Roman Kazarin
Michael Nekrasov
CS290B – Peter Cappello

Paper Summary: How to Build a Compute Farm

A compute farm is an extension of the Master – Worker Paradigm. In a compute farm there is a "Space" that mediates between a "Master" that assigns tasks and "Workers" that perform tasks. Each worker waits for a task from the space, executes it, then returns the result back to the space. There could be many workers that run on separate machines. The machines running the workers do not have to be of uniform hardware and can run at different rates.

The client does not have to worry about the workers or their implementation, and interacts solely with the space. They focus on the bigger problem they are trying to solve referred to by the paper as a "Job". The clients define these jobs and specify how they are subdivided into sub tasks. They hand these tasks over to the space which in turn sends them to the workers and returns the results which are combined back into an overall result.

The process of breaking a job into tasks and executing the tasks on workers can be done concurrently. Workers can start processing tasks, while the job is breaking them up. The paper encourages the Job to be run by a JobRunner that can abstract the process of how the job is run (through spaces or on a single JVM).

Spaces and workers are contacted through the Java RMI. Code does not have to be preloaded and can be downloaded dynamically from a codebase. The space, client, and workers, can be located on the same machine or distributed through the network.

The system should be fault tolerant, so that workers, spaces, and client can be started in any order. If workers crash the space should be able to recover and reassign the tasks to a new worker. Tasks that no longer need to be run should be cleaned up automatically. As long as the space is written in a way that does not leave it in possible indeterminate states. In the event of unexpected failure, clients can reissue tasks.

The usefulness of this approach depends on how parallizable the problem is. While the article gives a multiplication problem that does not provide a benefit to rescaling. Problems like the Traveling salesman problem and the Mandelbrot set can be easily parallized for fast computation across multiple machines.