CS 316: Multicore/GPUs

Kavita Bala

Fall 2007

Computer Science

Cornell University

Announcements

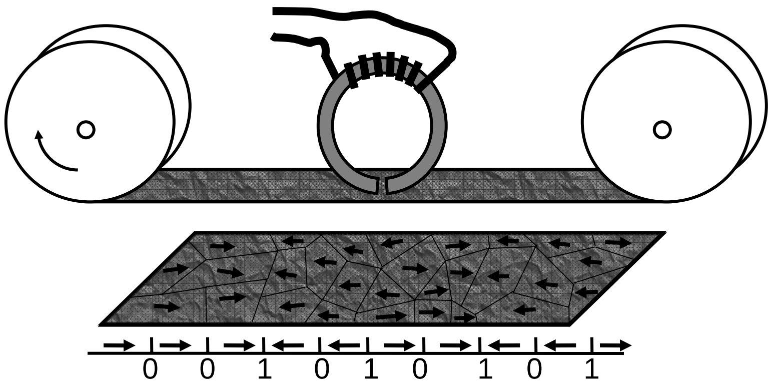
- Core Wars will be out in the next couple of days
  - Aim at having fun!
  - Number of points allocated to it is small (10-20% of other assignments)
  - 5 points for turning something in, 1 point more for going up the ladder
- Pizza party on last day of class
  - Showdown
  - Friday Nov 30th
- Final project (distributed ray tracer) out last week
  - Demoed: Dec 13 2-4:30

Memory Hierarchy

16 KB	registers/L1	2 ns, random access
512 KB	L2	5 ns, random access
2 GB	DRAM	20-80 ns, random access
300 GB	Disk	2-8 ms, random access
1 TB	Tape	100s, sequential access

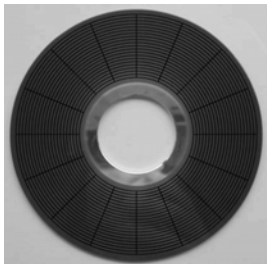
Tapes

- Same basic principle for 8-tracks cassettes, VHS, atari tape drive, tape storage

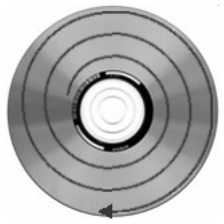


Disks & CDs

- Disks use same magnetic medium as tapes
  - concentric rings (not a spiral)

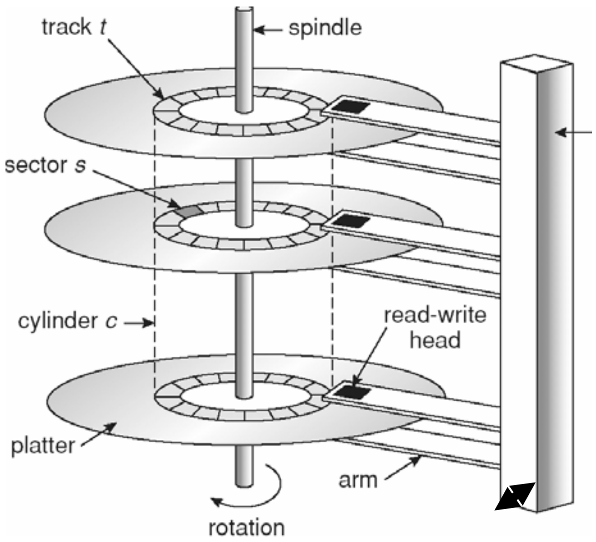


- CDs & DVDs use optics, and a single spiral track



- Non-volatile

Disk Physics

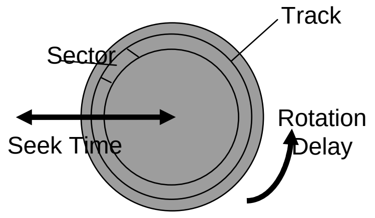


Typical parameters :
- 1 spindle
- 1 arm assembly
- 1-4 platters
- 1-2 sides/platter
- 1 head per side
- (but only 1 active head at a time)
- 4,200 – 15,000 RPM

