**HAZELCAST DISTRIBUTED OBJECTS:**

1. Creati staticamente da file configurazione
2. E’ possibile crearli dinamicamente. (IdGenerator)  
   In tal caso per garantire un id univoco su tutto il cluster e’ possibile servirsi del IdGenerator.  
   Normalmente in tal caso e’ bene far precedere il nome con cui crearlo da un tipo, esempio  
   Customers-<generatedId>
3. Reload: (singleton)  
   Possono essere richiesti ad Hazelcast tramite id, ed ogni chiamata restituira’ lo stesso (singleton Spring)
4. Possono essere rimossi tramite api (destroy())   
   In tal caso le successive chiamate per Id o i riferimenti esistenti (vars) ne creeranno uno nuovo con lo stesso id, senza dare eccezione.
5. E’ possibile assegnarli ad una partizione da codice o da config.

**DISTRIBUTED EXECUTOR SERVICE:**However the problem in a distributed system is that the default implementation ofthe Executor, the ThreadPoolExecutor, is designed to run within a single JVM. In a distributed system you want that

**a task submitted in one JVM, can be processed in another.**

Oggetti che possono essere inviati al Executor.submit() devono essere Serializable.

**Scaling up:**(Scaling verticale. Stessa macchina, upgrade hardware)  
Aumentare dimensioni thread pool. Verificare quanto e’ sostenibile dal sistema. Troppo/TroppoPochi portano entrambi ad un degrado delle prestazioni.

**Scaling out:**Scale orizzontale. Aggiungere macchine al cluster.  
Il thread pool e’ distribuito (ciascuna istanza mette a disposizione i thread configurati) viene cordinato da HZ per cui ogni thread viene eseguito su un membro HZ del cluster (eventualmente su un’altra macchina).

**Routing:**Scegliere quali macchine condividono un ExecutorService distribuito.

Policies:

1. Qualsiasi instanza HZ nel cluster (default)
2. Scegliere programmaticamente una o un gruppo di istanze nel cluster:

for ( Member member : hz. getCluster (). getMembers ())

1. Distribuire un calcolo su N macchine:  
   Es. Sommare tutti i valori contenuti in una Map distribuita.  
   Ogni istanza si prende cura della sua parte locale in un Callable.
2. Scegliere per key di partizione (ip)   
   Comodo per eseguire thread sulla stessa macchina dove si trovano i dati di cui necessita.  
     
   Nel task:

IMap.localKeySet() (sceglie le chiavi salvate sulla partizione locale)  
  
Submit del thread:  
executor.executeOnKeyOwner(new SomeTask(keyInDistrMap),keyInDistrMap );  
  
Possibilita di implementare interfaccia **HazelcastPartitionAware**

Da ricordare:

Unlike Hazelcast 2.x, in Hazelcast 3.0 it isn’t possible to cancel a future.

One possible solution is to let the task periodically check if a certain key in a distributed map exists. A

task can then be cancelled by writing some value for that key. You need to take care removing keys to

prevent this map from growing; you can do this by using the time to live setting.

**ExecutionCallBack:**Interfaccia di callback che viene invocata quando il Future restituisce il risultato.