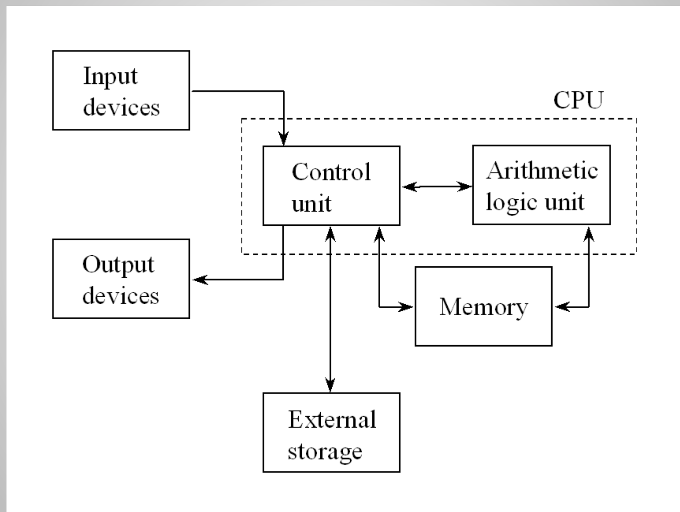
Computational Astrophysics: Methodology

1. Identify astrophysical problem
2. Write down corresponding equations
3. Identify numerical algorithm
4. Find a computer
5. Implement algorithm, generate results
6. Visualize data

Computer Architecture

- Components that make up a computer system, and their interconnections.
- Basic components:
    1. Processor
    2. Memory
    3. I/O
    4. Communication channels



Processors

- Component which executes a program.
- Most PCs (used to have) only one processor (CPU) – these are “serial” or “scalar” machines.
- High-performance machines usually have many processors – these are “vector” or “parallel” machines.
- Nowadays PCs have CPUs with 16-32 cores, and GPU (graphic processor units) with 2600 cores.

Fetch-Decode-Execute

- Processors execute a…

  fetch - get instruction from memory

  decode - store instruction in register

  execute - perform operation

  …cycle. (e.g., LD A,R1; LD B,R2; ADD R1,R2,R3; STORE R3,C).
  - Instruction address held in program counter (PC).
  - PC incremented after each cycle.
- Very primitive commands! “Compilers” or “interpreters” are used to translate high-level code into such low-level operations.

Cycles

- Timing of cycle depends on internal construction & complexity of instructions.
- Quantum of time in a processor is called a “clock cycle”. All tasks take an integer number of clock cycles to occur.
- The fewer the clock cycles for a task, the faster it occurs.
- NOTE: Higher clock speeds imply faster heating of components, increasing cooling requirements: GHz are common these days.

Measuring CPU Performance

- Time to execute a program:

  $t = n_i \times CPI \times t_c$

  where

  n_i = number of instructions

  CPI = cycles per instruction

  t_c = time per cycle

Improving Performance

1. Obviously, can decrease t_c. Mostly engineering problem (e.g. increase clock frequency, use better chip materials, …).
2. Decrease CPI, e.g. by making instructions as simple as possible (RISC --- Reduced Instruction Set Computer). Can also "pipeline" (a form a parallelism/latency hiding).

Improving Performance, Cont'd

3. Decrease n_i any one processor works on:
   - Improve algorithm.
   - Distribute n_i over n_p processors, thus ideally $n_i^* = n_i/n_p$.
     - Actually, process of distributing work adds overhead: $n_i^* \rightarrow n_i/n_p + n_o$.
     - Will return to high-performance/parallel computing toward the end of the course.

Defining Performance

- MIPS – "million instructions per second": not useful due to variations in instruction length, implementation, etc.
- MFLOPS – "million floating-point operations per second": measures time to complete a meaningful complex task, e.g. multiplying two matrices ~ n^3 ops.

Defining Performance, Cont'd

- Computer A and computer B may have different MIPS but same MFLOPS.
- Often refer to "peak MFLOPS" (highest possible performance if machine only did arithmetic calculations) and "sustained MFLOPS" (effective speed over entire run).
- Nowadays, supercomputer centers aim at Peta-flop (Peta=10^{15}) performance
- "Benchmark": standard performance test.

Memory

- Passive component which stores data or instructions, accessed by address.
- Data flows <u>from</u> memory ("read") or <u>to</u> memory ("write").
- RAM: "Random Access Memory" supports both reads and writes.
- ROM: "Read Only Memory" – no writes.

Bits & Bytes

- Smallest piece of memory = 1 bit (off/on)
  - 8 bits = 1 byte
  - 4 bytes = 1 word (on 32-bit machines)
  - 8 bytes = 1 word (on 64-bit machines)
- 1 word = number of bits used to store single-precision floating-point number. Usually equals width of data bus.
- Typical home computers these days have ~ 1-64 Gbyte of useable RAM.
- 1 MB = 1 megabyte or 1,048,576 (2^{20}) bytes (sometimes just 10^6).
- 1 Mb = 1 megabit or 10^6 bits (rarely 2^{20}).

