	28
		29	Jan 17, 2006 4:11:44 PM	93	1,091	24	24
		28	Jan 17, 2006 4:10:54 PM	94	1,039	24	24
		27	Jan 17, 2006 4:10:04 PM	94	991	26	26
		26	Jan 17, 2006 4:09:14 PM	94	942	24	24
		25	Jan 17, 2006 4:08:24 PM	167	897	30	30
		24	Jan 17, 2006 4:07:34 PM	172	840	24	24
		23	Jan 17, 2006 4:06:44 PM	47	786	20	20
		22	Jan 17, 2006 4:05:54 PM	47	747	32	32
		21	Jan 17, 2006 4:05:04 PM	47	698	30	30

Console | rascal:9520

```

1 2006-01-17 16:25:57.984 [GC] INFO com.t.obejctserver.api.ObjectManager - GC: notifying gc complete ...
1 2006-01-17 16:25:57.984 [GC] INFO com.t.obejctserver.api.ObjectManager - GC: 2-pass rescue: true
1 2006-01-17 16:25:57.984 [GC] INFO com.t.obejctserver.api.ObjectManager - GC: rescue 1 time: : 141 ms.
1 2006-01-17 16:25:57.984 [GC] INFO com.t.obejctserver.api.ObjectManager - GC: rescue 2 time: : 0 ms.
1 2006-01-17 16:25:57.984 [GC] INFO com.t.obejctserver.api.ObjectManager - GC: rescued gc time: : 0 ms.
1 2006-01-17 16:25:57.984 [GC] INFO com.t.obejctserver.api.ObjectManager - GC: total gc time: : 172 ms.
1 2006-01-17 16:25:57.984 [GC] INFO com.t.obejctserver.api.ObjectManager - GC: STOP 46
  
```

Added new DSO client: 127.0.0.1:1319

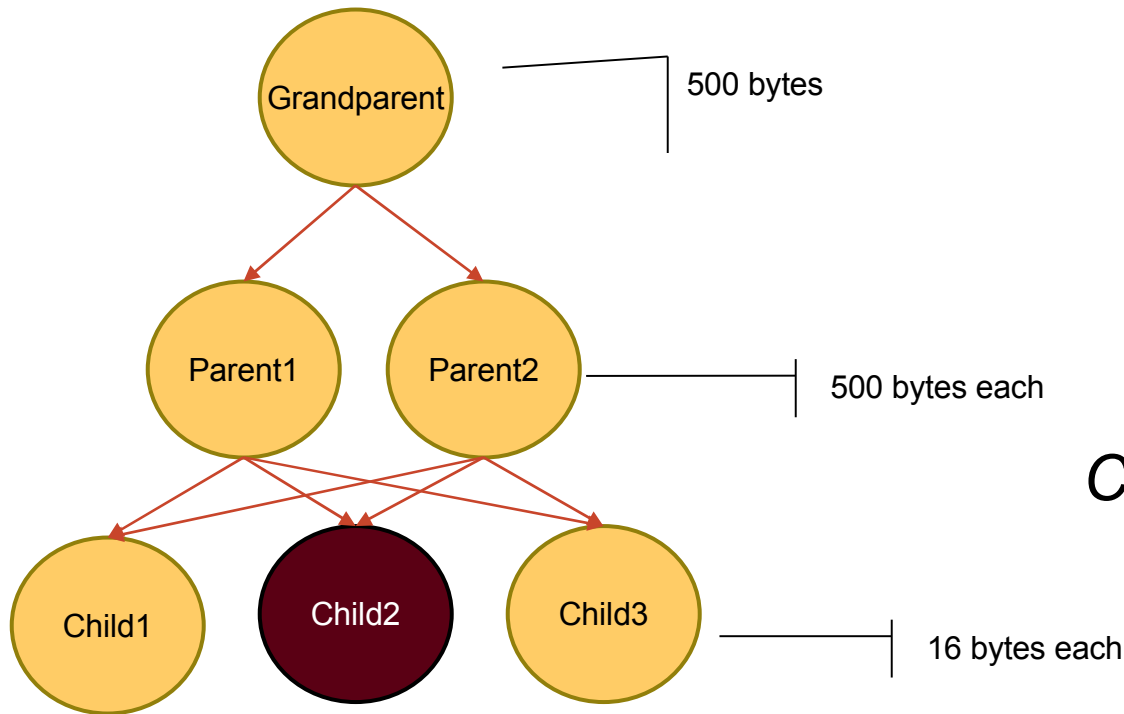


HTTP Session Clustering

- La memoria occupata dalle sessioni cresce 1:1 con gli utenti che usano il sistema.
- C'è quindi bisogno di una cluster orizzontale, con un load balancer (sticky).
- Terracotta supporta la replicazione delle sessioni senza bisogno di API speciali, senza bisogno della serializzazione e non introduce i noti problemi della full replication.



