The Simpler Way to Availability, Scalability and Performance

Presenters

Jamshaid Iqbal Janjua
Shahid M. Awan

KICS, UET, Lahore.
Agenda

- Background to Availability, Scalability and Performance
- What is Terracotta?
- How does it work?
- How you use it?
- Demonstration
- Questions & Answers
Terracotta

- www.terracotta.org
- Open Source since December 2006.
- Distributed under the Terracotta Public License that is based on the Mozilla Public License 1.1
- Sponsored by Terracotta Inc.
- Suitable for enterprise applications
Enterprise Application Characteristics

- Availability
- Scalability
- Performability
- Reliability
- Interoperability
- Accessibility
- Security
- Maintainability ….
How Do We Secure These Properties?

- **Availability**
  - The proportion of time that the system is in a functional condition.

- **Scalability**
  - The ability of a system to handle a bigger workload when more resources are made available to the system.

- **Performance**
  - How fast a system can execute a specific task - it is given.
Availability

We have two servers to get good availability but…

- If one of the servers goes down we will loose the HTTP-sessions in that server.
- One HTTP-session = One shopping basket = One Order = Money!
- The web service is not always available.
- The database is not always available.
Scalability

- With increased load on the application servers the load on the Database and Web-Service will increase.
- The database and web-service will not be able to handle the increased load.
- Scaling up the database can be expensive.

To abstract the database away using an ORM is a beautiful thought, but…

- The ORM invites to save all kind of states in the database.
  - Conversational
  - Data that is built up in pieces over time.
  - Throw-away-data
- We are Tormenting our database… an extra effort …
Performance

- Some database interaction will take long time and the system will be perceived as slow.
- The external web-service might slow down under heavy load.

*All these 3 properties Relates to…*

*… how we handle States!*
Java Enterprise System Setup Example
Traditional Solution

- Availability => Session Replication
- Performance/Scalability => Caching
Challenges with Session Replication

- How do we do the replication? There is no standard way.
  - We could serialize the session to the database at each HTTP request.
  - We could replicate over the network.

- Memory demands on the servers will increase in order to hold all the sessions.

- The web application has to be written for distribution.
  - Minimal session/setAttribute/invalidate etc.
Challenges with Caching

- The cache should be up-to-date.
  - Meaning is often depending on the application.

- Do we require coherence between application servers?
- If the cache is non-persistent it can only hold “mirrored” data.
- Consumes memory.
- Complicated to implement yourself.
- Might be complicated to configure and tune.
What is Terracotta?

Terracotta offers:

- **A Coherent, Distributed and Persistent JVM heap.**
  - Java objects created on the heap are available in all JVMs.
  - Objects survive a JVM restart.
  - Object identity is preserved between JVMs (no copies!)

- The heap follows the memory and thread model of Java.
  - Java objects have coherent state between JVMs.
  - Threads in different JVMs interact just like threads in the same JVM.

- Requires no specific Java APIs.
- Integrates with other Java frameworks
Take Your Applications from this...

Clustered App Servers Are Expensive
…To This

- Business Logic
  - Frameworks
  - Frameworks
  - App Server
  - Terracotta DSO
    - JVM
  - JVM

- Business Logic
  - Frameworks
  - Frameworks
  - App Server
  - Management Console
    - Terracotta Server
Features

- JVM Coordination
  - Distributed Synchronized
  - Distributed wait()/notify()
  - Consistency Locking
  - Distributed Method Invocations

- Large Virtual Heaps
  - As large as available disk
  - Dynamic paging

- Management
  - Runtime visibility
  - Data introspection
  - Cluster monitoring
Network Attached Memory (NAM)
Terracotta

- Terracotta Server (100% Java)
  - Can be configured to persist all states on disk.
  - Handles Distributed Object, Memory and Locks.

- Terracotta Clients
  - Is loaded into the JVM at boot time.
  - Instruments specified Java classes with cluster behavior.
  - Automatically connects to the Terracotta server at boot time.
  - Can be started with specific wrapper script (dso-java.sh/.bat).
Terracotta – Scale Out Approach
How Does it Work?

JVM

TC-Grid

TC-Server

Disk

TC-Server

Disk
How Does it Work?
DSO / ROOTs

- **Distributed Shared Objects (DSO)**
  - When you make changes to a clustered object in your code, Terracotta keeps track of those changes and sends them to all Terracotta server instances.
  - Server instances, in turn, makes sure those changes are visible to all the other JVMs in the cluster as necessary.
  - This way, clustered objects are always up-to-date whenever your code accesses them, just as they are in a single JVM.

- **Root**
  - It is a core concept that enables Terracotta to identify which objects should be clustered, and which should not.
How Do You Do It?

- Define a DSO
- Define “root” objects in your Java classes
- Instrument the classes that are to be clustered.
- All objects reachable from a “root” are clustered.
Terracotta Configuration

```xml
<dsos>
  <instrumented-classes>
    <include>
      <class-expression>HelloWorld</class-expression>
    </include>
  </instrumented-classes>
  <roots>
    <root>
      <field-name>HelloWorld.counter</field-name>
      <root-name>counter</root-name>
    </root>
  </roots>
  <locks>
    <autolock auto-synchronized="false">
      <method-expression>* HelloWorld.main(..)</method-expression>
      <lock-level>write</lock-level>
    </autolock>
  </locks>
</dsos>
```
Conclusions & Reflections

- Terracotta is no database!

- We can build real Object Oriented Domain Models without constraints.

- Terracotta makes itself transparent by clustering the JVMs.

- Terracotta is built to make the world simpler (for developer and operator).

- Terracotta has been open sourced for two years.
Open Source Integrations

- Tomcat
- Geronimo
- Jetty
- Struts
- Wicket
- Rife
- Webflow

- Hibernate
- iBatis
- Harmony(!!!)
- Others coming soon…
Sources

- Learn Terracotta
  - [http://www.terracotta.org](http://www.terracotta.org)

- Download Terracotta today:
  - [http://www.terracotta.org/dl/](http://www.terracotta.org/dl/)

- Articles:

- Documentation and blogs:
  - [http://www.terracotta.org/documentation/](http://www.terracotta.org/documentation/)
  - [http://www.callistaenterprize.se](http://www.callistaenterprize.se)
  - [http://blog.terracottatech.com/](http://blog.terracottatech.com/)
DEMONSTRATION
Thanks!
Network Attached Memory (NAM)
Use cases for Terracotta

- Distributed caching (HashMap, EHCache)
- Session Replication (Out-of-the-box).
- Offload the database.
  - Handle objects that does not have to be stored in the database.
- Simple messaging (LinkedBlockingQueue).
- Workload partitioning.