Concepts and Models of Parallel and Data-centric Programming

Shared Memory I

Lecture, Summer 2020

Dr. Christian Terboven <terboven@itc.rwth-aachen.de>

i12 High Performance Computing

RWTH AACHEN UNIVERSITY

Outline

0. Organization
1. Foundations
2. **Shared Memory**
3. GPU Programming
4. Bulk-Synchronous Parallelism
5. Message Passing
6. Distributed Shared Memory
7. Parallel Algorithms
8. Parallel I/O
9. MapReduce
10. Apache Spark

a. Processes and Threads
b. Threading in C++
c. RAII idiom, Move Semantics
d. Mutual Exclusion
e. Condition Variable
f. Example: Queue

Shared Memory

Foundations

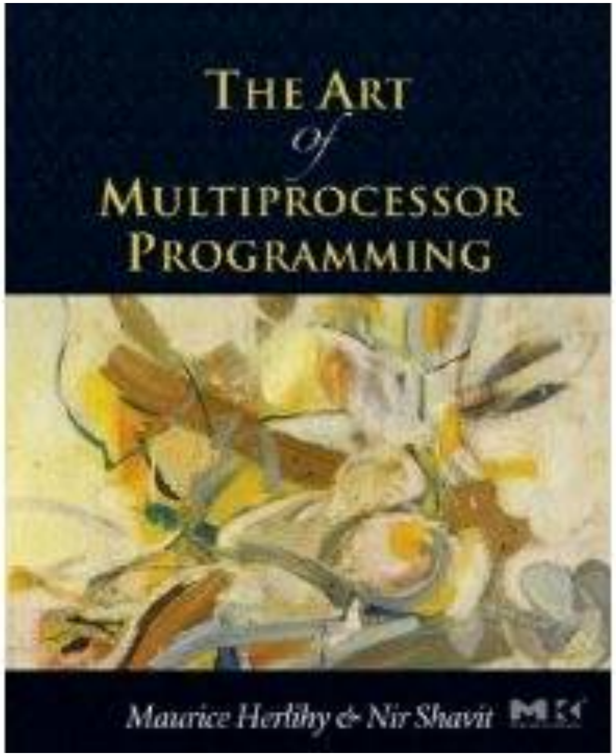
What is this chapter about?

- Important Concepts of Threading and Vectorization …
  - Processes and Threads and Tasks
  - Thread Management
  - Mutual Exclusion and Locking
  - Granularity of Synchronization
  - Vectorization for SIMD architectures

- … illustrated with the C++ Threading Model
  - Translatable to Java, OpenMP and other models
  - Problem-oriented approach: ADT for use in parallel

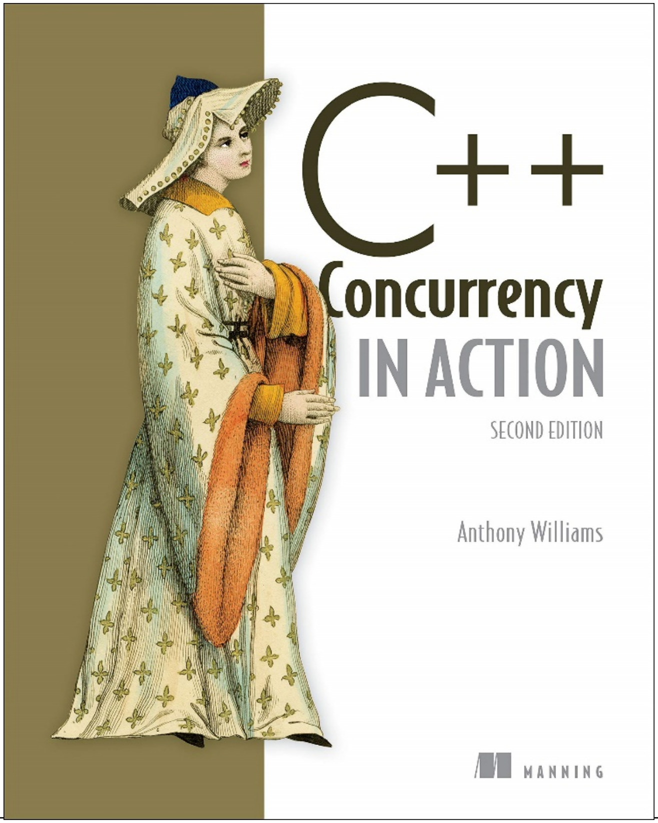
Book / 1

- Java-based examples – if any are left – and some illustrations used in this chapter are taken from the book The Art of Multiprocessor Programming
  - Maurice Herlihy & Nir Shavit


THE ART of MULTIPROCESSOR PROGRAMMING
Maurice Herlihy & Nir Shavit
MK

Book / 2

- C++-based examples used in this chapter are taken from the book C++ Concurrency in Action
  - Anthony Williams