Serializzazione scambio dati non ottimizzato



Total session size:
1548 bytes

Cost of 16 byte change:

1548 bytes

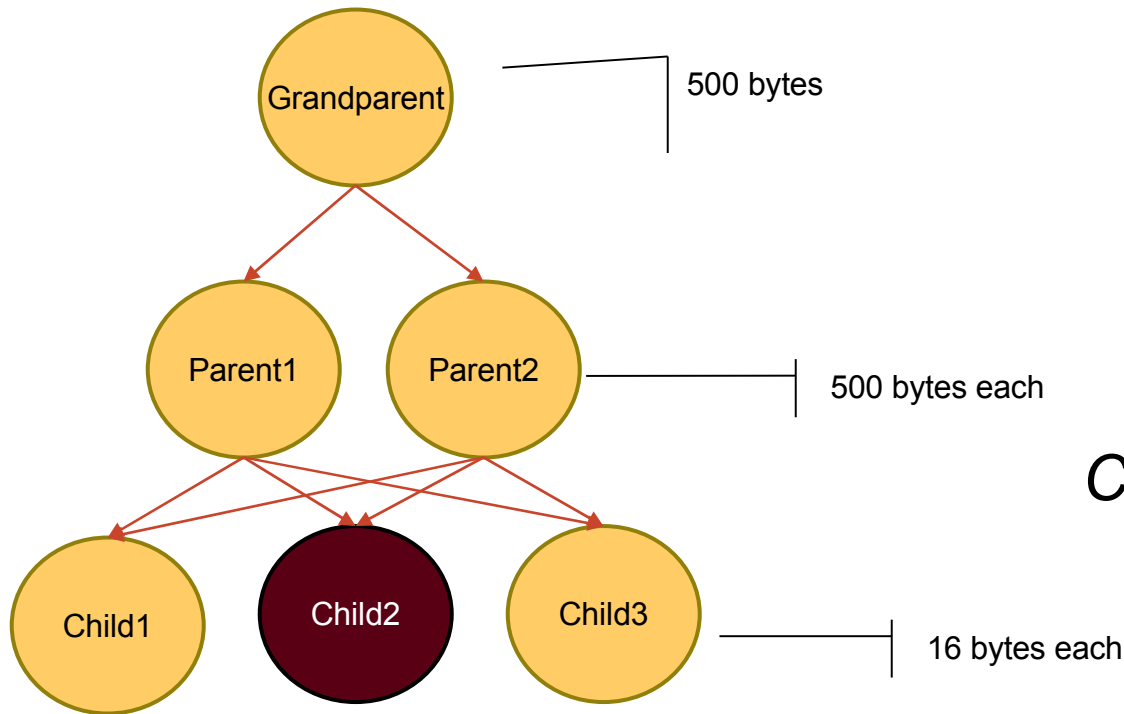


Fine-Grained Change Replication

- Ogni cambiamento ad un oggetto genera una transazione
- Tutti i cambiamenti fatti all'interno di un blocco sincronizzato vengono inviati in una unica transazione.
- Le transazioni contengono solo i dati dei campi cambiati.
- Le transazioni vengono inviate al server che le replica alle altre JVM.
- Il server filtra i dati delle transazioni in modo da mandare i cambiamenti dei soli oggetti posseduti dalle singole JVM.



Terracotta DSO scambio dati: ottimale



Total session size:
1548 bytes

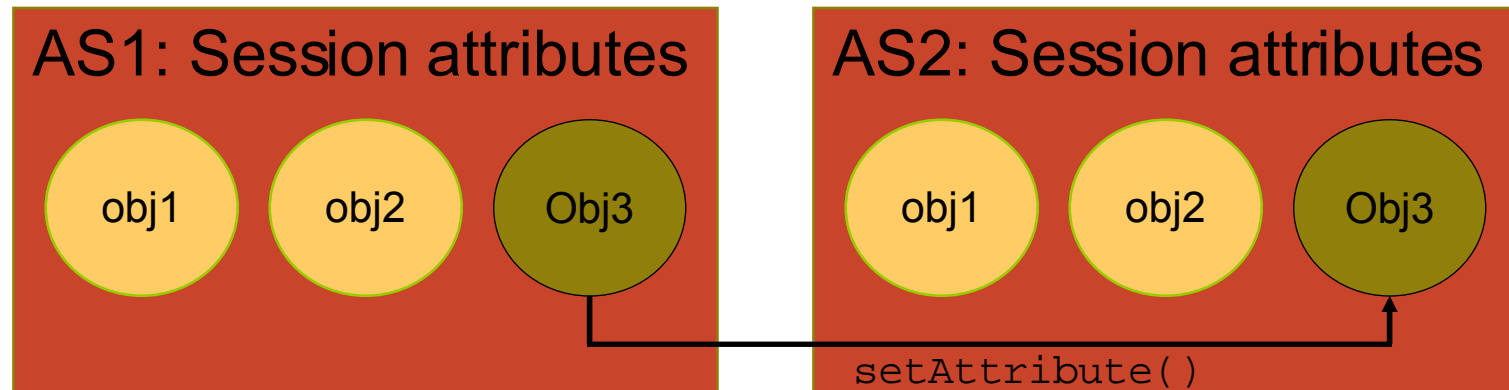
Cost of 16 byte change:

