Electrical & Computer ENGINEERING

How to Write Fast Code

18-645, spring 2008

21^{st} Lecture, Apr 2^{nd}

Instructor: Markus Püschel

TAs: Srinivas Chellappa (Vas) and Frédéric de Mesmay (Fred)

Previous Lecture

- **Parallelism is the future**
- **Extracting/using parallelism: ongoing challenge**
  - Hardware is ahead of software:
  - Producing parallel hardware currently easier than producing parallelized software
- **"Our industry has bet its future on parallelism(!)"**

  \- David Patterson, UC Berkeley
- **Challenge: how to "map" a given problem to a parallel architecture/platform**

Overview

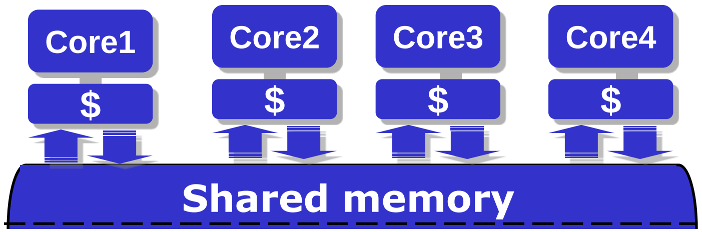
- **Parallelizing: case studies**
  - MMM
  - WHT
- **SMP programming with OpenMP**
  - Useful for your projects
  - In-class demo
- **Admin stuff**
  - Check project meeting schedule

Parallelizing a Problem

- **(Blackboard)**
  - MMM
  - WHT

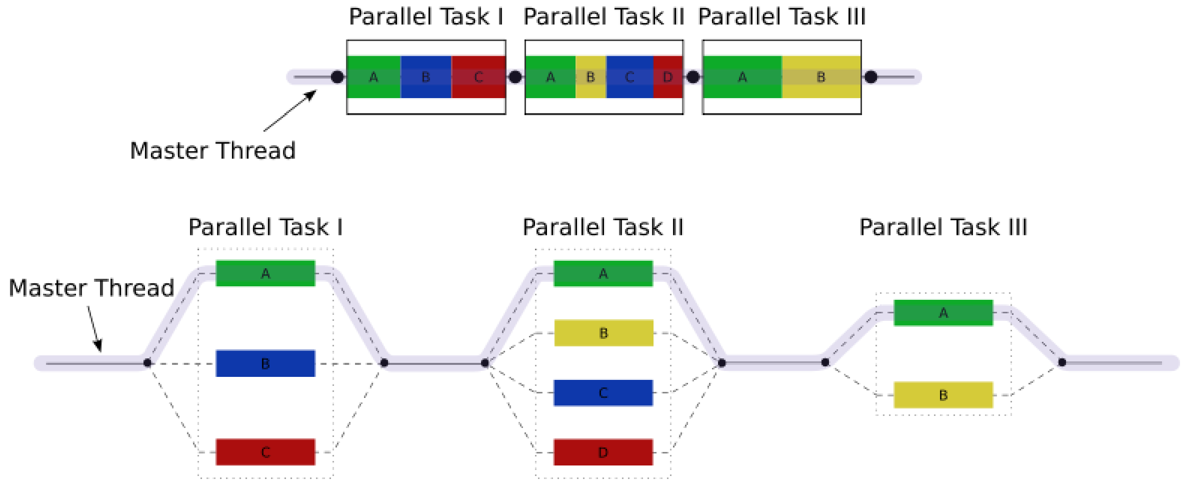
- **Take-away ideas**
  - Data parallel partitioning
  - Boils down to: partitioning work in a load-balanced manner among the processors
  - Might be able to express parallelism in mathematical constructs
  - Important considerations:
    - Minimize data transfer among processors
    - Minimize barriers / synchronization
    - Big SMP issue: false sharing

SMP – A Refresher



- **SMP (symmetric multiprocessing): smaller CPUs**
  - Multi-core, Multi-CPU, Hybrids, FPGAs etc.
- **The good:**
  - Easy to program
- **The bad:**
  - System complexity is pushed to hardware design
  - Bottleneck: contention to shared resource (memory)
  - Coherency protocols – difficult to implement, expensive
  - Scalability is an issue

Designing Parallel Programs



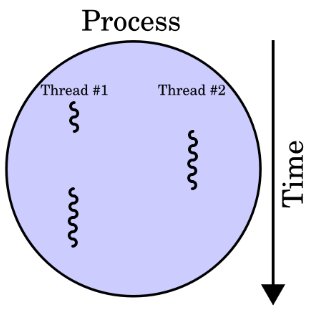
- **Central idea: expose parallelism inherent in the problem by splitting it into independent tasks**
- **Might have one or more split/converge stages**

Multiprocessing: primitives

- **Task/thread creation and scheduling**
  - (spawn/fork/exec)
- **Data exchange**
  - Threads/SMP: trivial, since memory space is shared
  - MPI: send/receive explicitly
- **Task synchronization (barriers/fences)**
  - Why?
  - Critical sections, mutexes, semaphores
    - Hardware support (for correctness, performance)
  - Barriers