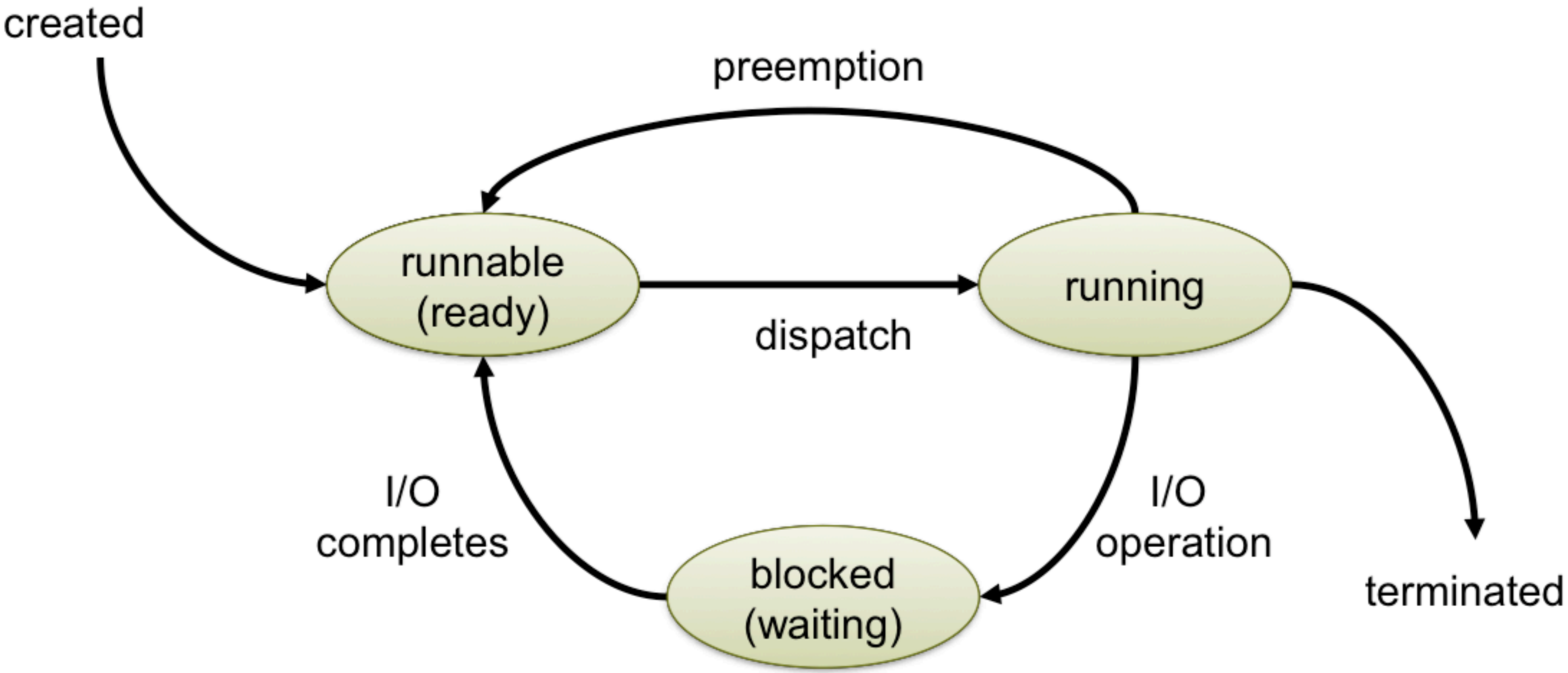


Processes and Threads

Processes

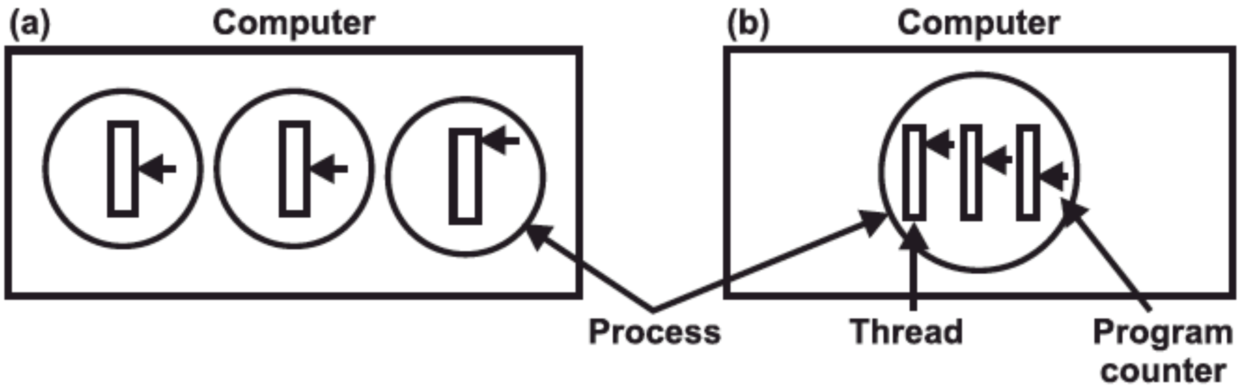
- A *process* is the execution of a program with restricted rights.
  - Computer System := Kernel + Processes
- A process consists of:
  - Virtual processor
    - Address space
    - Instruction pointer / program counter
  - Object code (the actual instructions)
  - Data (static, heap, stack)
  - Operation evironment:
    - Open files and sockets
    - CPU share, privileges, etc.

