Advanced Topics in Numerical Analysis: High Performance Computing

MATH-GA 2012.001 & CSCI-GA 2945.001

Georg (Stadler)
Courant Institute, NYU
stadler@cims.nyu.edu

Spring 2017, Thursday, 5:10–7:00PM, WWH #512

Feb. 17, 2017

Outline

Organization issues

Last class summary

Memory hierarchies (single CPU)

Tool/command of the week

Organization issues

- I'm looking for a room/time to makeup for the cancelled classes. Will post options on Piazza–please vote!
- There will be a new homework posted by tomorrow.

Organization issues

- We got access and computing time on Stampede at the Texas Advanced Computing Center (TACC).
- You'll hear from Bill concerning registration. We'll use Stampede for homework problems and you can use it for final projects (more how to use it, in the next classes).
- Currently, Stampede is #17 on the Top 500 list, but in the process of being upgraded to Stampede 2 this summer.
- We share compute time on that resource, so please don't be wasteful.

Outline

Organization issues

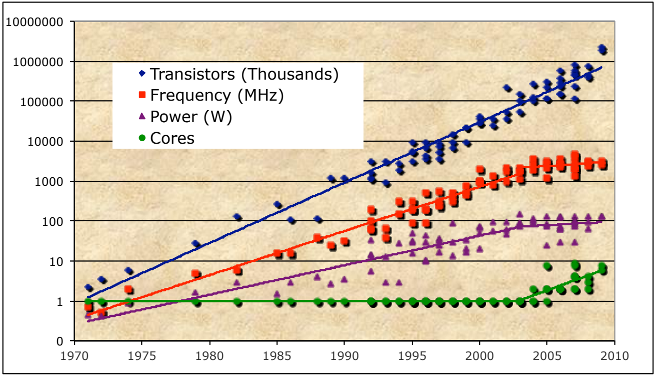
Last class summary

Memory hierarchies (single CPU)

Tool/command of the week

Moore's law today

- Frequency/clock speed stopped growing in ~ 2004
- Number of cores per CPU
- Moore's law still holds
- Energy use ~bounded



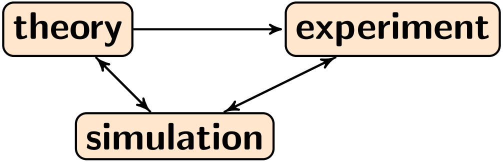
Source: CS Department, UC Berkeley.

Parallel computing $\subset$ high-performance computing

- All major vendors produce multicore chips—need to think differently about applications.
- How well can applications and algorithms exploit parallelism?
- Memory density (DRAM) grows at slower rate.
  Loading/writing to memory is slow ($\mathcal{O}(100)$ clock cycles)
- Top500 list: leading machines have $> 10^6$ processor cores, and often two different kinds of compute chips (CPUs and some kind of accelerators).

Do we really need larger and faster?

Simulation has become the third pillar of Science:



HPC computing used in: weather prediction, climate modeling, drug design, astrophysics, earthquake modeling, semiconductor design, crash test simulations, financial modeling, . . .

Basic CS terms recalled

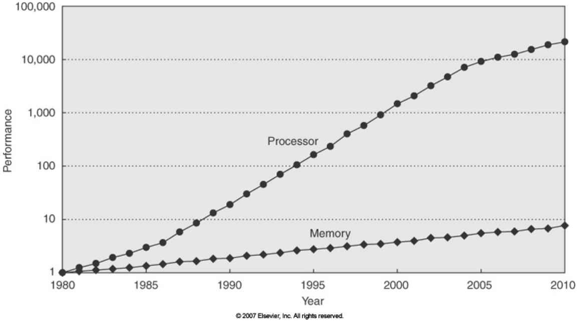
- compiler: translates human code into machine language
- CPU/processor: central processing unit caries out instructions of a computer program, i.e., arithmetic/logical operations, input/output
- core: individual processing unit in a CPU, “multicore” CPU; will sometimes use “processors” in a sloppy way, and actually mean “cores”
- clock rate/frequency: indicator of speed in which instructions are performed
- floating point operation: multiplication add of two floating point numbers, usually double precision (64 bit, about 16 digits)
- peak performance: fastest theoretical flop/s
- sustained performance: flop/s in actual computation
- memory hierarchy: large memories (RAM/disc/solid state) are slow; fast memories (L1/L2/L3 cache) are small

Outline

- Organization issues
- Last class summary
- Memory hierarchies (single CPU)
- Tool/command of the week

Flop/s versus Mop/s

For many practical applications, memory access is the bottleneck, not floating point operations.