16 bytes



Propagazione tramite `setAttribute()`

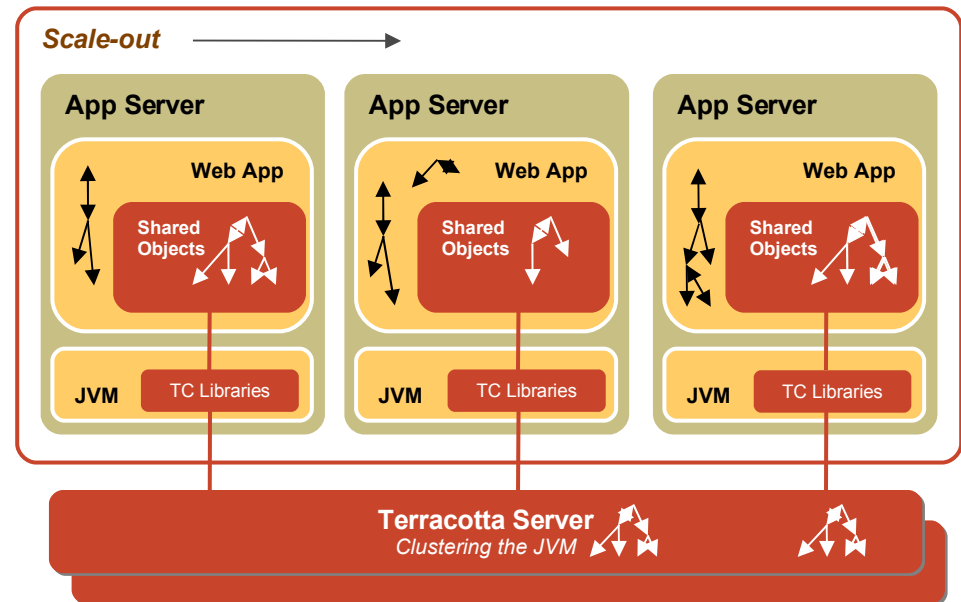
in molte soluzioni per replicare una modifica effettuata ad un oggetto, è necessario chiamare il metodo `setAttribute()`





Terracotta preserva l'identità di un oggetto

- Non serve chiedere la copia “fresca”
- NO `getAttribute()` / `setAttribute()`
- Non ci sono copie
- Un oggetto condiviso si comporta come un qualsiasi altro oggetto
- Ogni cambiamento effettuato ad un oggetto condiviso, è disponibile su tutti gli altri oggetti che possiedono la sua reference
- Dati due ref. 'a' e 'b' che referenziano 'c',
 - `a.equals(b)`
 - `a == b.`





Agenda

(1) Introduzione Terracotta DSO

(2) Clustering with Terracotta

(3) Architettura JEE

(4) Demo



Principi

- Terracotta DSO usa ASM per aggiungere istruzioni bytecode a runtime
- Terracotta Server è una applicazione pure java che usa Berkley DB come database di oggetti.
- Tramite un protocollo TCP le jvm comunicano con il TC Server per lo scambio e la sincronizzazione degli oggetti condivisi.



