

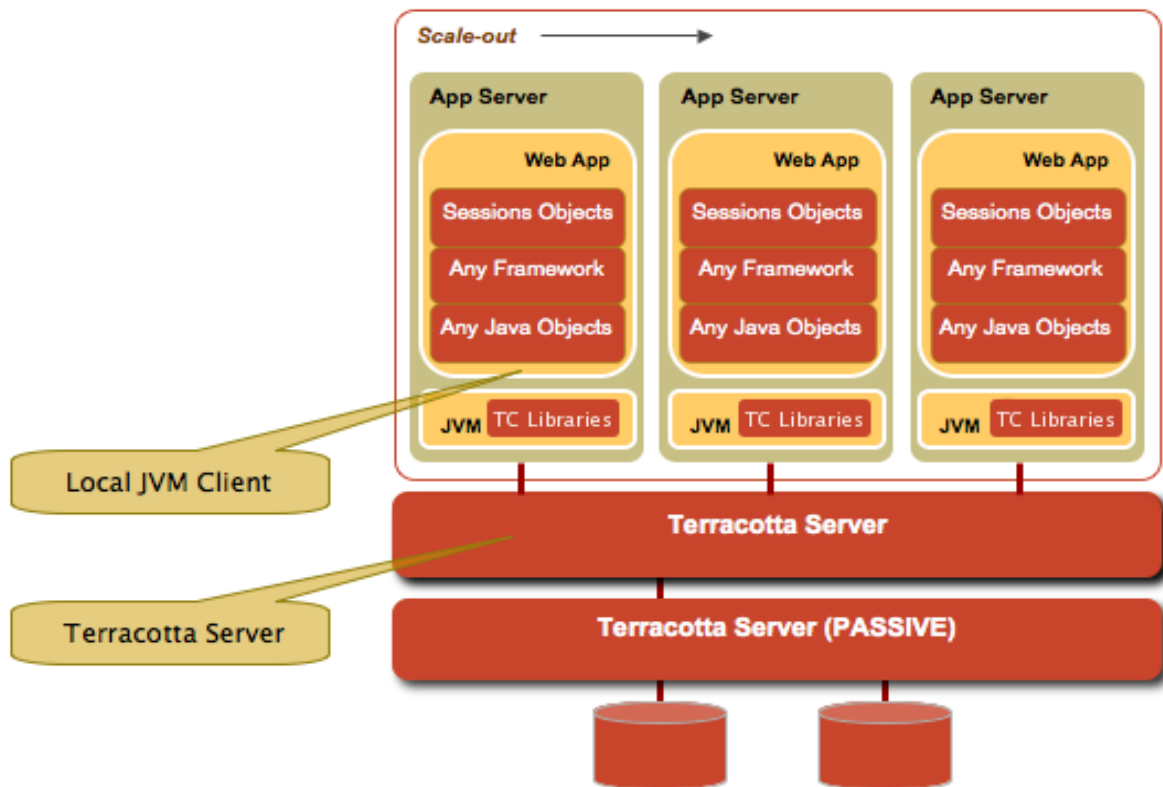
Présentation de TerraCotta

Les solutions classiques de clustering fonctionnent pour les applications JEE soit en s'appuyant sur JMS pour synchroniser différentes instances d'une application tournant sur différentes JVM ou sur des solutions de cache distribué du type [JCache \(JSR 107\)](#). De façon générale, ces solutions passent par la sérialisation.

