A Free Online Textbook and Interactive Exercises for Introducing Parallel Computing

Suzanne J. Matthews

U.S. Military Academy
West Point, NY USA

Tia Newhall

Swarthmore College
Swarthmore, PA USA

Kevin C. Webb

Swarthmore College
Swarthmore, PA USA







diveintosystems.org

The opinions expressed in this presentation are solely of the authors and do not necessarily reflect those of the U.S. Military Academy, the DoD or the U.S. Army.

Dive into Systems: free online textbook & exercises

- Introduces large set of systems & parallel computing topics
  - assumes only CS1 background of reader
  - primary intro systems textbook
  - aux reference for upper-level courses
  - Community involvement in development
  - 70+ institutions currently using
- diveintosystems.org

15 Chapters:

1. C intro
2. C depth
3. C Debugging
12. Code Optimization

6-10. Assembly
x86-64, ARM 64, IA32
(RISC-V in development)

Appendix: Using Unix

4. Binary Representation
5. Architecture
11. Memory Hierarchy

13. Operating Systems

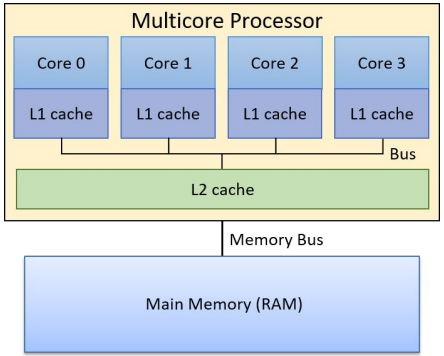
14. Shared Memory Parallel
15. Other Parallel

Chapter 14: Shared Memory Parallelism

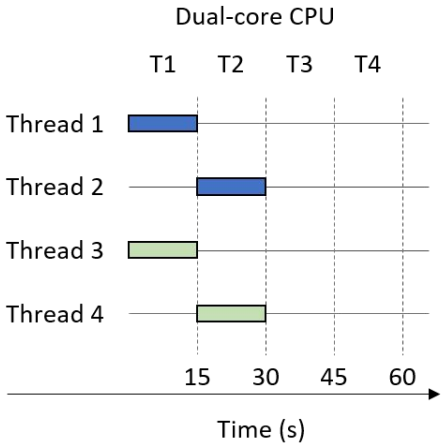
Main focus, as first introduction to parallelism

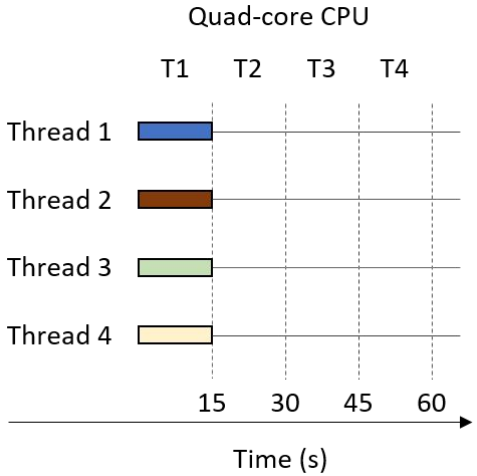
- single focus to go more in depth
- develop parallel thinking