Configurazione

```
<tc:tc-config xmlns:tc="http://www.terracotta.org/config"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.terracotta.org/schema/terracotta-3.xsd">

  <servers>
    <server host="localhost">
      <data>/server-data</data>
      <logs>/server-logs</logs>
    </server>
  </servers>

  <clients>
    <logs>/client-logs</logs>
  </clients>

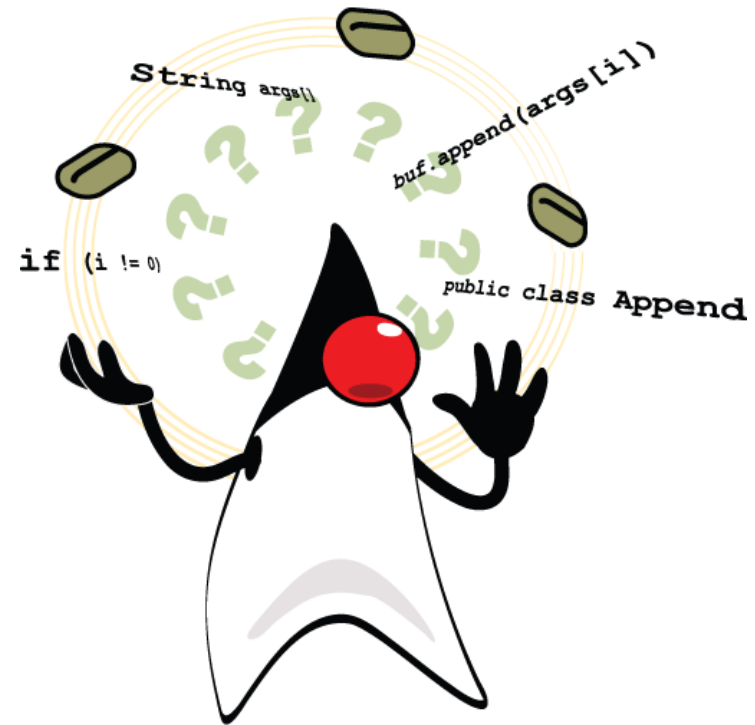
  <application>
    <dso />
    <spring />
  </application>

</tc:tc-config>
```



Configurazione DSO

```
<dso>  
  <roots/>  
  <transient-fields/>  
  <locks/>  
  <instrumented-classes/>  
  <distributed-methods/>  
  <web-applications/>  
</dso>
```





Hello World tutorial/HelloWorld.java

```
package tutorial;

import java.util.*;

public class HelloWorld {
    private List<String> hellos = new ArrayList<String>();

    public void sayHello() {
        synchronized(hellos) {
            hellos.add("Hello, World " + new Date());
            for (String hello : hellos) {
                System.out.println(hello);
            }
        }
    }

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





Hello World tc-config.xml

```
<tc:tc-config xmlns:tc="http://www.terracotta.org/config">
  <application>
    <dso>
      <roots>
        <root><field-name>tutorial.HelloWorld.hellos</field-name></root>
      </roots>
      <locks>
        <autolock>
          <method-expression>* tutorial.HelloWorld*.*(..)</method-expression>
          <lock-level>write</lock-level>
        </autolock>
      </locks>
      <instrumented-classes>
        <include><class-expression>tutorial..*</class-expression></include>
      </instrumented-classes>
    </dso>
  </application>
</tc:tc-config>
```



Hello World startup



- JAVA_OPTS

- -Dtc.config=<location of configuration information>
- -Dtc.install-root=<location Terracotta is installed>
- -Xbootclasspath/p:<dso-boot-jar-path>

dso-java.sh -Dtc.config=/path/tc-config.xml tutorial.HelloWorld



Heap Condiviso

Grafo degli oggetti

- Le root sono identificate da un FQN Field (it.ktech.package.Class.field)
- A partire da ogni “root” si formano dei grafi di oggetti condivisi.
- La prima istanza del root field non può mai essere sostituita
- Tutte le successive assegnazioni verranno ignorate
- Le jvm che proveranno ad assegnarlo riceveranno invece il valore dell'oggetto condiviso
- Questo rappresenta il maggiore Cambio di Semantica effettuato dalle librerie DSO
- Se un oggetto viene referenziato da un oggetto condiviso, esso e l'intero grafo di oggetti raggiungibili da esso diventeranno anch'essi condivisi
- Un oggetto clusterizzato ha assegnato un cluster-unique-id e rimarrà clusterizzato per tutto il suo ciclo di vita.
- Qualora un oggetto diviene irraggiungibile da ogni root e non ci sono istanze di questo in alcuna JVM allora questo diviene idoneo per la rimozione da parte del “cluster garbage collector” sel TC Server.



Cluster Injection and Bytecode Instrumentation

- Terracotta usa la tecnica di bytecode injection (ASM) come fanno tanti Aspect-Oriented Programming frameworks come AspectJ e AspectWerkz.
- Per gestire i cambiamenti dell'oggetto, sono state sovrascritte le istruzioni bytecode PUTFIELD e GETFIELD (AASTORE / AALOAD)
- Per coordinare i threads invece, sono state modificate le istruzioni bytecode MONITORENTER e MONITOREXIT per i blocchi sincronizzati come anche l'istruzione INVOKEVIRTUAL per i metodi Object.wait() e Object.notify() .



ASM: Field Operations

