

Parallel Computing: MultiProcessing, MultiThreading, and MultiCore

Mike Bailey

Oregon State University



"If you were plowing a field, which would you rather use -- two strong oxen or 1024 chickens?"

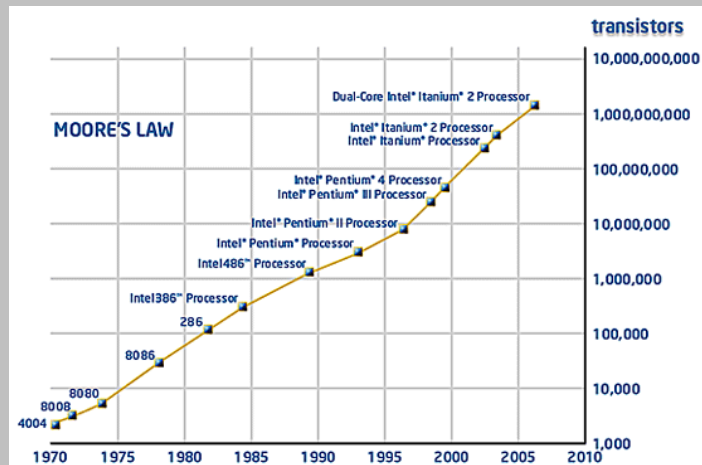
-- Seymour Cray



Oregon State University
Computer Graphics

Moore's Law

"Transistor density doubles every 1.5 years."



Note that oftentimes this proxy is used: "Clock speed doubles every 1.5 years."



Oregon State University
Computer Graphics

Source: <http://www.intel.com/technology/mooreslaw/index.htm>

mjb - November 11, 2007

Clock Speed

1981	IBM PC	5 MHz
1995	Pentium	100 MHz
2002	Pentium 4	3000 MHz (3 GHz)
2007		3800 MHz (3.8 GHz)

Clock speed has hit a wall, largely because of power consumption.

$$\text{PowerConsumption} \propto \text{ClockSpeed} \times \text{Voltage}^2$$

But, the supply voltage you need to provide is proportional to the clock speed, so really:

$$\text{PowerConsumption} \propto \text{ClockSpeed}^3 \quad \text{Yikes!}$$

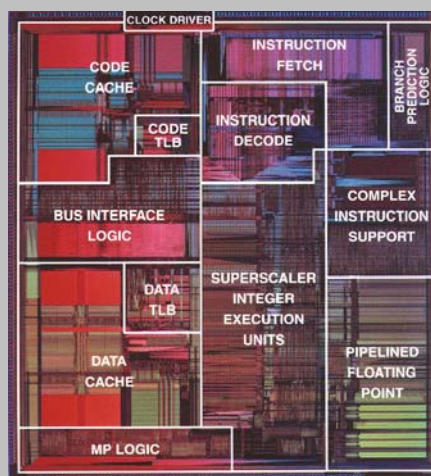


Clock Speed

There is another reason that clock speeds have gotten hard to dramatically increase.

A CPU chip is divided into logical sections, all of which must be able to interact with each other. The clock pulse comes in on a single pin of the chip housing, which is then routed all over the chip.

As clock speeds have increased, it is harder to keep the clock signal synchronized all over the chip. This is known as *clock skew*.



The Problem

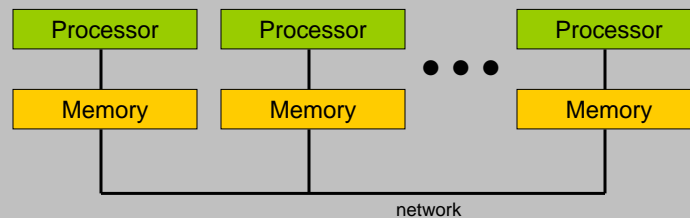
So, if clock speeds have gotten hard to dramatically increase, but we still demand an increase in performance for desktop applications, what can we do?

1. We combine several computer systems, all working together.
2. We look at where CPU time is being wasted, and try to reduce that waste.
3. We get cleverer with architecture and programming.



1. Multiprocessing with Distributed Memory

Instead of one processor in a computing system, put in more than one. Distribute the overall work to the posse of processors. This is called *Parallel Programming*, or *Multiprocessing*.



Because each processor and its memory are distributed across a network, this is oftentimes referred to as *Distributed Computing* or *Cluster Computing*.

This is interesting for many things, but desktop games and simulation are not among them...



2. Eliminating Wasted Time: A Scenario

You are trying to watch two football games on TV at the same time. The commercials come at unpredictable times and are exceedingly long and annoying.

What strategy do you follow to maximize your football viewing and minimize the commercials?

1. Just watch one, and put up with the commercials as wasted time.
2. Flip back and forth at fixed time intervals (e.g., every minute)
3. Watch one until it goes to a commercial, then flip to the other.



