

Lecture 1: January 23

*Lecturer: Horst Simon**Scribes: Man-Kit Leung*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

Today's class will be an overview of many things. The paradigm has shifted from making powerful computers parallel to making all computers parallel. The application space for parallel programs has also changed from scientific, super computing applications to basically all applications. The material in this class is becoming more applicable to all applications since all computers are becoming parallel.

We started off with some interesting quotes (tunnel vision) from experts in the field:

- "I think there is a world market for maybe five computers."
- "There is no reason for any individual to have a computer in their home"
- "640K [of memory] ought to be enough for anybody."
- "On several recent occasions, I have been asked whether parallel computing will soon be relegated to the trash heap reserved for promising technologies that never quite make it."

Thinking machine cooperation was one of the pioneers in parallel computing. Though the company has proven to be a disappointment of early 1990s, things are coming back. People have taken interests in looking at parallel computation again for several reasons. The "Cloud computing" article in Business Week (Google) wrote about how computing power are owned exclusively by big players (i.e. Amazon, Google, MS, and etc.). One can see the two extremes on how parallel computing technology will develop, whether it will be provided as services or as off-the-shelf commodities for everyone.

Technology Trends: Microprocessor Capacity

- Moore's Law
- double transistors/Chip every 1.5 years
- microprocessors have become smaller, denser, more powerful
- exponential growth
- significant increase of processor speed from shrinking the size of transistor

Impact of Device Shrinkage

- increase of power

- increase of heat generated
- increasing cost

Limiting forces. In order to shrink more and more device, you have to spend more money. Cost of semiconductor factories is also increasing exponentially. Consequently, only a few big players are left in these fab lines. Increasing the density of the chip also increase the heat discipation. There is a heat limit. Therefore, clock rate can no longer be pushed up because of the heat limit. Rather than pushing for faster clock rate, processor designers are looking into parallelism. However, number of transistor per chip is still going up. The question is what can we do with these extra hardware?

Why parallelism (2008)? All major processor vendors are producing multicore chips. Every machine will soon be a parallel machine. All programmers will be parallel programmers. This is a big change for everybody. There is a lot transition going on. This mean a lot of re-evaluation and re-investigation in face of parallelism. Computing will be changed dramatically. (i.e. GPU, etc.) What will be the killer apps for multicore machine. How should the chips be designed, and what the programming model will be?

Is multicore the correct response? Maybe, maybe not. At least we are not quite sure yet.

- Kurt Keutzer: “This shift toward increasing parallelism is not a triumphant stride forward based on breakthroughs in novel software and architectures for parallelism; instead, this plunge into parallelism is actually a retreat from even greater challenges that thwart efficient silicon implementation of traditional uniprocessor architectures.”
- David Patterson: “Industry has already thrown the hail-mary pass. . . But nobody is running yet.”

There are more exotic solutions, like FPGA, GPU, DF, tiled processor architectures, and Cell. We are basically in the middle of the turmoil of many earlier ideas that have been forgo.

There are seven major questions for parallelism.

- Applications:
 1. What are the apps?
 2. What are kernels of apps?
- Hardware:
 3. What are the HW building blocks?
 4. How to connect them?
- Programming Model / Systems Software:
 5. How to describe apps and kernels?
 6. How to program the HW?

- Evaluation:

7. How to measure success?

This class will focus on the application and the kernels. This research is application-driven. We initially started the concept of “Dwarves” to classify different applications into 7 categories. Now, there are more newly added categories, and are now called “Motifs” (13 of them). The concept of dwarves is very useful because we can talk about how to transform these dwarves into parallel applications.

MapReduce used to be called embarrassingly parallel because its concept was very simple. The basic concept was to partition a given task into independent sub-components, and distribute them onto clusters of machines for execution. However, MapReduce is more than embarrassingly parallel because it has the whole software framework to parallelize a task efficiently.

We are currently in around the Tera, Peta age. You can go to the website www.top500.org to look up the current list of the fastest machines. Most of these machines are built and sited for government agencies. The performance and capability of systems grew tremendously since 1993. To give a sense of how far we are, the Sigpost System in 2005 was the first machine to break the 10,000 process barrier.

Let’s talk about what parallel computers are used for and their application space. The statement is made that the computer science concepts, tools and theorems are getting integrated into every science. The reason why is because computing power is exponentially increasing. Another reason is because a lot of science disciplines are data science. They are based on experiments and observations on data. They requires a increasing demand on analysis on data. Simulation has almost replace most real experiments to overcome the difficulty, expenses, slowness, and danger.

Computer Science Engineering (CSE) is a evolving field that concerns with the science of and engineering of systsm and methodlogies to solve computational problems arising throughout science and engineering. CSE is not just programming.

Another important element in high performing computing is the economic impact. Semiconductor and securities are large customers of HPC but we don’t hear much about them because of the nature of both fields. We went over an example problem on Global Climate Modeling. The problem is basically on doing hundred-year simulations for global climate.

Why is writing parallel programs hard? There are many automatic parallelism in modern machines, which include:

- bit level parallelism
- instruction level parallelism (ILP)
- memory system parallelism
- OS parallelism

but, unfortunately, none of these scales to ten of thousands of cores

There are multiple ways to look at a performance chart. Peak advertised performance (PAP) is the physical maximum computing speed you can get. LINPACK is like the “hello world” program for measuring parallel computing speed. It is highly tuned. There is also the “Gordon Bell Prize winning applications performance”, which is basically the right application/algorithm/platform combination plus years of work. The application is run under highly idealized environment. Last but not least, we have the average sustained applications performance. It’s the performance one can expect for standard applications.

Administrivia. We expect this class to hold about 25 grads and 5 undergrads. We will form interdisciplinary teams for the project groups. I will pass around a signup sheet for scribe next time. There is questionnaire on the website that everyone needs to fill out, so we can make the project teams based on that. There will be 3 homework assignments and one final project for the class. The class website is <http://www.cs.berkeley.edu/~skamil/cs267/>.
