Mike MacFerrin – CS 7900 – PhD Intro
Assignment 11 – Envisioned PhD Thesis Abstract
November 27, 2006

Flexible Distributed Computing using Ability-Specific Nodes

One of the greatest advances in modern computing was the invention of *programmable, general-purpose* computers. With the unveiling of the ENIAC in 1946 (the first working electronic programmable computer), scientists & corporations gained the ability to easily solve huge varieties of tasks using common hardware and a common programming interface. Modern computing was on its way.

Six decades later, programmable machines are still quite a hit. Today, the grand promise of future technology leans primarily on the shoulders of nano-technology, intelligent systems, and parallel computing. In order to deliver upon these promises while keeping the technology affordable and flexible, similar ideas must be adapted to work with distributed systems of interchangeable components.

The premise of my (envisioned and possibly entirely hypothetical) thesis is to devise & implement a simple interface that allows small (presumably cheap) nodes, each of which performs a specific computational task, to work in conjunction with each other to solve large and arbitrarily complex computational problems. Such a system is nothing novel in itself, but the interface will be designed to be easily scalable to a large variety of tasks, both computational and/or physical. In theory, the capabilities of the system are bounded only by the number of nodes (robots) you have available and the collective number of primitive tasks these robots are capable of performing. Given a common interface, such nodes could be interchanged as needed, and reprogrammed on the fly, to perform a large variety of complicated tasks.

In theory, such a system, using a common interface, could conceivably program small armies of nodes/robots to heat your house, program your Tivo, cook you breakfast in the morning, clean toxic environmental spills, execute search-and-rescue operations, and perform surgeries. These far-reaching applications are (of course) well beyond the scope of a single CS-Major's thesis. For now, I'll just concentrate on getting them to compute simple things. Using a set of (real or simulated) nodes capable of performing computational primitives (such as multiplying an array of numbers, e.g.), my system should be able to take any basic program written in a common language (such as BASIC or C), parse it into a grid of primitive components (with an accompanying grid of dependencies between these components), and "farm" these components out to the available nodes, which will perform the required operations, returning results either to the original computer or to each other in the network. The success of the task will be easily quantifiable: can my system of nodes perform the same task that a compiled program (executed on a single multi-purpose machine) was designed to do? Can it perform its task on a computationally tractable timescale? Can it be scaled to perform the same task given an arbitrarily large (or small) number of primitive nodes available?

That last question is the deal-breaker, and its answer will determine whether the idea is bound for future development or simply shelved as an expensive way to perform otherwise simple tasks.