2a. Instruction Level Parallelism (ILP)

All code is in one "chunk". When an instruction blocks, keep going as far as you can in that block before you really need that result.

```
x = y + 3;  
fscanf( diskfile, "%f", &z );  
a = x + z;
```



```
fscanf( diskfile, "%f", &z );  
x = y + 3;  
a = x + z;
```



```
fscanf( diskfile, "%f", &z );
```

```
x = y + 3;
```

```
a = x + z;
```



2b. Thread Level Parallelism (TLP)

Code is divided into multiple chunks. When an instruction blocks, or its time slice is over, switch to a new chunk. It's like ILP, but much coarser.

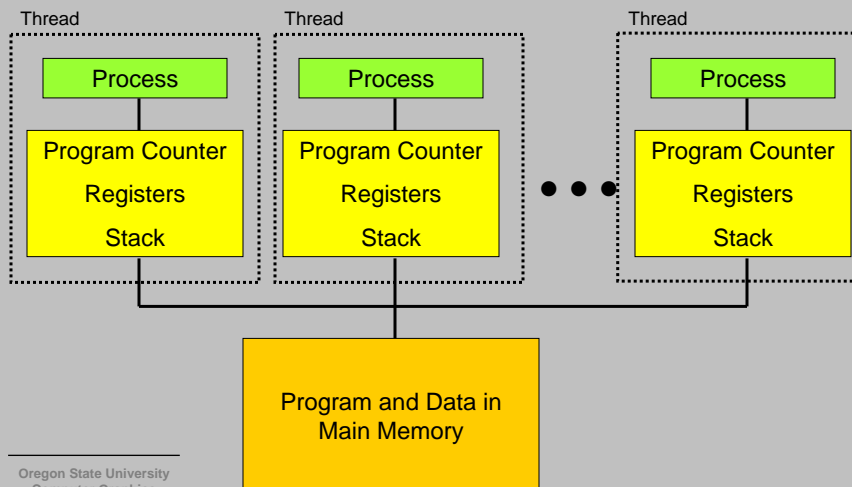
Advantage: sometimes a program can be naturally subdivided into *independent* pieces. Web serving and transaction processing are examples.

Or, sometimes programs can be naturally divided into *cooperating* pieces. In a game, for example, there might be the User Interface, the AI, Physics, and the Graphics all working independently, but cooperating with each other.



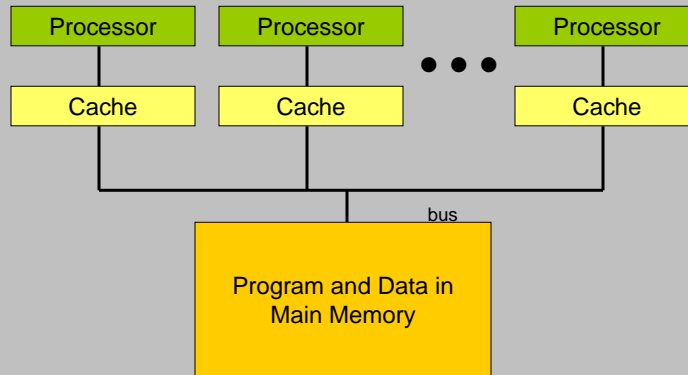
What Exactly is a Thread?

A *thread* is a separate process that has enough state to allow it to execute a common program



3. Clever Architectures: Multiprocessing and Shared Memory

Parallel Programming, or Multiprocessing, can also be accomplished on a single system through shared memory. This is called *Shared Memory Multiprocessing*.



Because each processor has equal access to the whole memory, this is oftentimes referred to as *Symmetric Multiprocessing (SMP)* or *Unified Memory Access (UMA)*

But, adding all those CPU chips becomes expensive!



Oregon State University
Computer Graphics

mjb - November 11, 2007

However, Multiprocessors are Not a Free Lunch: Amdahl's Law

If you put in N processors, you should get N times speedup, right?

Wrong!

If you think of the tasks that a program needs to do as divided between a fraction that is parallelizable and a fraction that isn't (i.e., is stuck at being sequential), then Amdahl's Law says:

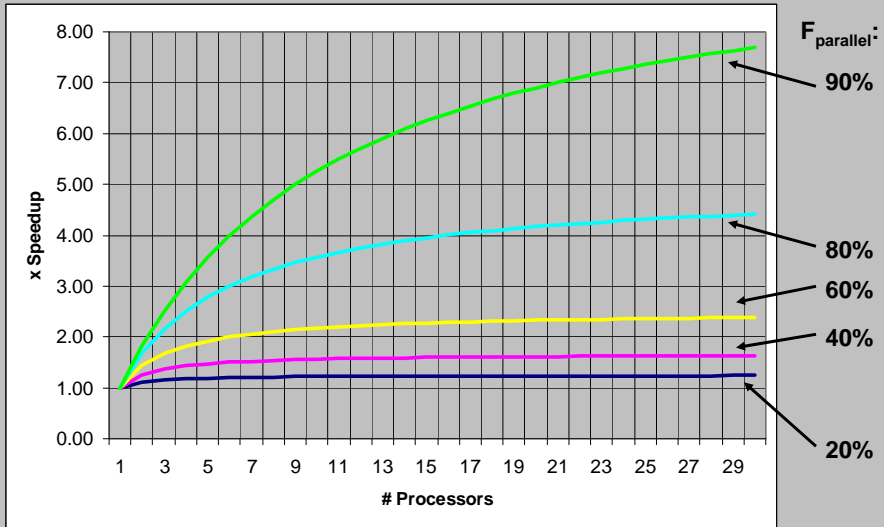
$$xSpeedup = \frac{1}{\frac{F_{parallel}}{\#processors} + F_{sequential}}$$