```
class Foo {
    private Bar bar;
    public Bar get() {
        if (isShared() && bar==null)
            bar = ManagerUtil.resolveReference(this, "Foo.bar");
        return bar;
    }

    public void set(Bar bar) {
        if (isShared())
            ManagerUtil.fieldChanged(this, "Foo.bar", bar);
        this.bar = bar;
    }
}
```





Clustered Locks

- I cambiamenti ad un oggetto fatti in un blocco “sincronizzato” formano una Terracotta transaction.
- La definizione di una Terracotta transaction è qualcosa di differente da una JTA transaction, questa infatti è molto più simile ad una transazione usata nel Java memory model.
- Per ottenere il lock su un'oggetto condiviso, il thread deve ottenere, oltre al lock della jvm, anche un cluster lock. Il thread sarà quindi bloccato finché non ottiene entrambi i lock, locale e del cluster.
- Tutti i cambiamenti effettuati tra in blocco sincronizzato sono salvati da Terracotta in una transazione locale e inviati al server.
- Viene garantito, prima che un altro thread di un'altra jvm riesca ad ottenere il lock, che tutti i cambiamenti effettuati sugli oggetti coinvolti sono stati replicati.
- Le transazioni, ovviamente possono contenere cambiamenti a qualsiasi oggetto, non solo all'oggetto di cui si detiene il lock.



Auto locks, named locks

Lock Types

- Auto locks
 - » convertono un blocco “synchronized” in un cluster-wide synchronized block.
- Named locks
 - » Rendono possibile l'esecuzione thread-safe di un qualsiasi metodo (anche se non synchronized) attraverso il meccanismo di sincronizzazione del cluster.
- Write Locks
 - » Multi-threaded synchronized lock.
- Synchronized-write Locks
 - Il server non consegna l'ACK di risposta finché la transazione non viene committata su disco (persistent store)
- Read Locks
 - » Permette multipli lettori, ma qualora arriva uno scrittore, vengono fermati tutti i lettori.
- Concurrent Locks
 - » Nessun cluster-wait / L'ultimo thread vince
 - » Ovviamente non è garantita la transazione





Distributed Methods

Esistono dei metodi che possono essere invocati in tutte le jvm del cluster contemporaneamente?

il metodo deve essere invocato su un oggetto condiviso.



Non sostituisce RMI

Comodo per le GUI, messaggio broadcast o refresh della view.



Slider tutorial/Slider.java

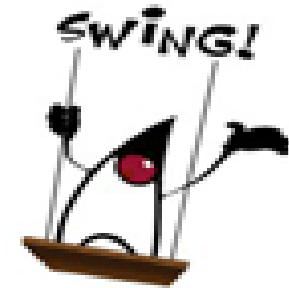
```
package tutorial;

public class Slider {

    private DefaultBoundedRangeModel rangeModel;

    public static void main(String[] args) {
        new Slider();
    }

    Slider() {
        JFrame frame = new JFrame("Slider Test");
        rangeModel = new DefaultBoundedRangeModel();
        frame.getContentPane().add(new JSlider(rangeModel));
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);
    }
}
```





Slider tc-config.xml

