Unit 8: Parallel Processing

October 13, 2015

1 Computer architecture

Computers now come with multiple processors for doing computation. Basically, physical constraints have made it harder to keep increasing the speed of individual processors, so the chip industry is now putting multiple processing units in a given computer and trying/hoping to rely on implementing computations in a way that takes advantage of the multiple processors.

Everyday personal computers often have more than one processor (more than one chip) and on a given processor, often have more than one core (multi-core). A multi-core processor has multiple processors on a single computer chip. On personal computers, all the processors and cores share the same memory.

Supercomputers and computer clusters generally have tens, hundreds, or thousands of 'nodes', linked by a fast local network. Each node is essentially a computer with its own processor(s) and memory. Memory is local to each node (distributed memory). One basic principle is that communication between a processor and its memory is much faster than communication between processors with different memory. An example of a modern supercomputer is the Edison supercomputer at Lawrence Berkeley National Lab, which has 5576 nodes, each with two processors and each processor with 12 cores, giving 133,824 total processing cores. Each node has 64 Gb of memory for a total of 357 Tb.

There is little practical distinction between multi-processor and multi-core situations. The main issue is whether processes share memory or not. In general, I won't distinguish between cores and processors. We'll just focus on the number of cores on given personal computer or a given node in a cluster.

2 Parallel processing with shared memory

The core content on parallel processing that we'll cover in the course is in the tutorial on basic parallel processing in R, Python, Matlab and C: see https://github.com/berkeley-scf/tutorial-parallel-basics.

You can ignore the material in the tutorial that is related to Python, Matlab, and C for the purpose of this course.

For now, you can also ignore the section on random number generation as we'll be talking about that in the simulation unit in a couple weeks.

3 Other approaches to parallel processing

Other approaches to parallel processing include:

- GPUs
- Spark and Hadoop
- Tools in R, Python and Matlab that allow for parallelizing loops across multiple machines
- Distributed computing using MPI

The tutorial on basic parallel processing has some links to information on these topics. Unfortunately, except for Spark, we won't have time in this course to go into these topics.

4 Using an Amazon Web Services EC2 virtual machine

We have a small grant from Amazon to use their EC2 service for class work. Mostly this will be used for Spark, but you can also use it for parallel processing on a single machine, either to test out the code in the parallel processing tutorial or to work on problem set questions. That said, for the most part you can also just use the BCE VM on your own machine provided your machine has enough cores, memory, and disk space that can be shared with the VM.

For details on starting up an Amazon EC2 instance that uses the BCE VM, please se the *AWS-setup.txt* and *startEC2virtualMachine.txt* files in the *howtos* directory of the class repository. Please stop or terminate your instance as soon as you are done using it, so we don't run out of credits, and please don't run an EC2 instance for more than a couple of hours.