Parallel Processing
CS 350: Computer Organization & Assembler Language Programming
Lecture 20, Wed 4/10

We used the textbook's slides for other sections of Chapter 6 for Lecture 19 (Mon 4/8) and the rest of Lecture 20.

Warehouse Scale Computers (WSC)
- Extremely large-scale settings - one building holding 100,000 servers
  - Not just one huge cluster
  - Try to operate to some extent as one giant computer
  - Large reliance on independent processes and independent datasets
- WSC's differ from regular large servers
  - Parallelism is ample and easily available.
  - Operational costs are important
  - Both benefits but costs are magnified by the extremely large scale
- Programming style is very different
  - Book talks about MapReduce and Hadoop
  - Run program on thousands of servers simultaneously
    - Creating data and collecting it
    - How do problems and programs get thought about when you want this massive amount of parallelism?
- Requires whole other level of
  - Power distribution, cooling, processor monitoring/operations
  - Some modern systems are built in the Arctic to take advantage of cooler weather / cooler river/lake water
- Actual use is for systems that involve a lot of independent work
  - Emulate one huge web server
- Start thinking about possible computations as a mass instead of as a collection of individual processors

Activity Questions for Lectures 19 and 20 (Parallel Processing)
1. Is a parallel processing program an example of process-level parallelism? If not, why?
2. What are the main difficulties involved in writing software for parallel programs?
3. What is Amdahl's law, and what does it tell us?
4. What is a strongly-scaling problem? A weakly-scaling problem? How do they differ if you, say, double the number of processors?
5. Define the terms SISD, SIMD, MIMD, MISD, and SPMD. Which one is rare-to-nonexistent? Which term is not like the others?
6. Briefly, how are vector processors different from scalar processors?
7. How does a multi-core computer differ from a loosely-coupled cluster?
8. Why, with warehouse-scale computing, is the availability and cost of parallelism not an issue?

**Solutions to Activity Questions**

1. They are different: A parallel processing program is a single program that runs on multiple processors. In process-level parallelism, multiple programs are run concurrently on one or more processors.
2. Partitioning problems, coordinating activity across processors, and the communications overhead involved.
3. Amdahl's law says that the sequential (non-parallelizable) part of a program limits the speedup generated by improving the parallelizable parts. As the time taken by the parallelizable part of the program approaches zero, the overall time of the program approaches the time for the sequential part of the program.
4. In strong scaling, increasing the number of processors does not change the size of the problem being solved, so doubling the number of processors might halve the running time of the program. In weak scaling, increasing the number of processors increases the problem size proportionally, so doubling the number of processors doubles the problem size but leaves the running time unchanged.
5. SISD, SIMD, MIMD, and MISD give the four ways of combining Single/Multiple Instruction streams with Single/Multiple Data streams. SPMD is different because it discusses programs: Single Program, Multiple Data streams (a parallel program running on a MIMD machine). There are no examples of MISD involving one processor. MISD with the instruction streams on separate processors can be used for redundant processing such as "I tell you three times".
6. Vector processors have highly-pipelined arithmetic/function calculations; they can make data-parallel programming easier, and they can make it clearer when loops cause dependencies.
7. Multi-core and cluster computing both involve multiple CPUs, but in a multi-core machine, the cores share memory (and in practice, the OS). In a loosely-coupled cluster, the machines are independent and each has its own private memory and OS. Multiple cores communicate through shared memory or busses; clusters connect using I/O.
8. Warehouse-scale computers run applications that assume large numbers of independent datasets, which leads to processes that rarely require synchronization and therefore low communication cost. Datasets can be individually processed, which makes scheduling very easy, which reduces system overhead. The result is that parallelism is easily available and at low cost.