```
<dso>
  <roots>
    <root>
      <field-name>tutorial.Slider.rangeModel</field-name>
    </root>
  </roots>
  <instrumented-classes>
    <include>
      <class-expression>javax.swing.DefaultBoundedRangeModel</class-expression>
      <on-load>
        <execute>self.listenerList = new EventListenerList();</execute>
      </on-load>
      <honor-transient>>true</honor-transient>
    </include>
  </instrumented-classes>
  <additional-boot-jar-classes>
    <include>javax.swing.DefaultBoundedRangeModel</include>
    <include>javax.swing.event.EventListenerList</include>
  </additional-boot-jar-classes>
  <locks>
    <named-lock>
      <method-expression>
        void javax.swing.DefaultBoundedRangeModel.setRangeProperties(int, int, int, int, boolean)
      </method-expression>
      <lock-level>write</lock-level>
      <lock-name>setRangeProperties</lock-name>
    </named-lock>
  </locks>
  <distributed-methods>
    <method-expression>void javax.swing.DefaultBoundedRangeModel.fireStateChanged()</method-expression>
  </distributed-methods>
</dso>
```



Virtual Heap

- Un grafo condiviso potrebbe crescere al di sopra della dimensione massima raggiungibile da una singola JVM
- Terracotta permette un uso efficiente dell'heap locale
- tenendo solo gli oggetti di cui la JVM ha bisogno
- Raggiunta una percentuale terracotta libera l'HEAP locale di un 10% tenendosi dei piccoli “shadow object”.