Disk Accesses

- Accessing a disk requires:
  - – specify sector: C (cylinder), H (head), and S (sector)
  - – specify size: number of sectors to read or write
  - – specify memory address: bus address to DMA to

- Performance:
  - seek time: move the arm assembly to track
  - Rotational delay: wait for sector to come around
  - transfer time: get the bits off the disk



Example

- Average time to read/write 512-byte sector
  - – Disk rotation at 10,000 RPM
  - – Seek time: 6ms
  - – Transfer rate: 50 MB/sec
  - – Controller overhead: 0.2 ms
- Average time:
  - – Seek time + rotational delay + transfer time + controller overhead
  - – 6ms + 0.5 rotation/(10,000 RPM) + 0.5KB/(50 MB/sec) + 0.2ms
  - – 6.0 + 3.0 + 0.01 + 0.2 = 9.2ms

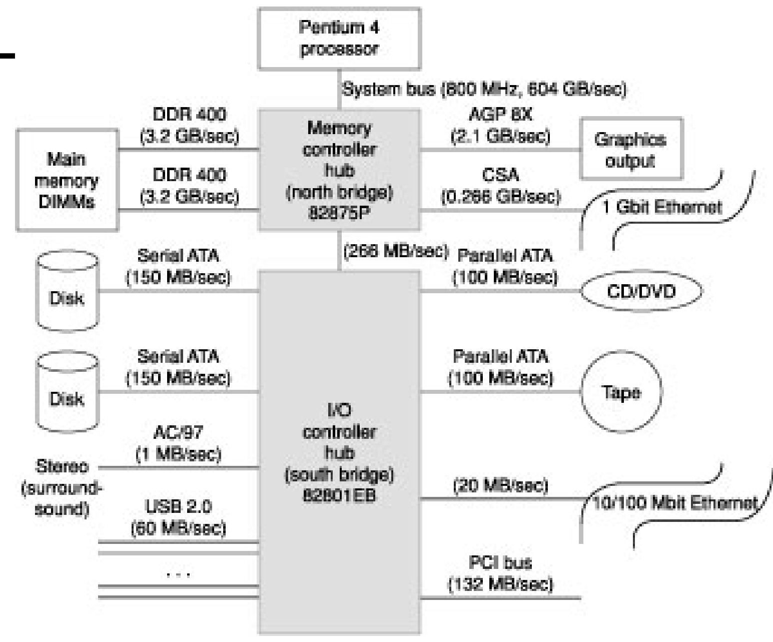
Disk Scheduling

- Goal: minimize seek time
  - secondary goal: minimize rotational latency
- FCFS (First come first served)
- Shortest seek time
- SCAN/Elevator
  - First service all requests in one direction
  - Then reverse and serve in opposite direction
- Circular SCAN
  - Go off the edge and come to the beginning and start all over again

What we didn't talk about

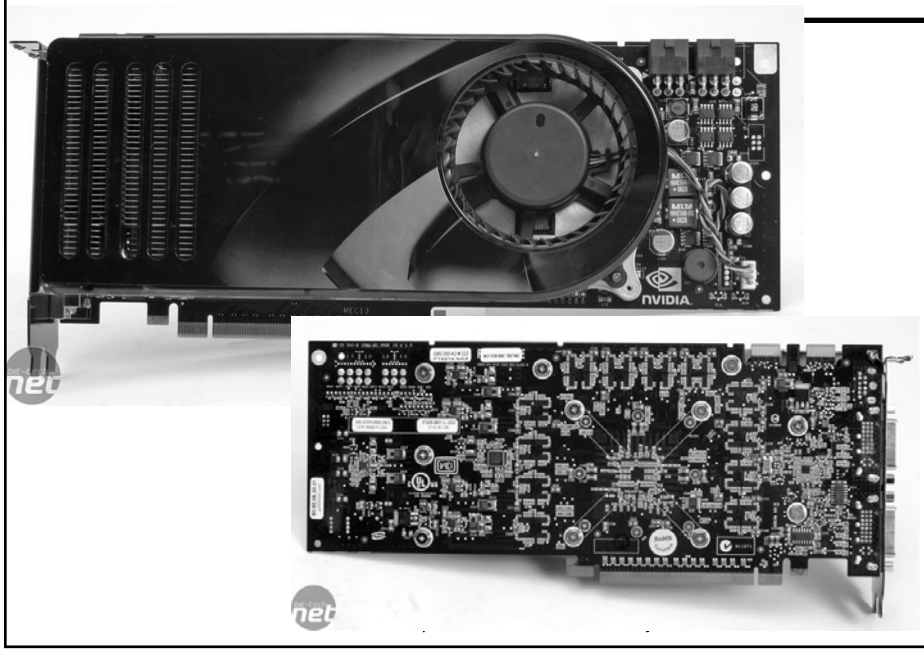
- RAID
  - Redundancy for fault tolerance
  - Speed
- Solid State drives
  - Very expensive still

