Cosa e’ Hazelcast:

What is Hazelcast?

When you write applications for the JVM professionally, it is likely that you are

going to write server-side applications. Although Java has support for writing

desktop applications, the server-side is where Java really shines.

Today, in the era of cloud computing, it becomes more and **more important that**

**server-side systems are:**

1. Scalable: just add and remove machines to match required capacity

2. Highly available: if one or more machines has failed, the system should continue

as if nothing happened.

3. Highly performant: performance should be fast, and cost effective

**Hazelcast is an In-Memory Data Gird. It is:**

1**. Highly available:** It does not lose data after a JVM crash. This is done by

automatically replicating partition data to other cluster members. In case

of a member going down, the system will automatically failover by restoring

the backup. Another important design feature of Hazelcast is that there is

no master member that can form a single point of failure; each member has

equal responsibilities.

2. **Lightning-fast**: Each Hazelcast member can do thousands of operations per  
Hazelcast on its own is elastic, but **not automatically elastic**: it will not itself

automatically spawn additional JVMs to become members in the cluster when the

load exceeds a certain upper threshold. Also it will not shutdown JVMs when the

load drops below a specific threshold. This can be achieved however by adding

glue code between Hazelcast and your cloud environment.

**Considerazioni da architetto**

One of the things I like most about Hazelcast is that it is unobtrusive; as a developer/

architect you are in control of how much Hazelcast you get in your system.

**Non invasivo:**

You are not forced to mutilate objects so they can be distributed,   
use specific application servers,   
complex APIs, or install software;  
  
 just add the hazelcast.jar to your classpath and you are done.

This freedom combined with very well thought out APIs makes Hazelcast a joy

to use. In many case, you simply use interfaces from java.util.concurrent, such as

Executor, BlockingQueue or Map. You can write a highly available, scalable and

high-performing system, in little time with simple and elegant code.