- In ogni momento la JVM tramite lo “shadow object” può richiedere il vero “missing object”.
- La percentuale di memoria che la JVM può usare per l'HEAP condiviso è configurabile.



Supported integrations

- 'Supported' significa:
 - Basta aggiungere un modulo per clusterizzare la tua applicazione
 - No c'è bisogno di alcun setup
 - Con Terracotta DSO tutto continua a funzionare
 - Tecnicamente, Terracotta supporta tutte le integrazioni di tutto ciò che gira su una JVM





Agenda

(1) Introduzione Terracotta DSO

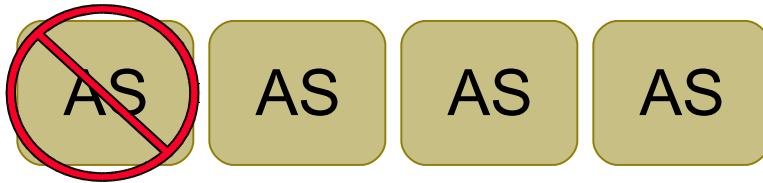
(2) Clustering with Terracotta

(3) Architettura JEE

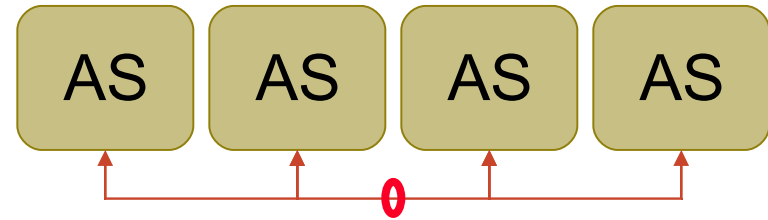
(4) Demo



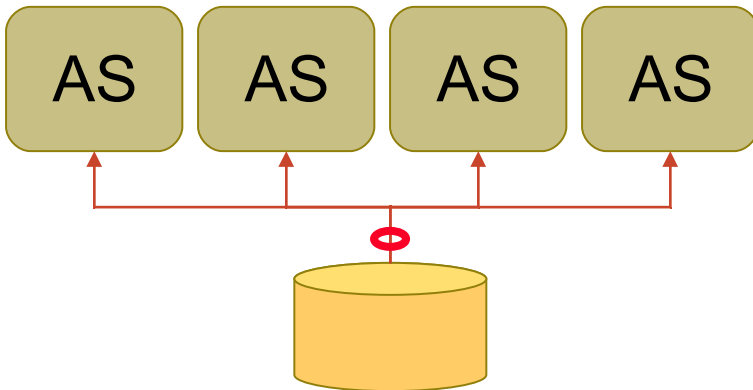
Typical Clustering Strategies



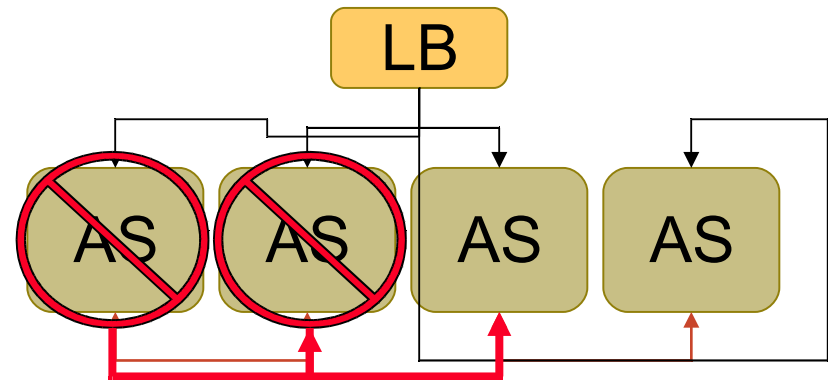
Single point of failure



Network bottleneck



SoR bottleneck

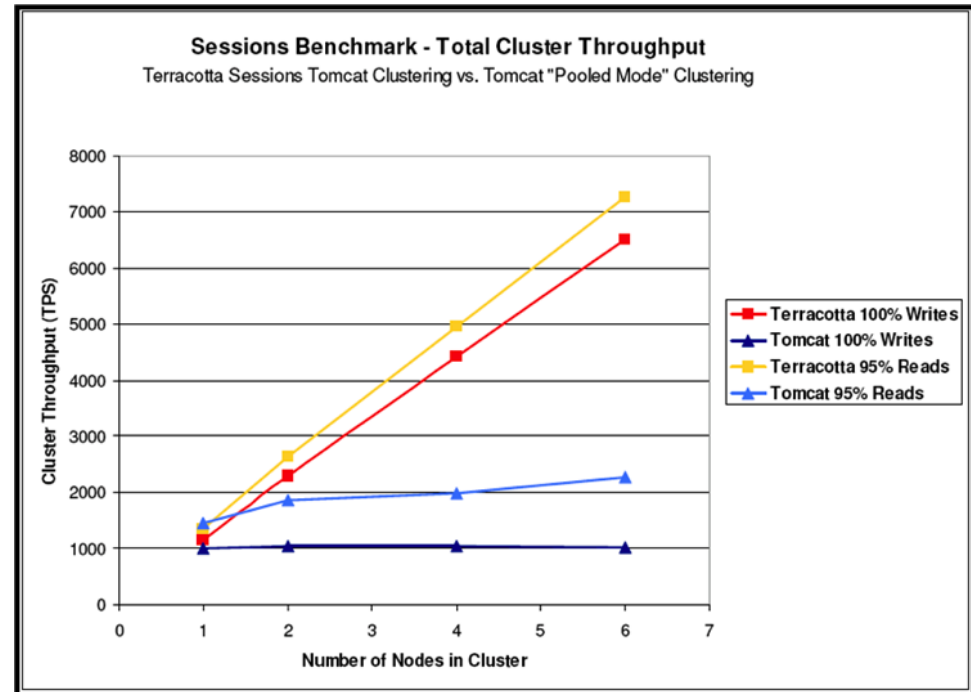


Cascading failures (Buddy system)



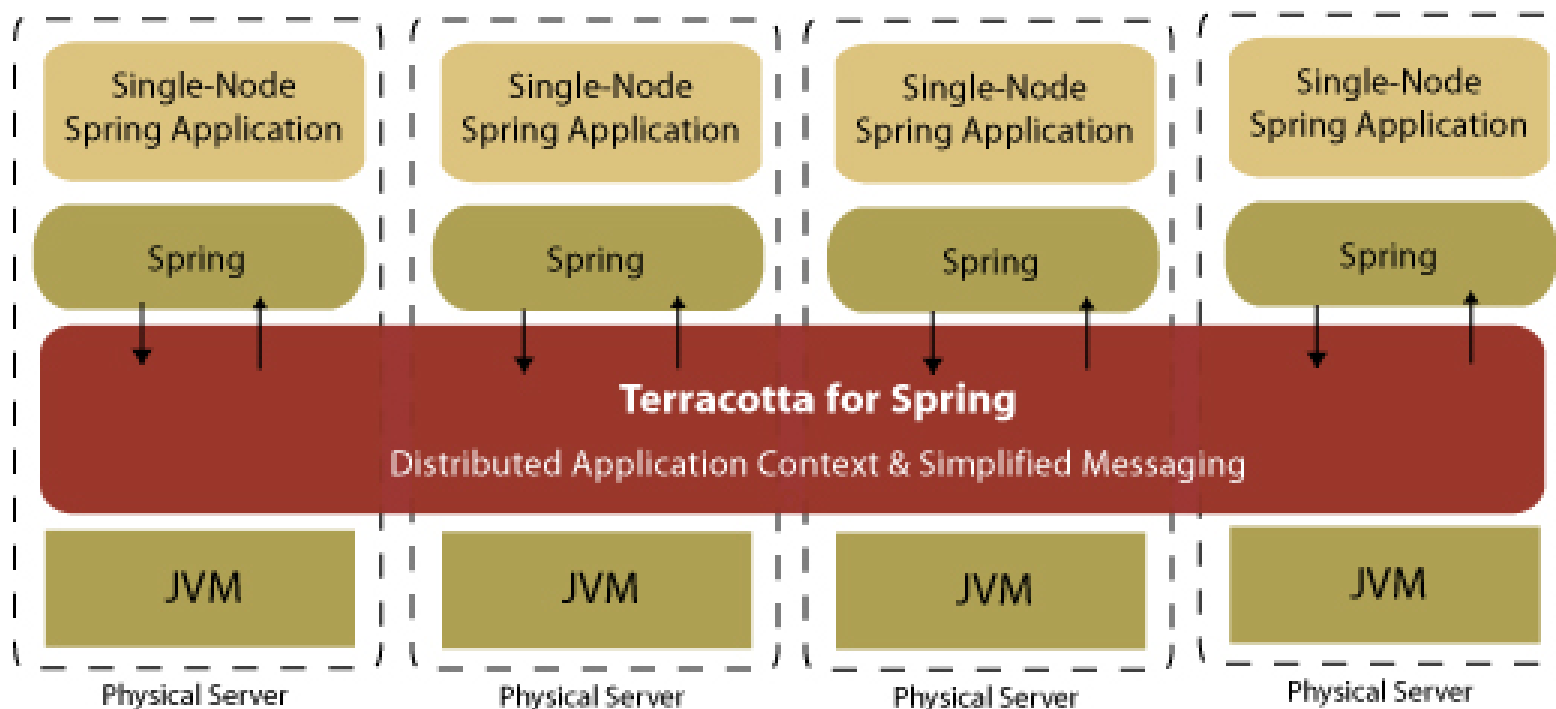
HTTP Session Clustering Benefits

- Terracotta Sessions gives you:
 - Near-Linear Scale
 - No java.io.Serializable
 - Large Sessions - MBs
 - Higher Throughput
- Supported Platforms:
 - Jetty,
 - JBoss 4.x,
 - Tomcat 5.0 & 5.5,
 - WebLogic 8.1, WebLogic 9.2,
 - WebSphere CE, Geronimo Alpha, WebSphere 6.1





Terracotta for Spring





Esempio Spring clustering

Terracotta config