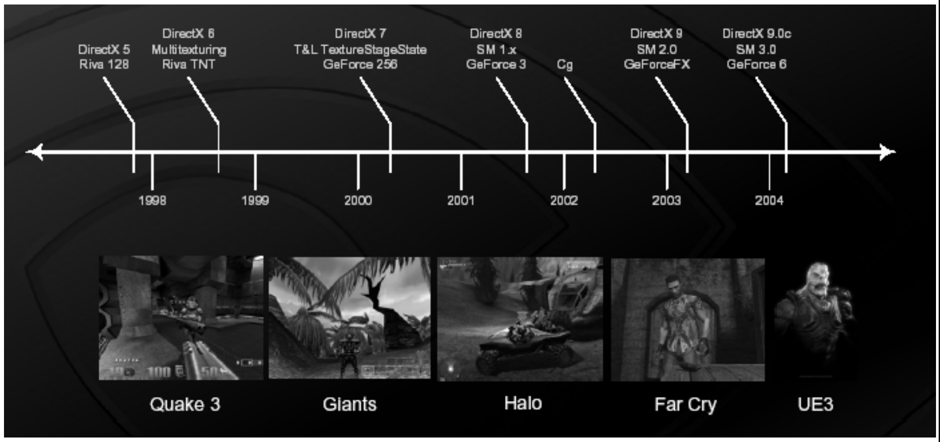
GPUs



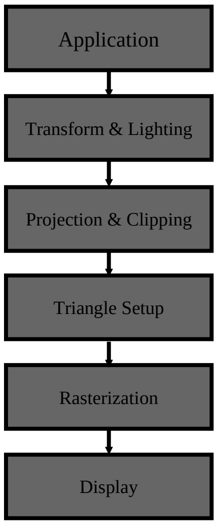
Pentium 4 processor
System bus (800 MHz, 604 GB/sec)
DDR 400 (3.2 GB/sec)
DDR 400 (3.2 GB/sec)
Main memory DIMMs
Memory controller hub (north bridge) 82875P
AGP 8X (2.1 GB/sec)
Graphics output
CSA (0.266 GB/sec)
1 Gbit Ethernet
(266 MB/sec)
Serial ATA (150 MB/sec)
Disk
Parallel ATA (100 MB/sec)
CD/DVD
Serial ATA (150 MB/sec)
Disk
Parallel ATA (100 MB/sec)
Tape
AC/97 (1 MB/sec)
Stereo (surround-sound)
I/O controller hub (south bridge) 82801EB
(20 MB/sec)
10/100 Mbit Ethernet
USB 2.0 (60 MB/sec)
. . .
PCI bus (132 MB/sec)

