CS290B: Java-centric cluster and concurrent computing
Marianne Melhoos, Johannes Lier, John-Olav Storvold, Julie Rekdal
UCSB, Spring 2015

Summary: How to build a ComputeFarm

Intro

Programs runs faster by dividing them in smaller parts and allowing these parts to run simultaneously on multiple processors. This paper is about ComputeFarm which is an open source Java Framework for making programs with the qualities mentioned above.

The Replicated-Worker Pattern

The *Replicated-Worker pattern* or *Master-Worker pattern* is a pattern where master process creates other tasks that needs to be done. The master process is the client of the ComputeFarm. The worker processes runs these tasks and returns the result to the master process. The masters and workers communicate through a *space*. A space holds the task objects and their results.

A worker goes through some steps in their lifecycle. These are:

- Wait for an available task from the space.
- Execute the task when available.
- Put the result from the task back into the space.
- Go back to step 1.

The problem the client is trying to solve is called a *Job*. The tasks done by the worker processes is "under the hood" as seen by the client. The steps taken for solving the problem seen from the client's point of view is these:

- The client creates a job and divides it into tasks.
- Each task is placed in the space.
- Each task is done by one of the workers.
- The results are placed in the space and picked up by the job which combines them into the final result.

Code Mobility

The ComputeFarm workers use dynamic code downloading i.e. Java RMI, to be able to be generic, which means that remote JVMs can access the Java classes by providing a URL where the classes can be downloaded from. To avoid configuration errors ComputeFarm provides a an embeddable server, ClassServer, that allows you to run the server in the same JVM as the client.

Fault Tolerance

The transition from a single process to multiple distributed processes changes what kind of failure the program can show. The distributed computing model does not try to solve them, but provides mechanisms for how the programmer can deal with them.

To make a program robust you need to handle three types of exceptions: (1) ComputeSpaceException, (2) CannotWriteTaskException and CannotTakeResultException and (3) CancelledException.