Parallelism and Concurrency

COS 326
David Walker
Princeton University
Parallelism

- What is it?

- Today's technology trends.

- How can we take advantage of it?

- Why is it so much harder to program?

- Some preliminary linguistic constructs
  - thread creation
  - thread coordination: futures and locks
PARALLELISM: WHAT IS IT?
Parallelism

• What is it?
  – doing many things at the same time instead of sequentially (one-after-the-other).
Flavors of Parallelism

- **Data Parallelism**
  - same computation being performed on a lot of data
  - e.g., adding two vectors of numbers

- **Task Parallelism**
  - different computations/programs running at the same time
  - e.g., running web server and database

- **Pipeline Parallelism**
  - assembly line:
Parallelism vs. Concurrency

**Parallelism**: performs many tasks *simultaneously*
- **purpose**: improves throughput
- **mechanism**:
  - many independent computing devices
  - decrease run time of program by utilizing multiple cores or computers
- **eg**: running your web crawler on a cluster versus one machine.

**Concurrency**: mediates multi-party access to shared resources
- **purpose**: decrease response time
- **mechanism**:
  - switch between different threads of control
  - work on one thread when it can make useful progress; when it can't, suspend it and work on another thread
- **eg**: running your clock, editor, chat at the same time on a single CPU.
  - OS gives each of these programs a small time-slice (~10msec)
  - often *slows* throughput due to cost of switching contexts
- **eg**: don't block while waiting for I/O device to respond, but let another thread do useful CPU computation
**Parallelism:**
perform several independent tasks simultaneously

**Concurrency:**
mediate/multiplex access to shared resource

Many efficient programs use some parallelism and some concurrency.
UNDERSTANDING TECHNOLOGY TRENDS
Moore's Law

• Moore's Law: *The number of transistors you can put on a computer chip doubles (approximately) every couple of years.*

• Consequence for most of the history of computing: *All programs double in speed every couple of years.*
  – Why? Hardware designers are wicked smart.
  – They have been able to use those extra transistors to (for example) double the number of instructions executed per time unit, thereby processing speed of programs

• Consequence for application writers:
  – *watch TV for a while and your programs optimize themselves!* 
  – perhaps more importantly: new applications thought impossible became possible because of increased computational power
CPU Clock Speeds from 1993-2005

![Graph showing CPU clock speeds from 1993 to 2005 for AMD and Intel](image)

- Y-axis: Frequency [MHz]
- Two lines represent AMD (green) and Intel (blue) clock speeds over time.
Next year’s machine is twice as fast!
CPU Clock Speeds from 1993-2005
CPU Power 1993-2005

CPU Power Consumption 1993 - 2005

AMD and Intel

Year

Power [W]


AMD

Intel
But power consumption is only part of the problem...cooling is the other!
The Heat Problem
The Problem

2005 Cooler

1993 Pentium Heat Sink
Today: water cooled!
Cray-4: 1994

Up to 64 processors
Running at 1 GHz
8 Megabytes of RAM
Cost: roughly $10M

The CRAY 2, 3, and 4 CPU and memory boards were immersed in a bath of electrically inert cooling fluid.
Power Dissipation

![Graph showing the trend of power density over time for different processors. The x-axis represents the year, ranging from 1980 to 2010, and the y-axis represents power density in W/cm². Key processors such as 386, 486, Pentium, Pentium 2, Pentium 3, Pentium 4, Core2Duo, Core i7, and Prescott are plotted along the curve, indicating increasing power density over time. The graph also includes a note for "Nuclear Reactor" at the top right, but it is not clear if this is a typo or a specific point of interest.]
Darn!
Intel engineers no longer optimize my programs while I watch TV!

Power to chip peaking

Intel CPU Trends
(sources: Intel, Wikipedia, K. Olukotun)
Parallelism

Why is it particularly important (today)?

– Roughly every other year, a chip from Intel would:
  • halve the feature size (size of transistors, wires, etc.)
  • double the number of transistors
  • double the clock speed
  • this drove the economic engine of the IT industry (and the US!)

– No longer able to double clock or cut voltage: a processor won’t get any faster!
  • (so why should you buy a new laptop, desktop, etc.?)
  • power and heat are limitations on the clock
  • errors, variability (noise) are limitations on the voltage
  • but we can still pack a lot of transistors on a chip... (at least for another 10 to 15 years.)
Multi-core h/w – common L2
Tilera announces 64-core processor

Posted on August 20, 2007 - 00:00 by Wolfgang Gruener

Palo Alto (CA) – Silicon Valley startup Tilera today announced the Tile64, a processor with 64 programmable cores that, according to the company, houses ten times the performance and 30 times the power efficiency of Intel's dual-core Xeon processors.

Intel may be getting tired of hearing about products performing better than its dual-core processors targeting server and embedded, as the company describes dual-core processors, at least when it comes to performance, as last year's product.

However, when there's a company claiming that it can beat Intel's last year's product by a factor of 10x and 30x, depending on discipline, it's certainly worth a look.

The Tile64 is a 90 nm RISC-based processor clocked between 600 MHz and 1 GHz aiming for integration in embedded applications such as routers, switches, appliances, video conferencing systems and set-top boxes. Its manufacturer claims that the CPU solves a critical problem in multi-core scaling and opens the door to hundreds or even thousands of cores using this new architecture.
GPUs

- There's nothing like video gaming to drive progress in computation!
- GPUs can have hundreds or even thousands of cores
- Three of the 5 most powerful supercomputers in the world take advantage of GPU acceleration.
- Scientists use GPUs for simulation and modelling
  - eg: protein folding and fluid dynamics
Instead of trying to make your CPU go faster, Intel’s just going to pack more CPUs onto a chip.

– last year: dual core (2 CPUs).
– this year: quad core (4 CPUs).
– Intel is testing 48-core chips with researchers now.
– Within 10 years, you’ll have ~1024 Intel CPUs on a chip.

In fact, that’s already happening with graphics chips (eg, Nvidia).

– really good at simple data parallelism (many deep pipes)
– but they are much dumber than an Intel core.
– and right now, chew up a lot of power.
– watch for GPUs to get “smarter” and more power efficient, while CPUs become more like GPUs.
STILL MORE PROCESSORS: THE DATA CENTER
... So I thought to myself, why clutter your brain up with memories when I can store them in the cloud?
Data Centers: *Lots* of Connected Computers!
Data Centers

• *10s or 100s of thousands* of computers
• All connected together
• Motivated by new applications and scalable web services:
  – let's catalogue all N billion webpages in the world
  – let's all allow any one in the world to search for the page he or she needs
  – let's process that search in less than a second
• It's Amazing!
• It's Magic!
Data Centers: Lots of Connected Computers

Computer containers for plug-and-play parallelism:
Sounds Great!

• So my old programs will run 2x, 4x, 48x, 256x, 1024x faster?
Sounds Great!

• So my old programs will run 2x, 4x, 48x, 256x, 1024x faster?
  – no way!
• So my old programs will run 2x, 4x, 48x, 256x, 1024x faster?
  – no way!
  – to upgrade from Intel 386 to 486, the app writer and compiler writer did not have to do anything (much)
    • IA 486 interpreted the same sequential stream of instructions; it just did it faster
    • this is why we could watch TV while Intel engineers optimized our programs for us
  – to upgrade from Intel 486 to dual core, we need to figure out how to split a single stream of instructions in to two streams of instructions that collaborate to complete the same task.
    • without work & thought, our programs don't get any faster at all
    • it takes ingenuity to generate efficient parallel algorithms from sequential ones