NVidia G80 Architecture

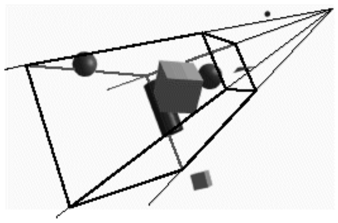


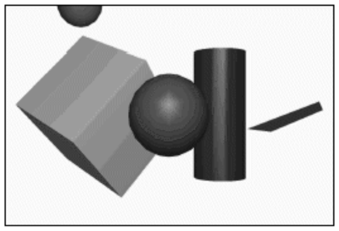


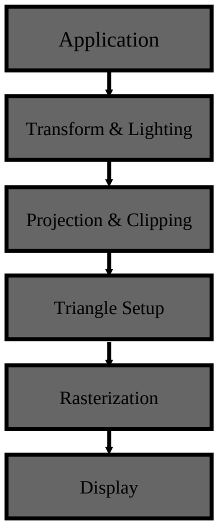
Traditional Graphics Pipeline



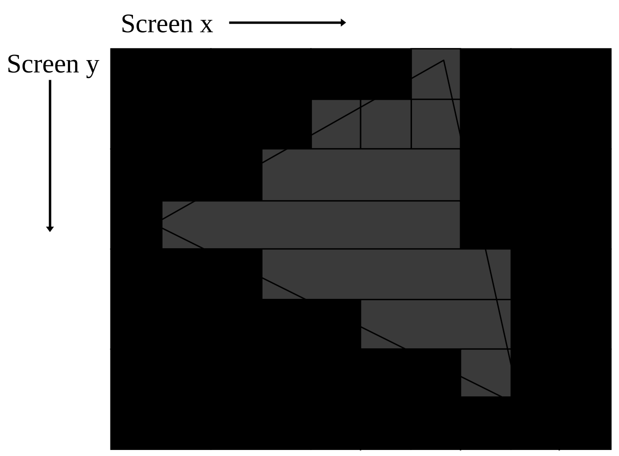
Projection & Clipping

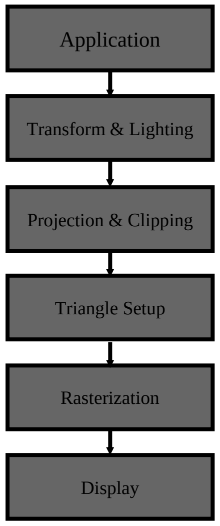

View frustum


Eye view

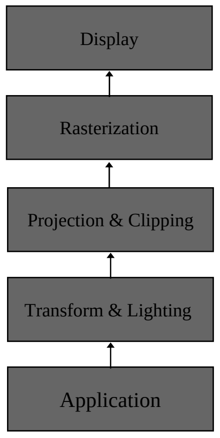


Rasterization & Z-buffer





Brief History



The dark ages (early-mid 1990's), when there were only frame buffers for normal PC's.

Some accelerators were no more than a simple chip that sped up linear interpolation along a single span, so increasing fill rate.

This is where pipelines start for PC commodity graphics, prior to Fall of 1999.

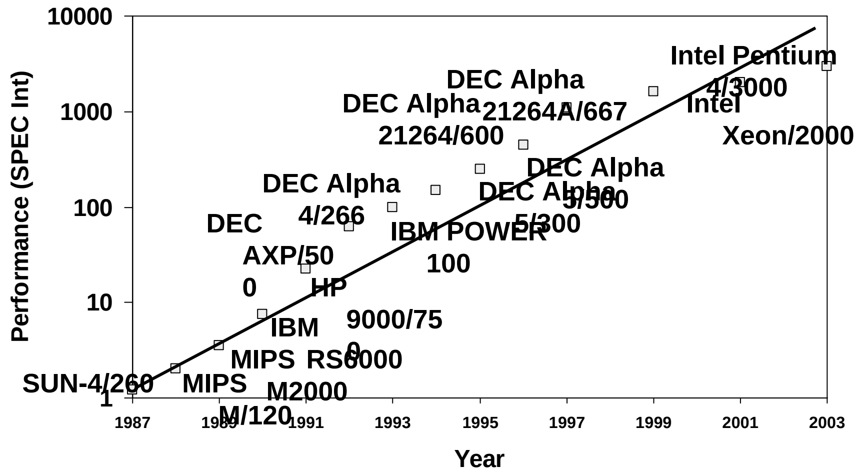
This part of the pipeline reaches the consumer level with the introduction of the NVIDIA GeForce256.

Hardware today is moving traditional application processing (surface generation, occlusion culling) into the graphics accelerator.