Memory Performance

- Determined by <u>access time</u> or <u>latency</u>, usually 10-80 ns.[a]
  - Latency hiding: perform other operations while waiting for memory to respond.
- Would like to build all memory from fastest chips, but this is often too expensive.
- Instead, exploit "locality of reference".

[a] Note: DDR-SDRAM (double data rate, synchronous dynamic RAM), the newest type of memory, is speed-rated in terms "memory cycles," i.e., the time required between successive memory accesses, typically ~ 10 ns or less.

Locality of Reference

- Typical applications store and access data in sequence.
- Instructions also sequentially stored in memory.
- Hence if address M accessed at time t, there is a high probability that address M + 1 will be accessed at time t + 1 (e.g. vector ops).

Hierarchical Memory

- Instead of building entire memory from fast chips, use hierarchical memory:
  - Memory closest to processor built from fastest chips – "cache".
  - Main memory built from RAM – "primary memory".
  - Additional memory built from slowest/cheapest components (e.g. hard disks) – "secondary memory".

Hierarchical Memory, Cont'd

- Then, transfer entire blocks of memory between levels, not just individual values.
  - Block of memory transferred between cache and primary memory = "cache line".
  - Between primary and secondary memory = "page".
- How does it work?

The Cache Line

- If processor needs item x, and it's not in cache, request forwarded to primary memory.
- Instead of just sending x, primary memory sends entire cache line (x, x+1, x+2, …).
- Then, when/if processor needs x+1 next cycle, it's already there.

Hits & Misses

- Memory request to cache which is satisfied is called a "hit".
- Memory request which must be passed to next level is called a "miss".
- Fraction of requests which are hits is called the "hit rate".
- Must try to optimize hit rate (> ~90%).

Effective Access Time

- $t_{eff} = (HR)\, t_{cache} + (1 - HR)\, t_{pm}$

  t_{cache} = access time of cache

  t_{pm} = access time of primary memory

  HR = hit rate

  e.g. t_{cache} = 1 ns, t_{pm} = 10 ns, HR = 98%

  ➔ t_{eff} = 1.18 ns, close to cache itself.

Maximizing Hit Rate

- Key to good performance is to design application code to maximize hit rate.
- One simple rule: always try to access memory contiguously, e.g. in array operations, fastest-changing index should correspond to successive locations in memory.

Good Example

- In FORTRAN:

```
DO J = 1, 1000
     DO I = 1, 1000
          A(I,J) = B(I,J) + C(I,J)
     ENDDO
ENDDO
```

- This references A(1,1), A(2,1), etc. which are stored contiguous in memory.

Bad Example

- This version references A(1,1), A(1,2), …, which are stored 1,000 elements apart. If cache < 4 KB, will cause memory misses:

```
DO I = 1, 1000
     DO J = 1, 1000
          A(I,J) = B(I,J) + C(I,J)
     ENDDO
ENDDO
```

NOTE: C, unlike FORTRAN, stores 2-D array data by column, not by row, so this is a *good example for C!*

I/O Devices

- Transfer information between internal components and external world, e.g. tape drives, disks, monitors.
- Performance measured by “bandwidth”: volume of data per unit time that can be moved into and out of main memory.

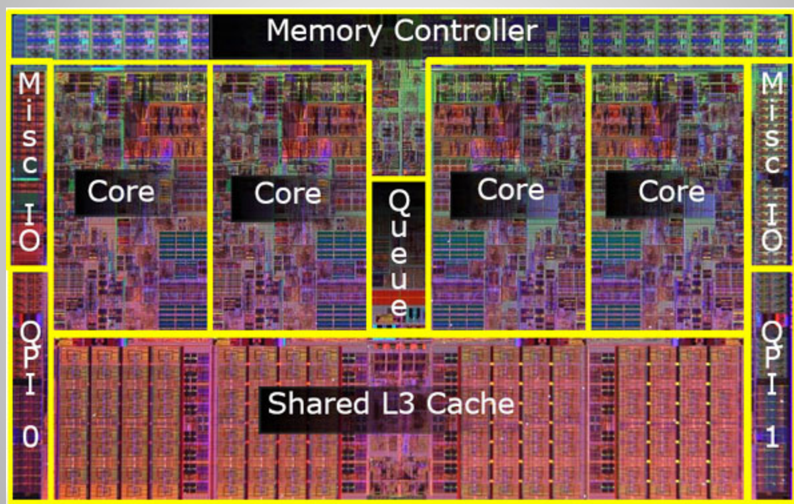
Communication Channels

- Connect internal components.
- Often referred to as a "bus" if just a single channel.
- More complex architectures use "switches".
  – Let any component communicate directly with any other component, but may get "blocking".