Process Lifecycle



(Figure by Hoefler)

Processes and Threads



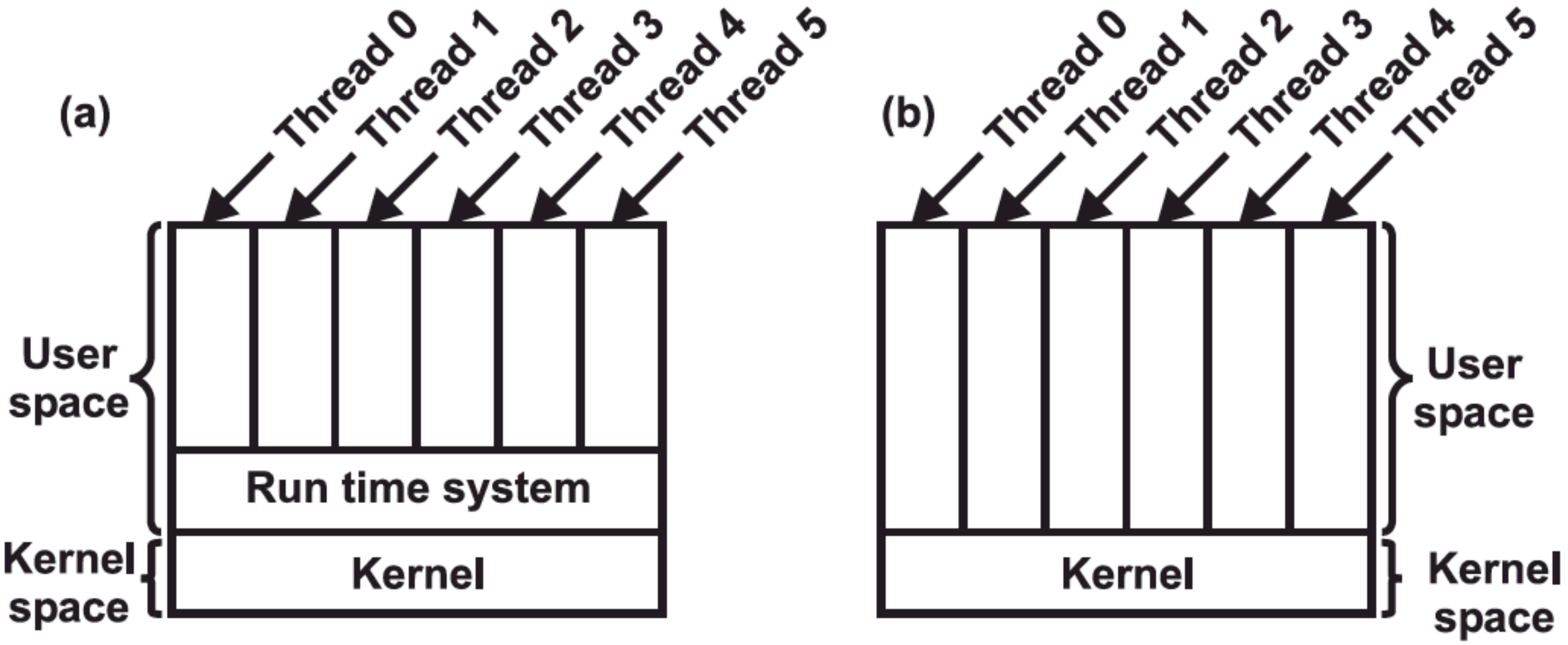
(a) Three processes with one thread each.
(b) One process with three threads.

Per thread items
Program counter Stack Register set Child threads State

Per process items
Address space Global variables Open files Child processes Timers Signals Semaphores Accounting information

(Figure by Tannenbaum and Bos)

Kinds of Threads



- User-level Threads
  - Operating System does not know about these threads
  - Model not popular anymore
- Kernel-level Threads
  - Operation System schedules threads in same way as proc.
  - Application profits from multi-core

(Figure by Tannenbaum and Bos)