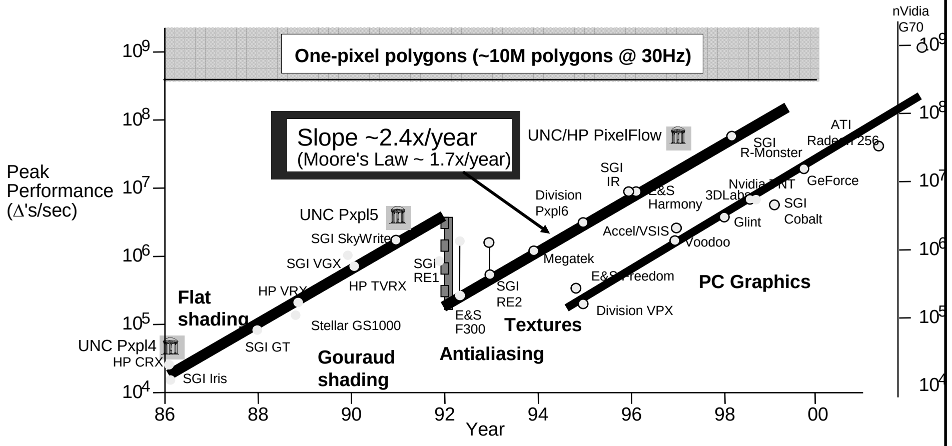
Moore's Law

- 1965
  - number of transistors that can be integrated on a die would double every 18 to 24 months (i.e., grow exponentially with time).
- Amazingly visionary
  - 2300 transistors, 1 MHz clock (Intel 4004) - 1971
  - 16 Million transistors (Ultra Sparc III)
  - 42 Million transistors, 2 GHz clock (Intel Xeon) – 2001
  - 55 Million transistors, 3 GHz, 130nm technology, 250mm^2 die (Intel Pentium 4) – 2004
  - 290+ Million transistors, 3 GHz (Intel Core 2 Duo) – 2007

Processor Performance Increase

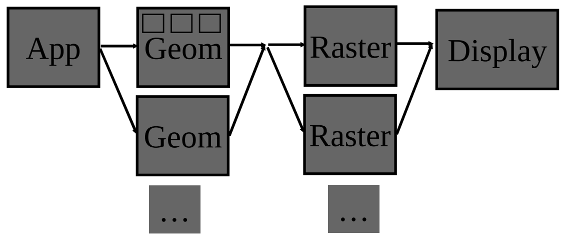


Faster than Moore’s Law



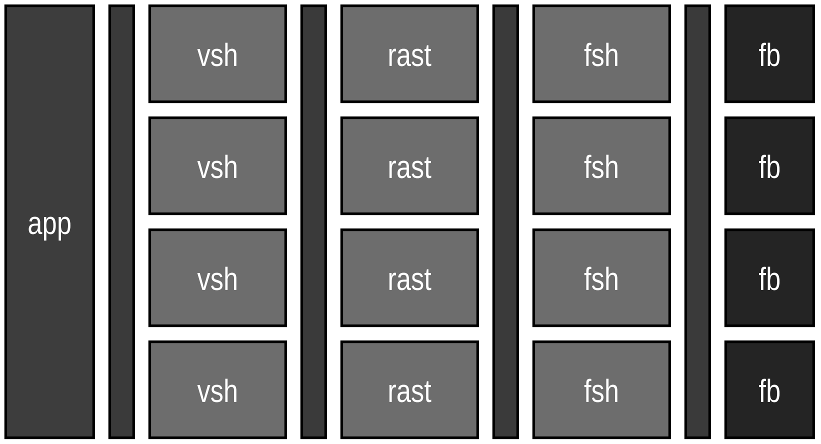
Graph courtesy of Professor John Poulton (from Eric Haines)

Why are GPUs so fast?