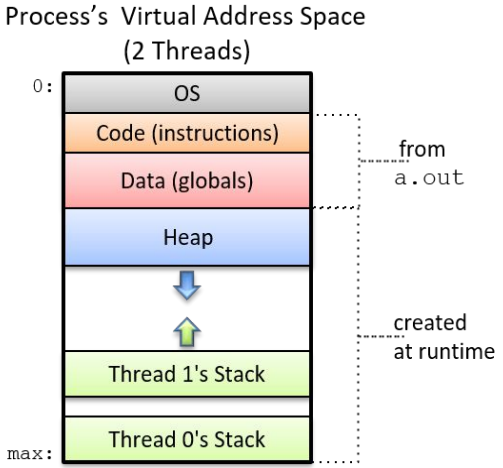
Multicore



Multithreaded Parallel Execution







POSIX Thread Programming

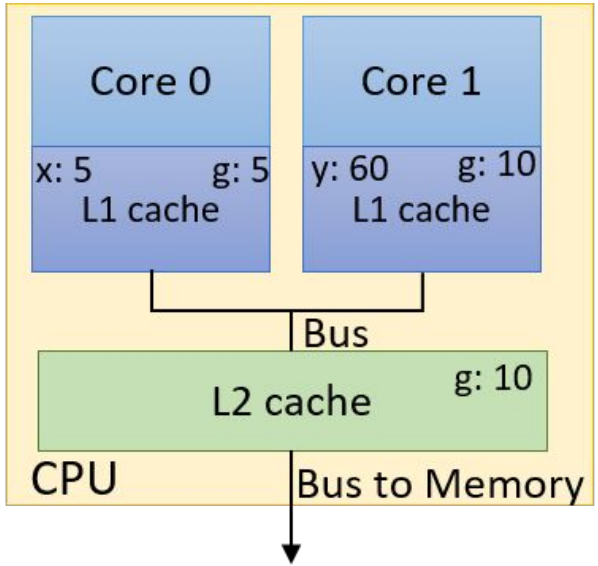
- Explicit Parallel Programming
  - develops parallel thinking skills
- pthread create/join, thread main function
- Race Conditions
- Synchronization
  - mutex
  - semaphore, barrier, condition variables (producer-consumer)
- Efficient programming with mutex
  - critical section placement, space/time: thread local vs. shared
- Deadlock

We revisit 2 examples to illustrate concepts:

1. Count Sort
2. Scalar Vector Multiply

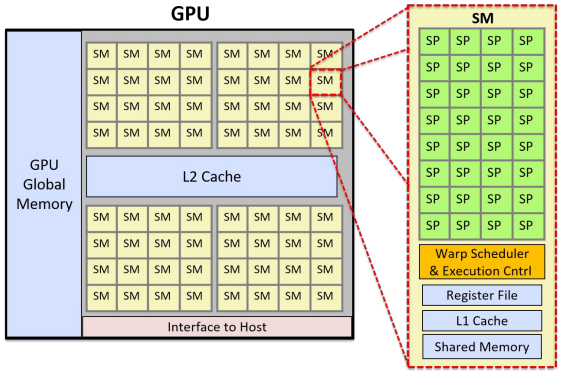
Also:

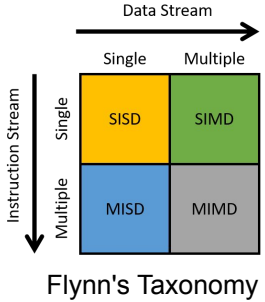
- Parallel Performance
  - Speedup, Scalability, Efficiency
    Amdahl's law, Gustafson-Barsis law
- Cache Coherency & False Sharing
- Implicit threading with OpenMP