Multithreading

- **Process: computer program that is being executed**
- **Thread: a program can split into multiple simultaneously executing tasks called threads**

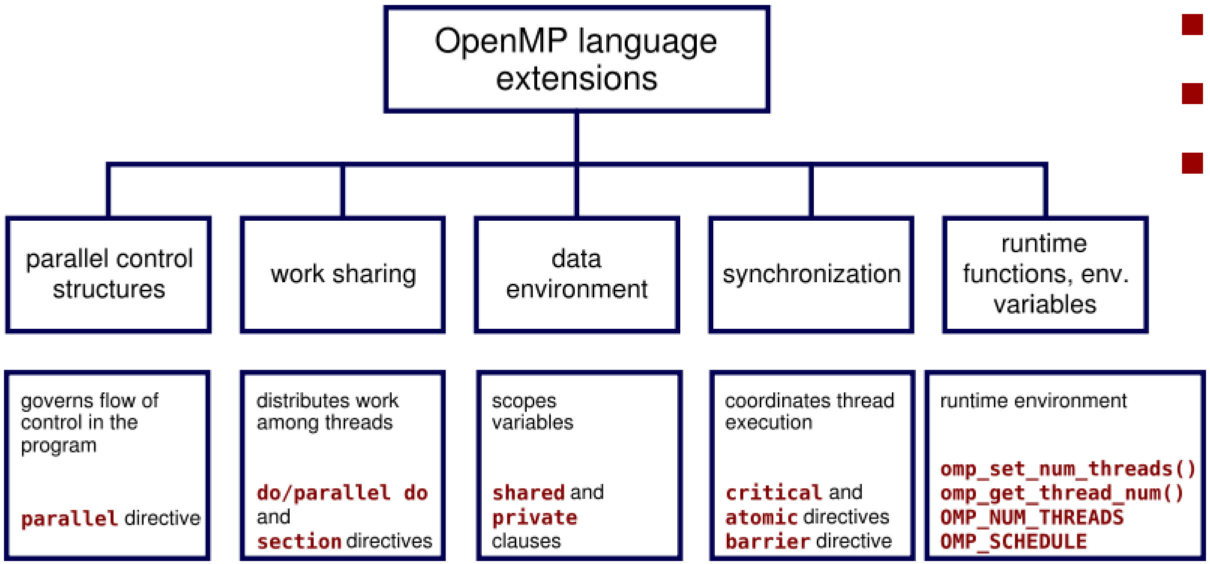


- **Why use threads?**
  - Logical partitioning of tasks
  - Current execution
  - Lightweight (compared to multiple processes)
  - Can share/sync with other threads in the process easily
  - Important: threads can be scheduled concurrently on multiple CPUs/cores

Pthreads / MPI

- **How does one do multiprocessing?**
  - Can do this manually
  - But libraries exist
- **Message passing (best for distributed/cluster)**
  - Computers in a cluster can use MPI to communicate
  - How is it used
- **Pthreads library (best for SMP)**
  - Standard API for creating and manipulating threads
  - C types, and C function calls
  - Fine-grained control of parallel programs
- **If you need only a subset…use OpenMP**
  - Good for parallelizing most numerical problems

OpenMP: Fundamentals



- **Parallel section**
- **Parallel loop**
- **Barrier/fence/sync**

- **What is it?**
  - API for programming multi-platform SMP in C/C++
- **Why use it: because it's easy!**
  - Much easier to use than Pthreads (tradeoff: power)

OpenMP: Demo

- **Reminder: What is our goal (in this lecture/class)?**
  - Map numerical code to multi-core chip
  - Reminder: what kind of parallelism? (Mostly data parallel)
  - Reminder: example parallel math construct?
- **How can we use OpenMP to achieve what we want?**
- **Compiling:**
  - Need OpenMP compiler (icc, gcc 4.2+)
  - #include <omp.h>
- **(Demo)**

Pitfalls

- **Minimize barriers**
  - Expensive on many systems
- **Minimize contention**
  - Read sharing
  - Write sharing
- **Cache coherence: big SMP issue**
  - Why cache coherence?
  - Manifestation: false sharing

Summary

- **Parallelized MMM, WHT**
- **SMP programming with OpenMP**
  - Use this in your projects!
- **Admin stuff: project meetings**

Meetings Apr 7 (next Monday)

Markus	
11 – 11:45	13
11:45 – 12:30	14
1:30 – 2:15	9
2:15 - 3	16
3 – 3:45	8
3:45-4:30	12
4:30 – 5:15	6
5:15 – 6	7

Fred	
3:45 – 4:30	3
4:30 – 5:15	1
5:15 – 6	2

Franz	
1 – 1:45	?
2 – 2:45	?
4:30 – 5:15	?

Vas	
3:45 – 4:30	4
4:30 – 5:15	10
5:15 - 6	15