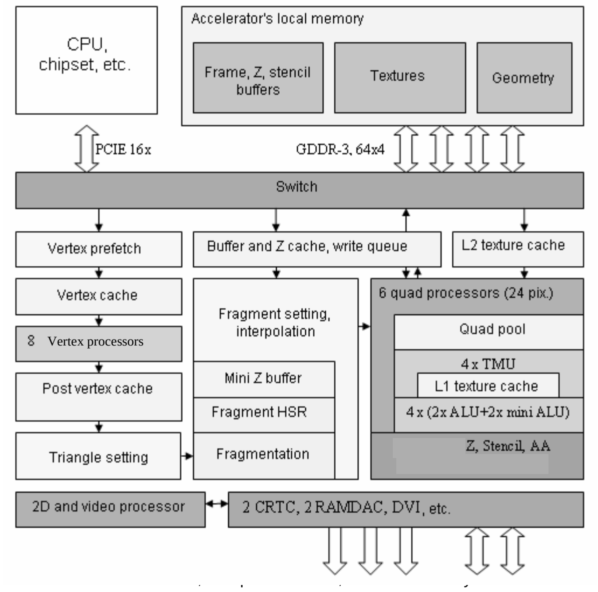
- Pipelined and parallel
- Very, very deep pipeline: 800-1000 deep

GPU Parallelism

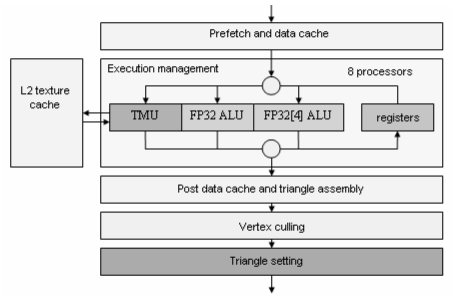


Kavita Bala, Computer Science, Cornell University

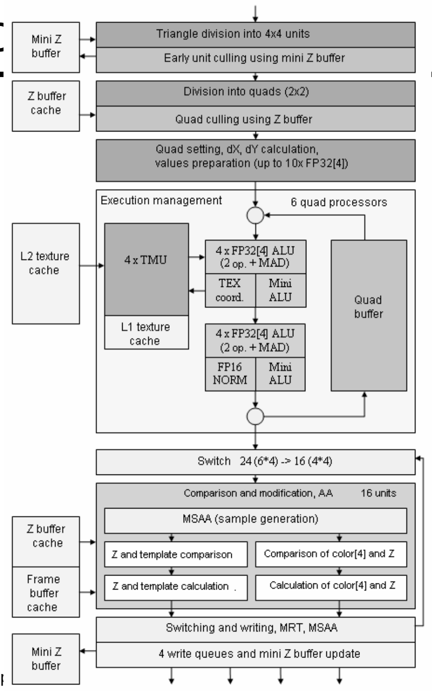
G70 Hardware Architecture



Vertex Processor



Pixel Processo


Mini Z buffer
Triangle division into 4x4 units
Early unit culling using mini Z buffer
Z buffer cache
Division into quads (2x2)
Quad culling using Z buffer
Quad setting, dX, dY calculation, values preparation (up to 10x FP32[4])
Execution management
6 quad processors
L2 texture cache
4 x TMU
L1 texture cache
4 x FP32[4] ALU (2 op. + MAD)
TEX coord.
Mini ALU
4 x FP32[4] ALU (2 op. + MAD)
FP16 NORM
Mini ALU
Quad buffer
Switch 24 (6*4) -> 16 (4*4)
Comparison and modification, AA
16 units
MSAA (sample generation)
Z buffer cache
Z and template comparison
Comparison of color[4] and Z
Frame buffer cache
Z and template calculation
Calculation of color[4] and Z
Switching and writing, MRT, MSAA
Mini Z buffer
4 write queues and mini Z buffer update

Parallelism

- Critical to achieving performance
- Flynn's taxonomy
  - SISD (Single instruction, single data)
    - Boring CPU
  - MISD (Multiple instruction, single data)
    - Redundant processing
  - SIMD (Single instruction, multiple data)
    - GPUs
  - MIMD (Multiple instruction, multiple data)
    - Multicore, Cell processor