Development of memory versus processor performance.

- Most applications run at $< 10\%$ of the theoretical peak performance.
- Mostly a single core issue; on parallel computers, things become even more difficult.

Memory hierarchies

Computer architecture is complicated. We need a basic performance model.

- Processor needs to be “fed” with data to work on.
- Memory access is slow; memory hierarchies help.
- This is a single processor issue, but it’s even more important on parallel computers.

Memory hierarchies

Computer architecture is complicated. We need a basic performance model.

- Processor needs to be "fed" with data to work on.
- Memory access is slow; memory hierarchies help.
- This is a single processor issue, but it's even more important on parallel computers.

More CS terms:

- latency:

Memory hierarchies

Computer architecture is complicated. We need a basic performance model.

- Processor needs to be "fed" with data to work on.
- Memory access is slow; memory hierarchies help.
- This is a single processor issue, but it's even more important on parallel computers.

More CS terms:

- latency: time it takes to load/write data from/at a specific location in RAM to/from the CPU registers (in seconds)

Memory hierarchies

Computer architecture is complicated. We need a basic performance model.

- Processor needs to be "fed" with data to work on.
- Memory access is slow; memory hierarchies help.
- This is a single processor issue, but it's even more important on parallel computers.

More CS terms:

- latency: time it takes to load/write data from/at a specific location in RAM to/from the CPU registers (in seconds)
- bandwidth:

Memory hierarchies

Computer architecture is complicated. We need a basic performance model.

- Processor needs to be "fed" with data to work on.
- Memory access is slow; memory hierarchies help.
- This is a single processor issue, but it's even more important on parallel computers.

More CS terms:

- latency: time it takes to load/write data from/at a specific location in RAM to/from the CPU registers (in seconds)
- bandwidth: rate at which data can be read/written (for large data); in (bytes/second);

Memory hierarchies

Computer architecture is complicated. We need a basic performance model.

- Processor needs to be "fed" with data to work on.
- Memory access is slow; memory hierarchies help.
- This is a single processor issue, but it's even more important on parallel computers.

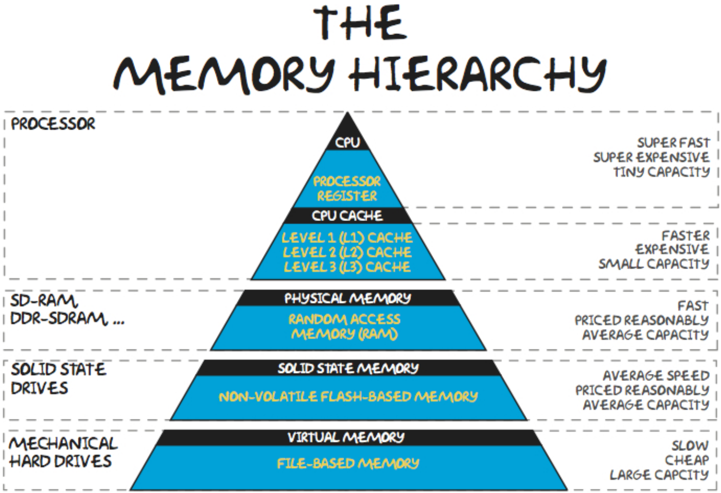
More CS terms:

- latency: time it takes to load/write data from/at a specific location in RAM to/from the CPU registers (in seconds)
- bandwidth: rate at which data can be read/written (for large data); in (bytes/second);

Bandwidth grows faster than latency.

Memory hierarchies

On my Mac Book Pro: 32KB L1 Cache, 256KB L2 Cache, 3MB Cache, 8GB RAM



CPU: $\mathcal{O}$(1ns), L2/L3: $\mathcal{O}$(10ns), RAM: $\mathcal{O}$(100ns), disc: $\mathcal{O}$(10ms)

Memory hierarchies

Decreasing memory latency

- Eliminate memory operations by saving data in fast memory and reusing them, i.e., temporal locality: Access an item that was previously accessed
- Explore bandwidth by moving a chunk of data into the fast memory: spatial locality: Access data nearby previous accesses
- Overlap computation and memory access (pre-fetching; mostly figured out by compiler, but the compiler often needs help)