Chapt. 15: Looking Ahead Other Parallel Paradigms

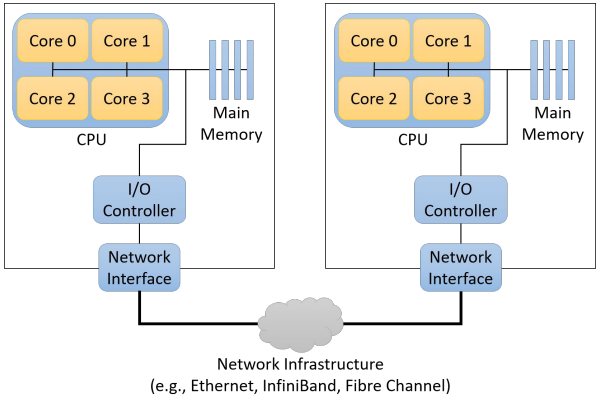
Accelerators/CUDA



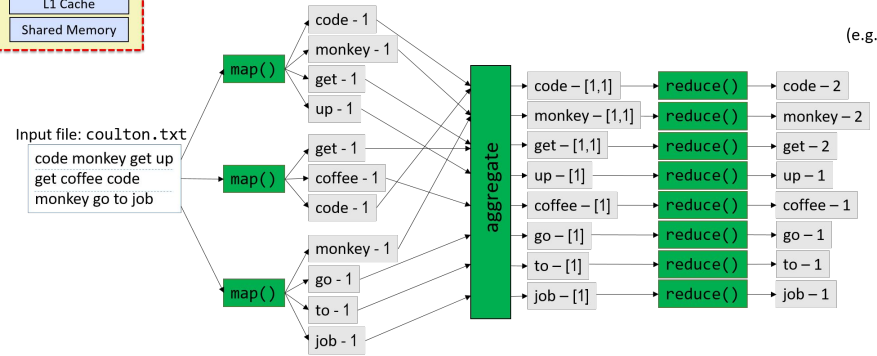


Flynn's Taxonomy

Distributed Memory/MPI

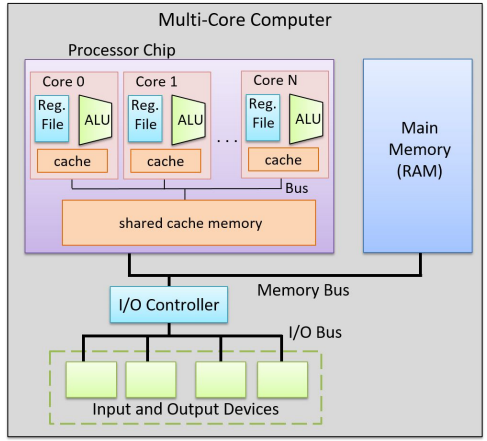


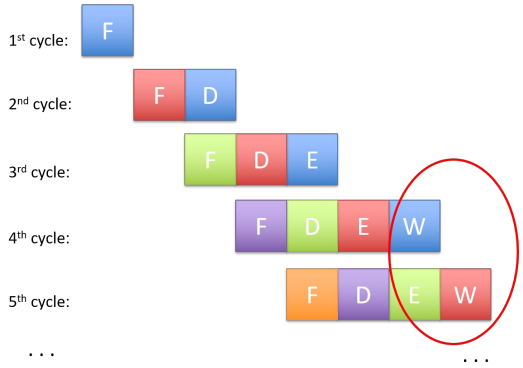
Cloud/ Map Reduce

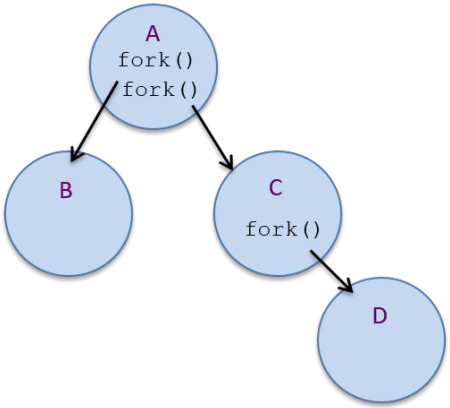


Other:

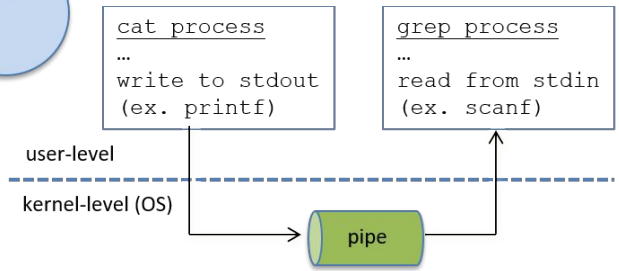
Chapt. 5 Architecture

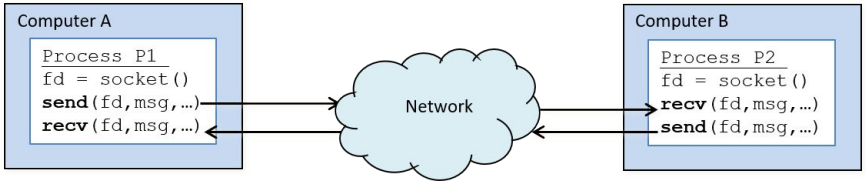




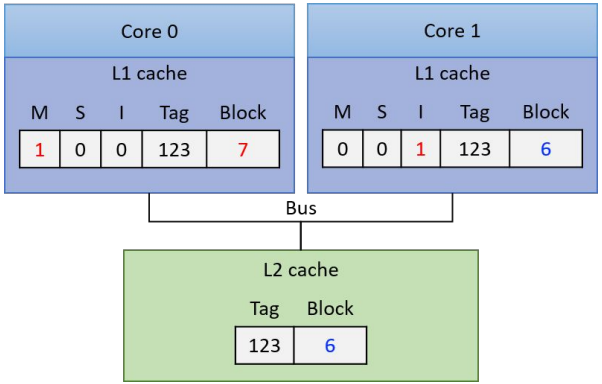


Chapt. 13 OS





Chapt. 11 Caching

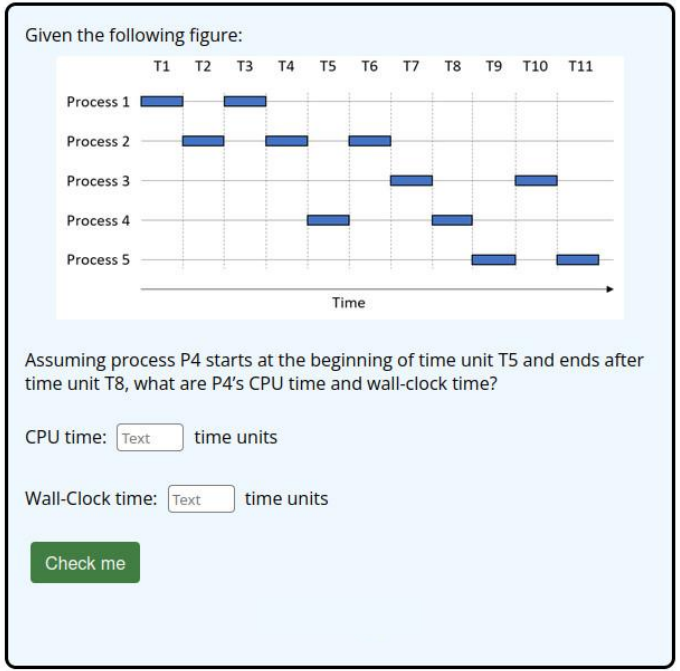


Interactive Exercises w/ each Chapter

Use Runestone: matching, fill in blank, parsons, …

Questions developed by participants in our 4 year NSF-Sponsored Exercise Developers Program

- Groups of Faculty from other institutions work together developing questions for chapters
  - Wide range of input, fit goal of our book being widely applicable



Ask me Another Components

Generate "infinite" practice problems

"Generate Another" button

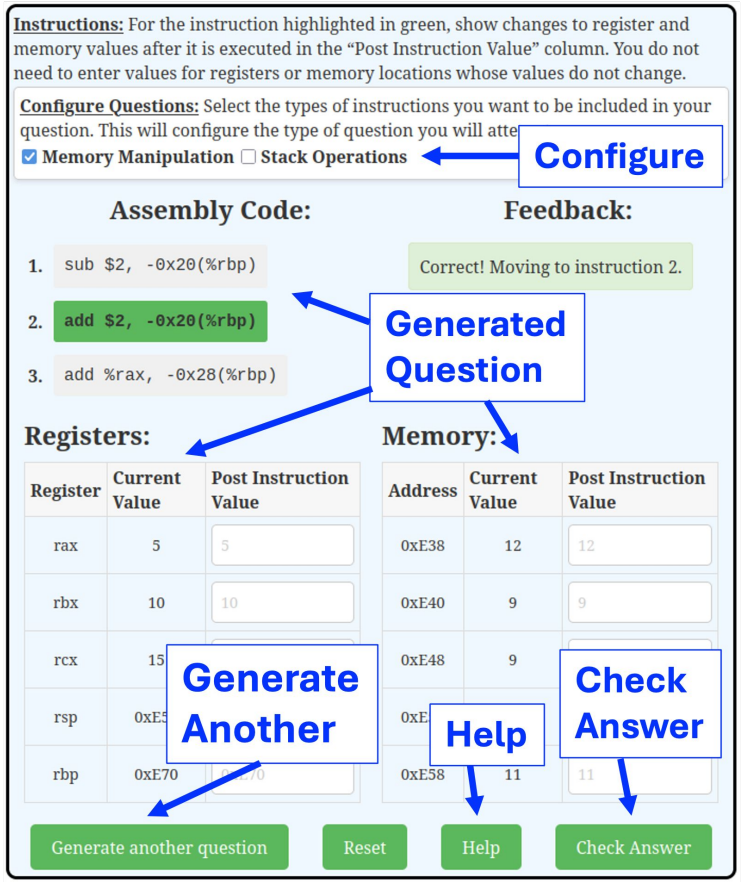
Use heuristics to generate good questions