Parallelism

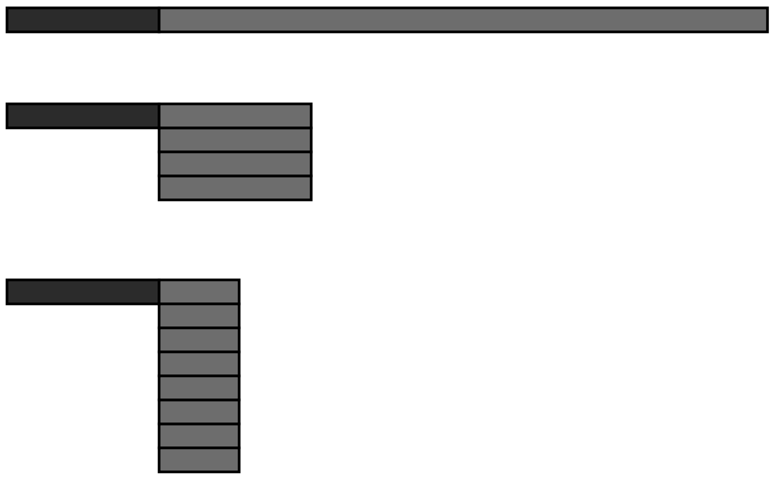
- *Must* exploit parallelism for performance
  - Lot of parallelism in graphics applications
- SIMD: single instruction, multiple data
  - Perform same operation in parallel on many data items
  - Data parallelism
- MIMD: multiple instruction, multiple data
  - Run separate programs in parallel (on different data)
  - Task parallelism

Performance Tuning

1. Identify bottleneck (slowest stage)
2. Improve performance of slowest stage

Repeat as necessary

Amdahl's Law



Amdahl’s Law

- Task: serial part, parallel part
- As number of processors increases,
  - time to execute parallel part goes to zero
  - time to execute serial part remains the same
- *Serial part eventually dominates*
- Must parallelize ALL parts of task

$$\text{Speedup}(E) = \frac{\text{Execution Time without } E}{\text{Execution Time with } E}$$

Kavita Bala, Computer Science, Cornell University

Amdahl’s Law

- Consider an improvement E
- F of the execution time is affected
- S is the speedup

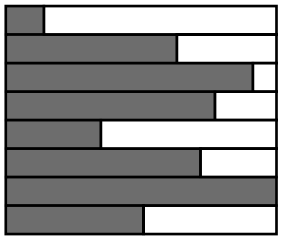
$$\text{Execution time (with } E) = ((1 - F) + F/S) \cdot \text{Execution time (without } E)$$

$$\text{Speedup (with } E) = \frac{1}{(1 - F) + F/S}$$

Kavita Bala, Computer Science, Cornell University

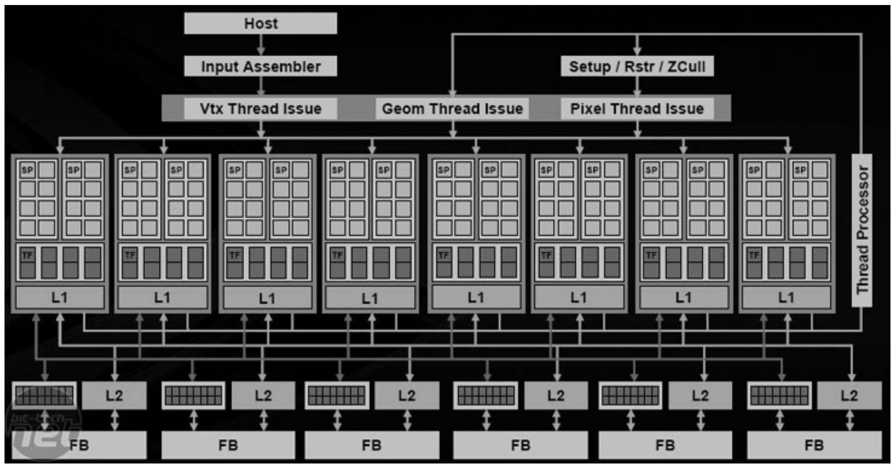
Load Balancing

- Need to manage work so all units are actually operating



Kavita Bala, Computer Science, Cornell University

G80



Kavita Bala, Computer Science, Cornell University

GPGPUs

- Scientific Computing
  - MATLAB codes
- Convex hulls
- Molecular Dynamics
- Etc.