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

March 9, 2017

Outline

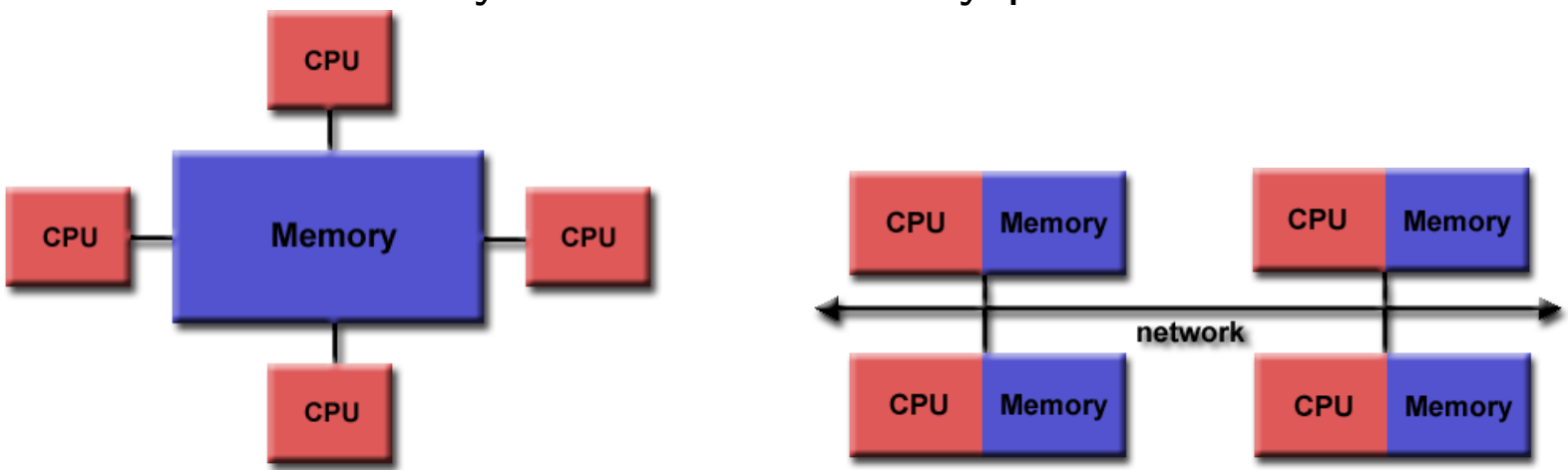
Organization issues

- OpenMP homework! This time for real. . . Homework hand in through a git repository. The repo should contain the code as well as a description of the results (either as TXT or TEX file). Also, include a Makefile
- Final projects! I've a list of proposed projects that I will post. I would like to (more or less) finalize projects during the next 2 weeks. Please find me (end of) next week if you want to discuss a project idea. Final projects are in groups of 1-2 people (2 preferred, 3 possible if needed for project)
- Final project presentations (max 10min each) in the week May 8–12.

Outline

Programming models

- Flynn's taxonomy:
  - Single instruction–single data (SISD)
  - Single instruction–multiple data (SIMD)
  - Multiple instruction–multiple data (MIMD)
- Distributed memory vs. shared memory parallelism



- Programming models: OpenMP vs. Message passing interface (MPI); and combinations thereof

Process vs. thread, stack vs. heap

- A process is an independent execution unit, which contains their own state information (pointers to instruction and stack). One process can contain several threads.
- Threads within a process share the same address space, and communicate directly using shared variables. Seperate stack but shared heap memory.
- Stack memory: Used for temporarily storing data; fast; last-in-first-out principle. Examples `int a=2;` `double b=2.11;` etc; no deallocation necessary; small size; static.
- Heap memory: Not managed automatically, manually allocate/de-allocate/re-allocate; slower; larger;

Outline

Debugging/profiling tools

- `printf` — fishing for bug, but sometimes useful
- `gdb` — GNU debugger: serial, but can also be attached to a parallel task
- Totalview (commercial debugger/profiler, available on many HPC resources)
- DDT (another commercial debugger/profiler; available on Stampede)
- `TAU`: Tuning and Analysis Utility
- `PAPI`: Performance Application Programming Interface
- `HPCtoolkit/perfexpert`: Suite of (open source) analysis and profiling tools
- `valgrind` (say: "val-grinned") and `cachegrind`: memory/cache debugger and profiler

Valgrind and cachegrind

Valgrind

- memory management tool and suite for debugging, also in parallel
- profiles heap (not stack) memory access
- simulates a CPU in software
- running code with valgrind makes it slower by factor of 10-100
- not installed by default on only available on Mac OS; use for MPI-parallel debugging on Mac limited
- Documentation: http://valgrind.org/docs/manual/

memcheck	cachegrind	callgrind
finds leaks	cache profiler	extension to
inval. mem. access	sources of cache	cachegrind
uninitialize mem.	misses	function call graph
incorrect mem. frees		

Valgrind and cachegrind

Usage (see examples):

Run with valgrind (no recompile necessary!)

```
mpirun -np 2 valgrind --tool=memcheck [options]
./a.out [args]
```

Test examples for valgrind memcheck:

https://github.com/NYU-HPC17/lecture6

Valgrind and cachegrind

Run cachegrind profiler:
`valgrind --tool=cachegrind [options] ./a.out [args]`

Visualize results of cachegrind:
`cg_annotate --auto=yes cachegrind.out***`

To illustrate the use of cachegrind, we used the vector multiplication problem:

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

```
valgrind --tool=cachegrind
        ./inner-mem vec_size no_of_reps skip
```

Outline

Introduction to MPI

Use B. Gropp's PPT slides

https://github.com/NYU-HPC15/lecture6