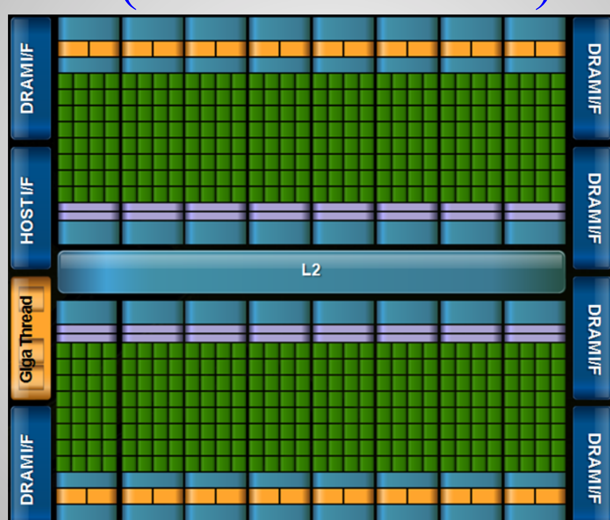
Programming Languages

- Low-level
  - Machine code, Assembly
- High-level
  – Interpreted (portable and easier to implement):
    - Java, Python, Matlab, Mathematica, IDL, etc
  – Compiled (faster, parallelizable to clusters of CPU/GPU):
    - Fortran, C, C++, CUDA (GPU), OpenCL(GPU)
    - Java, Python: bytecode

Example: 4-core CPU (i7)

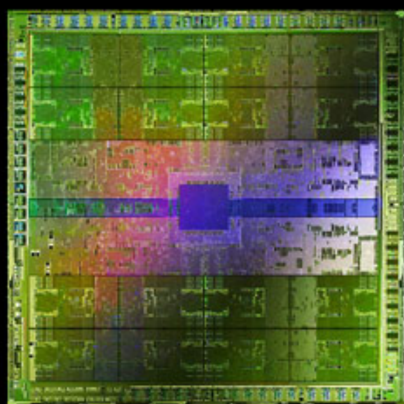


Example: 480-cores GPU (Nvidia GTX 480)



Introducing the 'Fermi' Architecture

The Soul of a Supercomputer in the body of a GPU



- 3 billion transistors
- Over 2x the cores (512 total)
- 8x the peak DP performance
- ECC
- C++

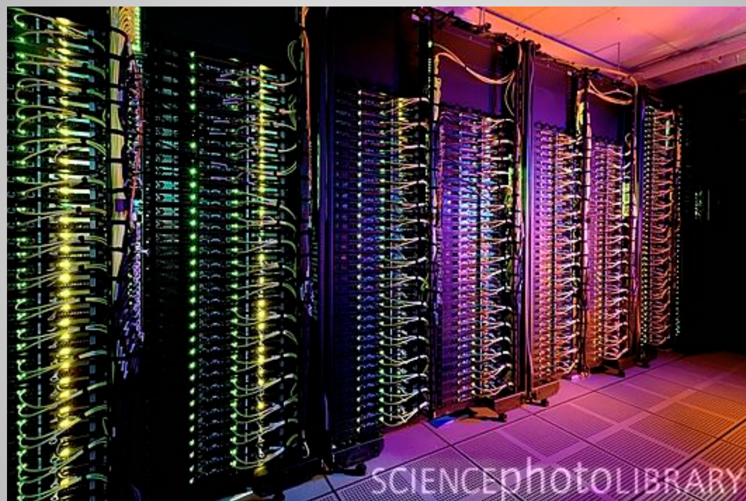
Shared Memory Supercomputers

Blacklight: The SGI® Altix® UV1000 system

Featuring 512 eight-core Intel Xeon 7500 (Nehalem) processors (4,096 cores) with 32 terabytes of memory, Blacklight is partitioned into two connected 16-terabyte coherent shared-memory systems — creating the two largest coherent shared-memory systems in the world.

Blacklight is now available for research through the TeraGrid allocation process

Computer Cluster (distributed memory)



Room containing a cluster of 420 computing nodes, each linked together by a fast speed network. The nodes are able to access a computer storage cluster with three petabytes of storage space. The system allows users to analyze large amounts of data using high speed computing power.

Photographed at the Genome Sciences Center, March 2011.

Parallel Computing

- Clusters: MPI (message passing interface)
  - Communication channel: ethernet, infiniband (speed 10-100 Gbit/s + latency ~200 nsec)
- Shared memory: OpenMP
  - Communication channel: memory bus
    (64 bits x clock~500 Gbit/s + latency ~10 nsec)
- GPU computing: CUDA (Nvidia), OpenCL, OpenHMPP
  - Communication channel: graphic card bus PCI-Xpress (~400 Gbit/s + latency: bottleneck is transfer of memory from CPU to GPU)

Code Optimizations

- Compiler optimization options: -O0,-O1,-O2,-O3
  - Loops optimization, Inlining of functions, etc
  - Compilation time longer and size of the executable larger
- Auto-parallelization (-openmp -parallel for intel compilers)
- Examples:

icc –o progx progrx.c –O3 –openmp –parallel (C compiler)

ifort –o progx progx.f –O3 –openmp –parallel (Fortran compiler)

Debuggers

- Instead of printing vars throughout the code
- Used to catch seg faults, fpe, and other errors
- Unfortunately catching fpe is compiler and OS dependent!! Need to experiment
- Example:

```
>cc -o debug debug.c -g –lm
>gdb ./debug
(gdb) r
```

(Note: -g option needed to load symbols table for debugging)