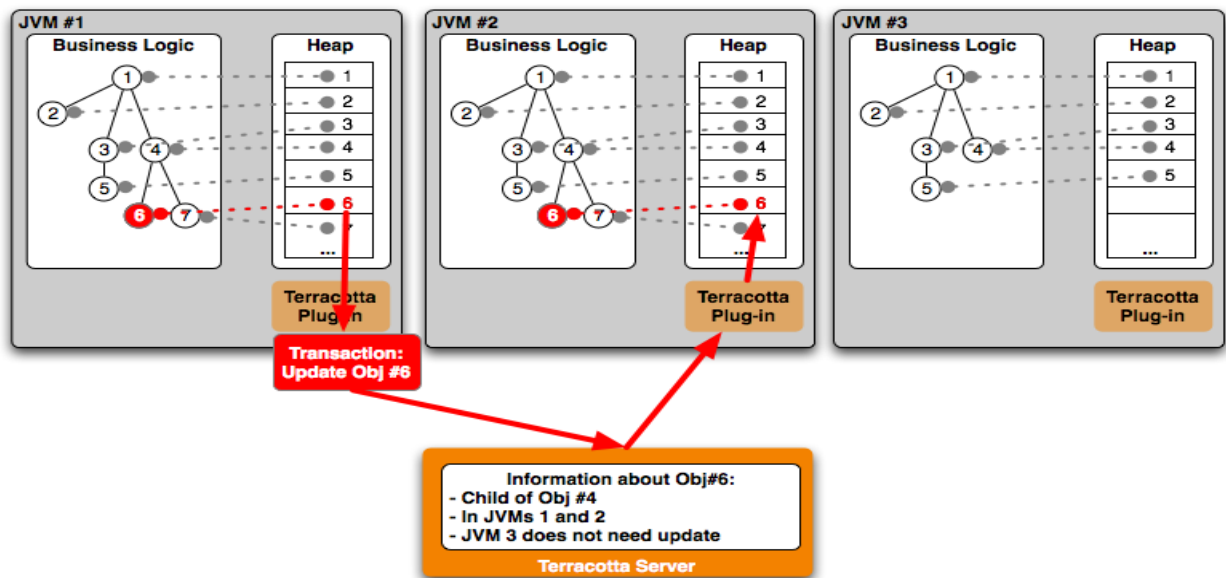
L'architecture de TerraCotta est pensée pour minimiser le dialogue réseaux, garantir qu'il n'y est pas de perte d'objet, fournir un control maximum, le tout en assurant de pouvoir une haute disponibilité et d'éviter les SPOF (Single Point Of Failure).

L'architecture présente les caractéristiques suivantes :

- **Client Nodes** Chaque Noeud client correspond a un noeud dans le cluster. Les noeuds clients s'exécute dans une JVM standard. TerraCotta est installer dans la JVM par le biais des librairies TerraCotta qui sont chargé au démarrage de la JVM.
- **Terracotta Server Cluster** - Terracotta Server Cluster fournit l'intelligence pour la mise en cluster. Chaque serveur au sein du cluster est un processus 100% JAVA. L'implementation actuel de TerraCotta opère dans le mode Actif/Passif, avec un serveur actif et un ou plusieurs serveur passif.
- **Storage** Le serveur TerraCotta utilise le stockage sur disque pour différente chose :
 - **Virtual Heap storage** Si la mémoire pour les objet est pleine alors les objets sont paginé sur disque
 - **Lock Arbiter** Pour s'assurer de ne pas rencontrer le probleme classique du «split-brain », TerraCotta s'appuis sur l'infrastructure disque pour fournit un verrou.
 - **Shared Storage** pour passer les objets de l'état actif à passif, les objets sont persisté sur disque.



Terracotta propose une approche différente, le clustering est proposé au niveau JVM directement. TerraCotta est capable de gérer un cluster de manière transparente, TerraCotta simplifie la tâche d'écriture d'application en cluster car il n'impose pas de model programmatique.



Clustering des applications spring

TerraCotta s'intègre de manière transparente dans les applications construite sur le framework Spring.

Pour ce faire TerraCotta change la sémantique des **ApplicationContext Events** Spring en les transformant en **Distributed Reliable Events**, il devient alors possible de diffuser de façon transparente les modifications concernant un POJO à l'ensemble des JVM où ce POJO est répliqué.

Vous pouvez développer comme si vous développez pour un seul serveur et ensuite vous avez juste à configurer TerraCotta.

TerraCotta ne requiert aucun changement dans le code Spring existant. L'application est instrumentée à l'exécution par le biais d'un fichier de configuration xml.

Vous pouvez déclarer quels seront les applicationContext qui seront clusterisés