Oregon State University
Computer Graphics

mjb - November 11, 2007

Amdahl's Law as a Function of Number of Processors and $F_{parallel}$



Oregon State University
Computer Graphics

mjb - November 11, 2007

Amdahl's Law

Note that these fractions put an upper bound on how much benefit you will get from adding more processors:

$$\max Speedup = \lim_{\# processors \rightarrow \infty} xSpeedup = \frac{1}{F_{sequential}} = \frac{1}{1 - F_{parallel}}$$

Fparallel	maxSpeedup
0.00	1.00
0.10	1.11
0.20	1.25
0.30	1.43
0.40	1.67
0.50	2.00
0.60	2.50
0.70	3.33
0.80	5.00
0.90	10.00
0.95	20.00
0.99	100.00



Oregon State University
Computer Graphics

mjb - November 11, 2007

MultiCore

So, to summarize:

1. Moore's Law of transistor density is still going strong, but the Moore's Law of clock speed has hit a dead end.
2. Multiple CPU chips are an option, but it is too expensive for desktop applications. Now what do we do?

Keep packing more and more transistors on a CPU chip, but don't increase the clock speed. Instead, increase computational throughput by using those transistors to have multiple processors on the same chip.

Originally this was called *single chip multiprocessing*, but now it is called *multicore*.



Oregon State University
Computer Graphics

mjb – November 11, 2007

MultiCore

MultiCore is a very hot topic these days. The chip vendors are implementing all new chips this way. We, as programmers, can no longer take the La-Z-Boy approach to getting program speedups.

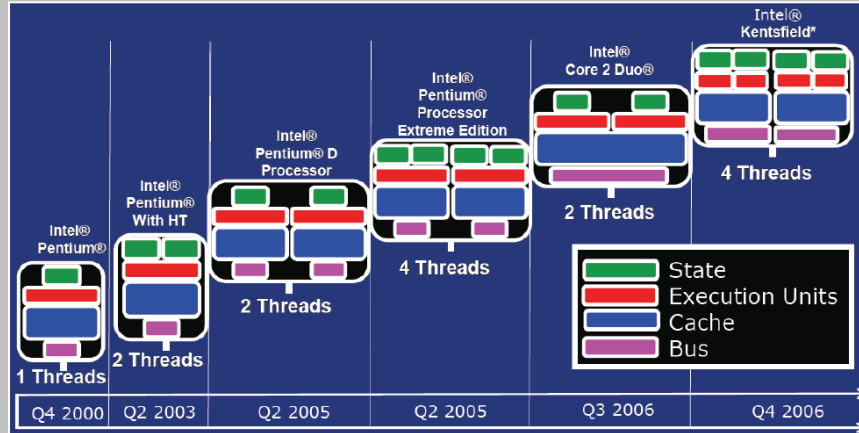
We need to be prepared to convert our programs to run on **MultiThreaded Shared Memory Multiprocessing** architectures.



Oregon State University
Computer Graphics

mjb – November 11, 2007

A Comparison of Intel Processors



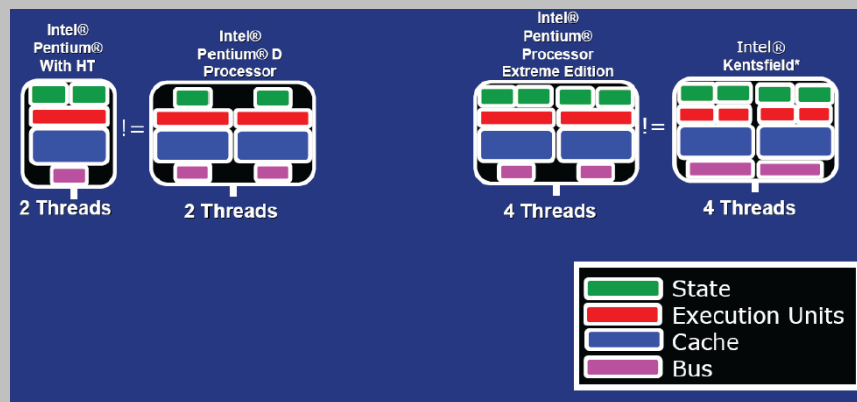
Credit: Intel



Oregon State University
Computer Graphics

mjb - November 11, 2007

A Comparison of Intel Processors



Credit: Intel



Oregon State University
Computer Graphics

mjb - November 11, 2007