- not randomization only
- not fixed pool of questions

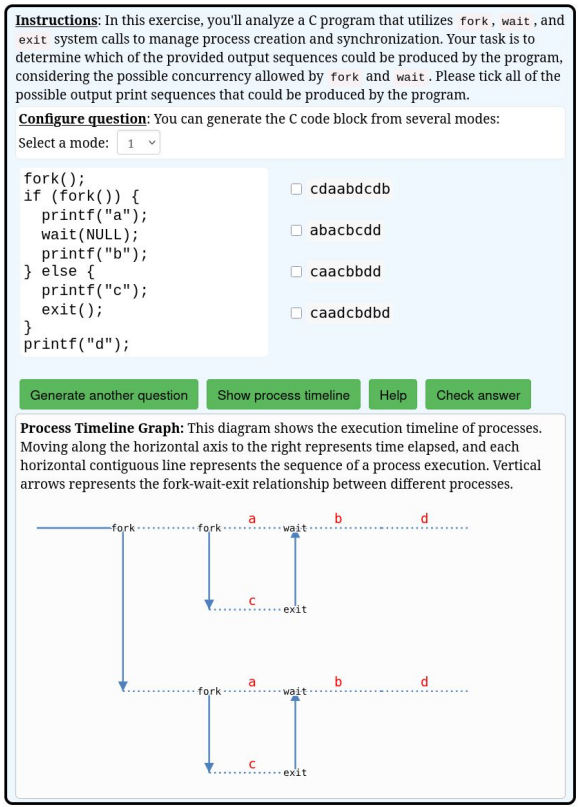
Each custom, but common look-and-feel

Developed with Swarthmore Students

34 different ones in Textbook Exercises



2 Processes & Concurrency



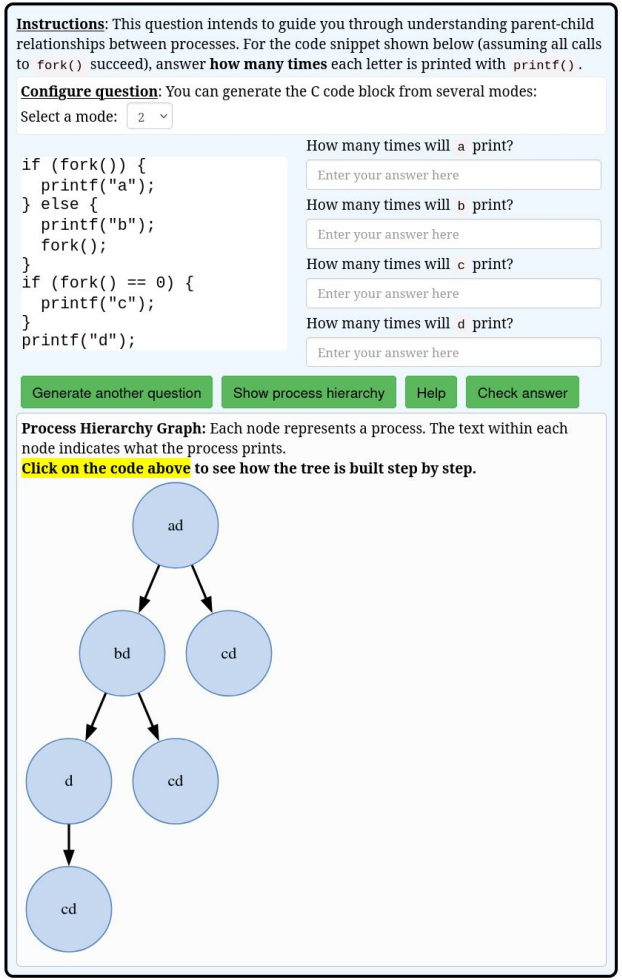
Generates C code w/fork & printf calls (one adds exit, wait)

