GPUs and Accelerators: Exercises

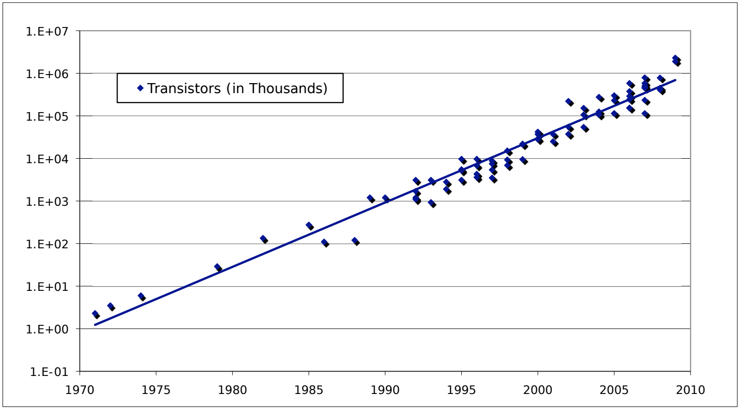
Karl Ljungkvist

19 February 2016

Today

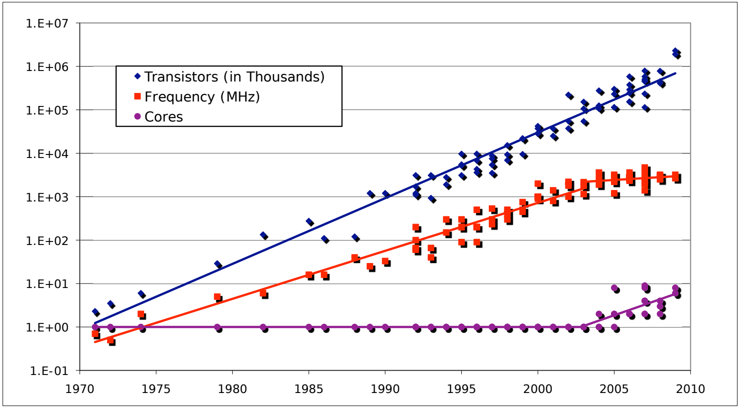
- Some contemporary history of processors
- Exercises

Moore's law



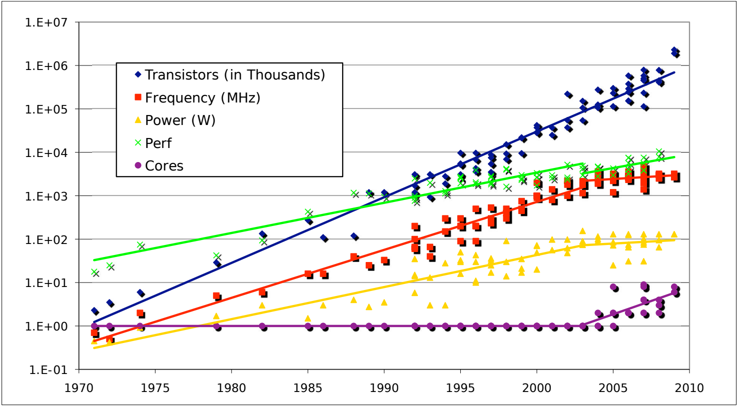
- Exponential increase in transistors on a chip

Moore’s law



- CPUs no longer become faster !
- Instead Mo(o)re cores.

Moore’s law



- Still maintain exponentially increasing performance

Why no faster CPUs? (1)

Three “walls”:

- Power wall
  - Shrinking transistors stopped lowering power consumption
  - Heat and cooling issues
- Memory wall
  - Memory frequency did not keep up with processor
- ILP wall
  - Could not find more parallelism in serial stream

Why no faster CPUs? (2)

Power consumption of chip:

- $P \propto f V^2$
- Performance $\propto f$
- V : Voltage
- f : Frequency

Previously – “Dennard scaling”:

- Smaller transistors $\Rightarrow$ lower V and higher f

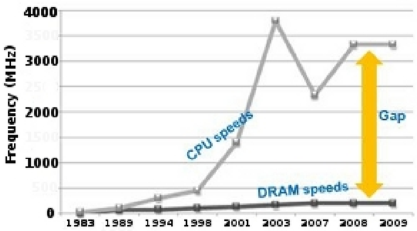
	Voltage	Freq.	Power	Perf.	Perf./Power
Old	1	1	1	1	1
New	$0.7\times$	$2\times$	$1\times$	$2\times$	$2\times$