```
<spring>
  <application name="tc-jmx">
    <application-contexts>
      <application-context>
        <paths>
          <path>*/applicationContext.xml</path>
        </paths>
        <beans>
          <bean name="clusteredCounter"/>
          <bean name="clusteredHistory"/>
        </beans>
      </application-context>
    </application-contexts>
  </application>
</spring>
```

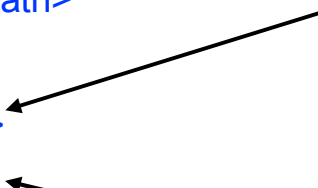
Spring config

```
<bean id="localCounter"
      class="demo.jmx.Counter"/>

<bean id="clusteredCounter"
      class="demo.jmx.Counter"/>

<bean id="localHistory"
      class="demo.jmx.HistoryQueue"/>

<bean id="clusteredHistory"
      class="demo.jmx.HistoryQueue"/>
```





Cluster Spring, Webflow & Events

- Si può realmente avere dei “singleton” beans (il pattern singleton può essere clusterizzato)
- Clusterizzare JMX MBean
- Distribuire gli Asynchronous Application Context Event
- Web-Flow support (including continuations)
 - Clustering of Web-Flow’s state machine
 - Transparent and high-performant **fail-over** for page flows
 - Potentially allow sharing a Web-Flow instance across an application, to be used by more then one user (when parallel tasks are required)
- Minimalistic config on component level



Cache distribuita

- Prendiamo una soluzione di cache
 - EHCACHE
 - JBoss TreeCache
 - OSCache
 - java Collections
- Clusterizziamola con terracotta
- Semplice, veloce e molto capiente...





Example Configuration Modules

EHCache clustering

```
<?xml version="1.0" encoding="UTF-8"?>
<tc:tc-config xmlns:tc="http://www.terracotta.org/config">
  <clients>
    <modules>
      <module name="clustered-ehcache-1.3" version="1.0.0"/>
    </modules>
  </clients>
  <application>
    <dso>
      <instrumented-classes>
        <include>
          <class-expression>tutorial.*.*</class-expression>
        </include>
      </instrumented-classes>
    </dso>
  </application>
</tc:tc-config>
```



Parlando di JEE

- EJB: ma ce ne è realmente bisogno?
- JMS: `Collections.SynchronizedList(list)`;
- JNDI: è un grafo di oggetti no?



Terracotta non sostituisce RMI



Architettura



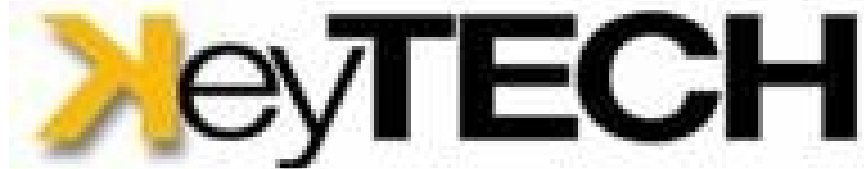


Agenda

- (1) Introduzione Terracotta DSO
- (2) Clustering with Terracotta
- (3) Architettura JEE
- (4) Demo



Questions?



KeyTECH

<http://www.k-tech.it>
s.federici@k-tech.it



Link utili

www.terracotta.org

www.springframework.org

www.javaportal.it

www.jugroma.it

www.aldaran.org/press/java/



Ringraziamenti

- Jonas Boner - Terracotta, Inc.
- K-Tech S.r.l. partner Terracotta
- Antonio Terreno - JugTorino
- Bruno Bossola - JugTorino
- Giorgio Vinci – Javaportal
- Mara Marzocchi – Javaportal