Questions about concurrent actions

- process creation
- parent/child relationships
- concurrent execution
- synchronization (exit, wait)

Help Visualizations

- process hierarchy
- process timeline



2 Thread Parallelism & Race Conditions

Generate thread main code for 2 threads and ask questions about parallel execution. One adds mutex lock/unlock.

- ID race conditions with shared var accesses
- ID synchronization with mutex lock and unlock calls

Config: difficulty level, answer mode

Instructions: Given the global variables and code for two threads below, select all possible values of the variables after the threads execute in parallel.

Configure question:

Select a mode: 1

- Mode 1: Each if/else body contains one expression.
- Mode 2: If/else bodies can contain two expressions.

Please select what type of question to generate: Multiple Choice

Shared Global Variables:

```
int x = 6;
int pthread_mutex_lock(pthread_mutex_t *mutex);
```

Thread 1:

```
int y = 4;
if (y >= x) {
     y = y + 8;
} else {
     pthread_mutex_lock(&mutex);
     x = x - 8;
     pthread_mutex_unlock(&mutex);
}
print("%d %d", x, y);
return NULL;
```

Thread 2:

```
int y = 4;
if (y >= x) {
     y = y + 8;
} else {
     pthread_mutex_lock(&mutex);
     x = x - 8;
     pthread_mutex_unlock(&mutex);
}
print("%d %d", x, y);
return NULL;
```

Select all the possible final states:

	x	y (thread1)	y (thread2)
☐	-2	4	12
☐	-2	12	4
☐	-10	4	4
☐	-10	4	12

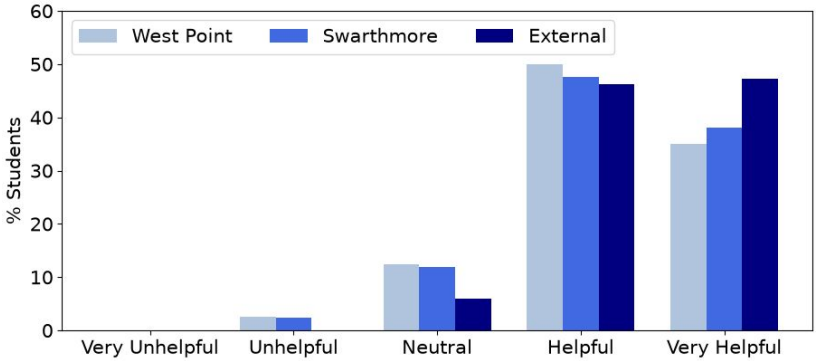
Generate another question | Check answer

Pilot Study: Process Ask me Another

15 institutions: Exercise using Component and Post Survey

Students (*n* = 298):

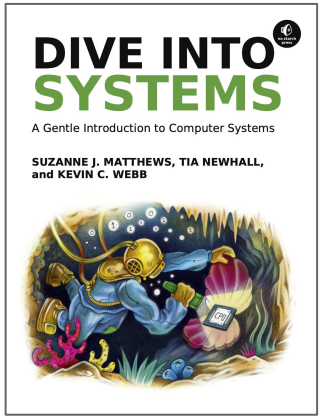
- > 85% found helpful or very helpful
  - \+ interactive process hierarchy
  - \+ "generate another" functionality



Faculty (*n* = 11):

- most felt matched well or extremely well to their focus
- appreciated "generate another" as more non-trivial practice
- felt that it really stress understanding fork and concurrent processes

Also low-cost print version of textbook
will always remain free online



Thank you!

Questions?

`diveintosystems.org`

Textbook, interactive exercises, other resources, mailing list

Upcoming Events:

SIGCSE'27: Affiliated Event & "Ask me Another" paper