Memory hierarchies

Decreasing memory latency

- Eliminate memory operations by saving data in fast memory and reusing them, i.e., temporal locality: Access an item that was previously accessed
- Explore bandwidth by moving a chunk of data into the fast memory: spatial locality: Access data nearby previous accesses
- Overlap computation and memory access (pre-fetching; mostly figured out by compiler, but the compiler often needs help)

More CS terms:

- cache-hit:

Memory hierarchies

Decreasing memory latency

- Eliminate memory operations by saving data in fast memory and reusing them, i.e., temporal locality: Access an item that was previously accessed
- Explore bandwidth by moving a chunk of data into the fast memory: spatial locality: Access data nearby previous accesses
- Overlap computation and memory access (pre-fetching; mostly figured out by compiler, but the compiler often needs help)

More CS terms:

- cache-hit: required data is available in cache $\Rightarrow$ fast access

Memory hierarchies

Decreasing memory latency

- Eliminate memory operations by saving data in fast memory and reusing them, i.e., temporal locality: Access an item that was previously accessed
- Explore bandwidth by moving a chunk of data into the fast memory: spatial locality: Access data nearby previous accesses
- Overlap computation and memory access (pre-fetching; mostly figured out by compiler, but the compiler often needs help)

More CS terms:

- cache-hit: required data is available in cache $\Rightarrow$ fast access
- cache-miss:

Memory hierarchies

Decreasing memory latency

- Eliminate memory operations by saving data in fast memory and reusing them, i.e., temporal locality: Access an item that was previously accessed
- Explore bandwidth by moving a chunk of data into the fast memory: spatial locality: Access data nearby previous accesses
- Overlap computation and memory access (pre-fetching; mostly figured out by compiler, but the compiler often needs help)

More CS terms:

- cache-hit: required data is available in cache $\Rightarrow$ fast access
- cache-miss: required data is not in cache and must be loaded from main memory (RAM) $\Rightarrow$ slow access

Memory hierarchy

Simple model

1. Only consider two levels in hierarchy, fast (cache) and slow (RAM) memory
2. All data is initially in slow memory
3. Simplifications:
    - Ignore that memory access and arithmetic operations can happen at the same time
    - assume time for access to fast memory is 0
4. Computational intensity: flops per slow memory access

$$q = \frac{f}{m}, \text{ where } f \ldots \#\text{flops}, m \ldots \#\text{slow memop.}$$

Actual compute time:

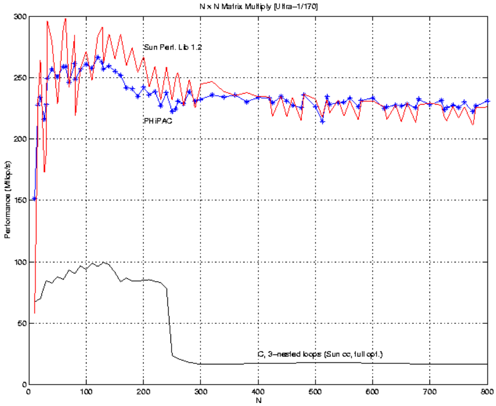
$$ft_f + mt_m = ft_f(1 + \frac{t_m}{t_f}\frac{1}{q}),$$

where t_f is time per flop, and t_m the time per slow memory access.
Computational intensity should be as large as possible.

Memory hierarchy

Example: Matrix-matrix multiply

Comparison between naive and blocked optimized matrix-matrix multiplication for different matrix sizes.



Comparison between optimized and naive matrix-matrix multiplication on old hardware with peak of 330MFlops.
Source: J. Demel, Berkely

BLAS: Optimized Basic Linear Algebra Subprograms

Memory hierarchy

To summarize:

- Temporal and spatial locality is key for fast performance.
- Simple performance model: fast and slow memory; only counts loads into fast memory; computational intensity should be high.
- Since arithmetic is cheap compared to memory access, one can consider making extra flops if it reduces the memory access.
- In distributed-memory parallel computations, the memory hierarchy is extended to data stored on other processors, which is only available through communication over the network.

https://github.com/NYU-HPC17/lecture2

Outline

Organization issues

Last class summary

Memory hierarchies (single CPU)

Tool/command of the week

The module command

Allows to switch the user environment (programs, compilers etc). Available on all UNIX-based systems, i.e., on CIMS computers, compute servers etc.

```
module list
module avail ...
module load python/3.4
module unload texlive-2016
module whatis gcc-6.1.0
```