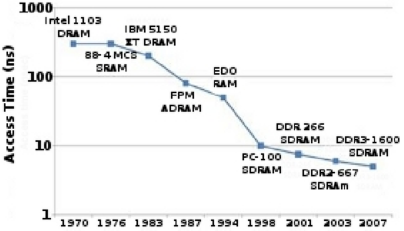
Now – “Power wall”:

- Can no longer lower voltage due to leakage currents
- $f \propto V$

	Voltage	Freq.	Power	Perf.	Perf./Power
Old	1	1	1	1	1
New	$2\times$	$2\times$	$8\times$	$2\times$	$0.25\times$

Why no faster CPUs? (3)





Memory wall:

- Memory frequency has not scaled at the same rate as the processor frequency
- Increasingly difficult to keep up with processor
- Caches can help, but only to some extent

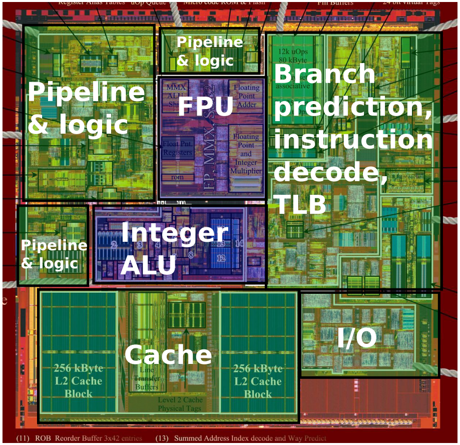
Why no faster CPUs? (4)

Instruction-Level Parallelism (ILP):

- Processor tries to find independent instructions in serial stream
- Pipeline, branch prediction, prefetching, etc
- Example: Pentium 4 (Northwood)

ILP wall:

- Increasingly difficult to find ILP
- This is inefficient!


Pipeline & logic
Pipeline & logic
FPU
Branch prediction, instruction decode, TLB
Pipeline & logic
Integer ALU
Cache
I/O
256 kByte L2 Cache Block
256 kByte L2 Cache Block
Level 2 Cache Physical Tags
(11) ROB Reorder Buffer 3x42 entries
(13) Summed Address Index decode and Way Predict

Multicores to the rescue

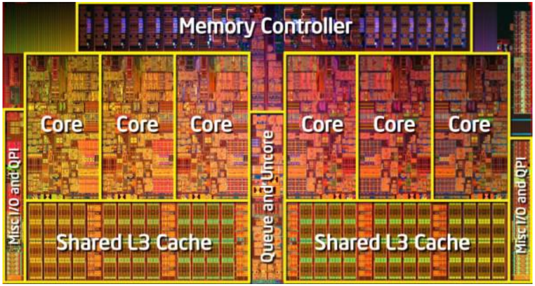
Idea:

- Power(1x Fast core) $>$ Power(2x Slower cores)
- Example

	Cores	Voltage	Freq.	Power	Perf.	Perf./Power
Single	1	1	1	1	1	1
Multicore	2	$0.75\times$	$0.75\times$	$0.84\times$	$1.8\times$	$1.8\times$

- More efficient!

Example:
Intel Xeon 5600 6-core



Power is everything

Change of architecture design:

- Previously:
  Try to make any serial program as fast as possible
- Now:
  Try to provide performance from transistors as efficiently as possible

Implications:

- Multiple cores
- Performance becomes a software issue
- More dedicated and specialized hardware $\Rightarrow$ accelerators

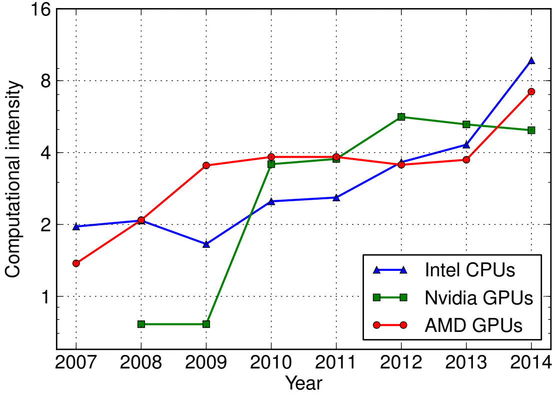
Operational intensity

Memory wall:

- Difference between compute and bandwidth is growing

Operational intensity:

$$CI = \frac{Gflop/s}{GB/s} = \frac{flop}{byte}$$

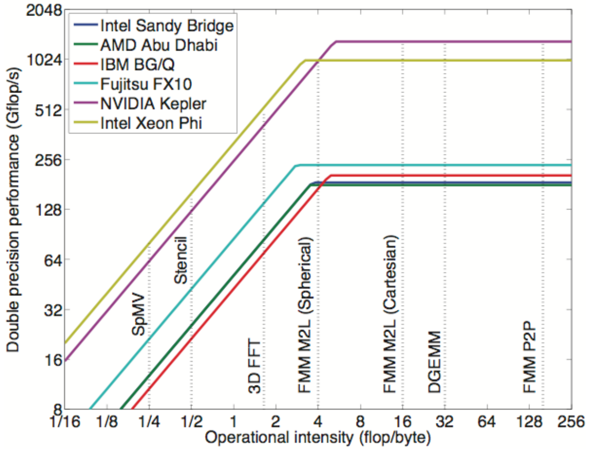


Bandwidth-bound algorithms:

- If low *CI*, no speedup!
- Trade computations for data ⇒ recompute
- Be wasteful!

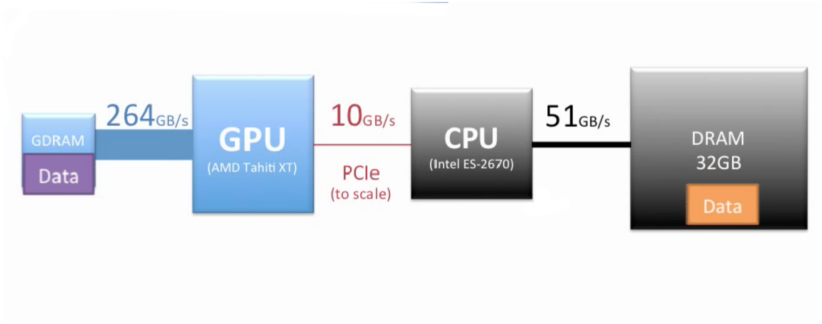
Operational intensity

Roofline model:



Exercises

CPU+Accelerator



What performance can we expect?

Example: vector update

Code:

```
for(int i=0; i<N; ++i)
  x[i] = x[i] + y[i];
```

System:

- Processor:
  - Compute power: 40 GFlop/s
  - Bandwidth: 25 GByte/s

Question:
What will the performance be? $N = 10^7$

Hint: Data $\sim 3N \cdot 4\text{Bytes}$, Flop $\sim N$

Example: vector update

Code:

```
for(int i=0; i<N; ++i)
  x[i] = x[i] + y[i];
```

System:

- CPU: 40 GFlop/s, 25 GB/s

Hint: Data $\sim 3N \cdot 4$Bytes, Flop $\sim N$

Solution:
Computations: $\frac{10^7\,\mathrm{Flop}}{40\mathrm{Gflop/s}} = \frac{10^7}{40 \cdot 10^9}\mathrm{s} = 0.25\mathrm{ms}$
Bandwidth: $\frac{12 \cdot 10^7\,\mathrm{Byte}}{25\mathrm{GByte/s}} = \frac{12 \cdot 10^7}{25 \cdot 10^9}\mathrm{s} = 4.8\mathrm{ms}$
Performance: $\frac{10^7\,\mathrm{Flop}}{\max(4.8\mathrm{ms}, 0.25\mathrm{ms})} = \frac{10 \cdot 10^6}{4.8 \cdot 10^{-3}}\mathrm{Flop/s} = 2\mathrm{GFlop/s}$

Example: vector update

Code:

```
for(int i=0; i<N; ++i)
  x[i] = x[i] + y[i];
```

System:

- Processor:
  - Compute power: 40 GFlop/s
  - Bandwidth: 25 GByte/s
- Accelerator:
  - Compute power: 500 GFlop/s
  - Bandwidth: 100 GByte/s
  - PCI-E bandwidth: 3 GByte/s

Question:
Performance on the accelerator? Assume data is in the system RAM. $N = 10^7$
Hint: Data $\sim 3N \cdot 4$Bytes, Flop $\sim N$

Example: vector update

Code:

```
for(int i=0; i<N; ++i)
  x[i] = x[i] + y[i];
```

System:

- CPU: 40 GFlop/s, 25 GB/s
- GPU: 500 GFlop/s, 100 GB/s, PCI-E: 3 GB/s

Hint: Data $\sim 12N$Bytes, Flop $\sim N$

Solution:

Computations: $\frac{10^7 \text{Flop}}{500\text{Gflop/s}} = \frac{10^7}{500 \cdot 10^9}\text{s} = 0.02\text{ms}$

Bandwidth: $\frac{12 \cdot 10^7 \text{Byte}}{100\text{GByte/s}} = \frac{12 \cdot 10^7}{100 \cdot 10^9}\text{s} = 1.2\text{ms}$

Performance: $\frac{10^7 \text{Flop}}{\max(0.02\text{ms}, 1.2\text{ms})} = \frac{10 \cdot 10^6}{1.2 \cdot 10^{-3}}\text{Flop/s} = 8\text{GFlop/s}$

But:

PCI-E: $\frac{12 \cdot 10^7 \text{Byte}}{3\text{GByte/s}} = \frac{12 \cdot 10^7}{3 \cdot 10^9}\text{s} = 40\text{ms} \Rightarrow 0.25\text{GFlop/s}$

Example: matrix-matrix multiplication

Code:

```
for(int i=0; i<N; ++i)
  for(int j=0; j<N; ++j)
    for(int k=0; k<N; ++k)
      C[i][j] += A[i][k]*B[k][j];
```

Same System:

- CPU: 40 GFlop/s, 25 GB/s
- GPU: 500 GFlop/s, 100 GB/s, PCI-E: 3 GB/s

Question:
Performance on CPU/accelerator? Assume data is in the system RAM. $N = 1000$

Hint: Data $\sim 3N^2 \cdot 4$ Bytes, Flop $\sim 2 \cdot N^3$

Example: matrix-matrix multiplication

System:

- CPU: 40 GFlop/s, 25 GB/s
- GPU: 500 GFlop/s, 100 GB/s, PCI-E: 3 GB/s

Hint: Data $\sim 12N^2$ Bytes, Flop $\sim 2 \cdot N^3$

CPU:

Computations: $\frac{2 \cdot 1000^3 \mathbf{Flop}}{40 \mathrm{Gflop/s}} = \frac{2 \cdot 10^9}{40 \cdot 10^9} \mathrm{s} = 50 \mathrm{ms}$

Bandwidth: $\frac{12 \cdot 1000^2 \mathbf{Byte}}{25 \mathrm{GByte/s}} = \frac{12 \cdot 10^6}{25 \cdot 10^9} \mathrm{s} = 0.48 \mathrm{ms}$

Performance: $\frac{2 \cdot 10^9 \mathbf{Flop}}{50 \mathrm{ms}} = 40 \mathrm{GFlop/s}$

Accelerator:

Computations: $\frac{2 \cdot 1000^3 \mathbf{Flop}}{500 \mathrm{Gflop/s}} = \frac{2 \cdot 10^9}{500 \cdot 10^9} \mathrm{s} = 4 \mathrm{ms}$

Bandwidth: $\frac{12 \cdot 1000^2 \mathbf{Byte}}{100 \mathrm{GByte/s}} = \frac{12 \cdot 10^6}{100 \cdot 10^9} \mathrm{s} = 0.12 \mathrm{ms}$

PCI-E: $\frac{12 \cdot 1000^2 \mathbf{Byte}}{3 \mathrm{GByte/s}} = \frac{12 \cdot 10^6}{3 \cdot 10^9} \mathrm{s} = 4 \mathrm{ms}$

Performance: $\frac{2 \cdot 10^9 \mathbf{Flop}}{4+4 \mathrm{ms}} = 250 \mathrm{GFlop/s}$