Overview

Concurrency Models

Yanqiao ZHU
Project Future, Google Camp
SSE, Tongji Univ.

Course Design

- Objectives: Concurrency programming **theories** and **practices**
- Lecturer: Yanqiao ZHU
  - Research interests:
    - Data Mining and Data Analysis
    - Machine Learning
    - Concurrency and Distributed Systems
- Teaching language: Chinese lectures, with English presentation slides
- No homework, but necessary preparation needed.

Prerequisites on Programming Languages

- Code examples will be drawn from a number of programming languages.
  - Java, C++, Clojure (Haskell), Elixir (Erlang), Python, Go, possibly more…
  - To make use of a specific concurrency framework, it requires a specific language.
- Refer to tutorials of different programming languages in advance.

Concurrent or Parallel?

- When we execute a program, we create a *process*.
  - A *sequential program* has a **single** thread of control.
  - A *concurrent program* has **multiple** threads of control.
- A single computer can have multiple processes **running at once**.
  - If that machine has a single processor, then **the illusion of multiple processes running at once** is just that: **an illusion**.
  - That illusion is maintained by the operating system; it coordinates access to the single processor by the various processes; only one process runs at a time.
- If a machine has more than a single processor, then true parallelism can occur.
  - You can have N processes running simultaneously on a machine that has N processors.

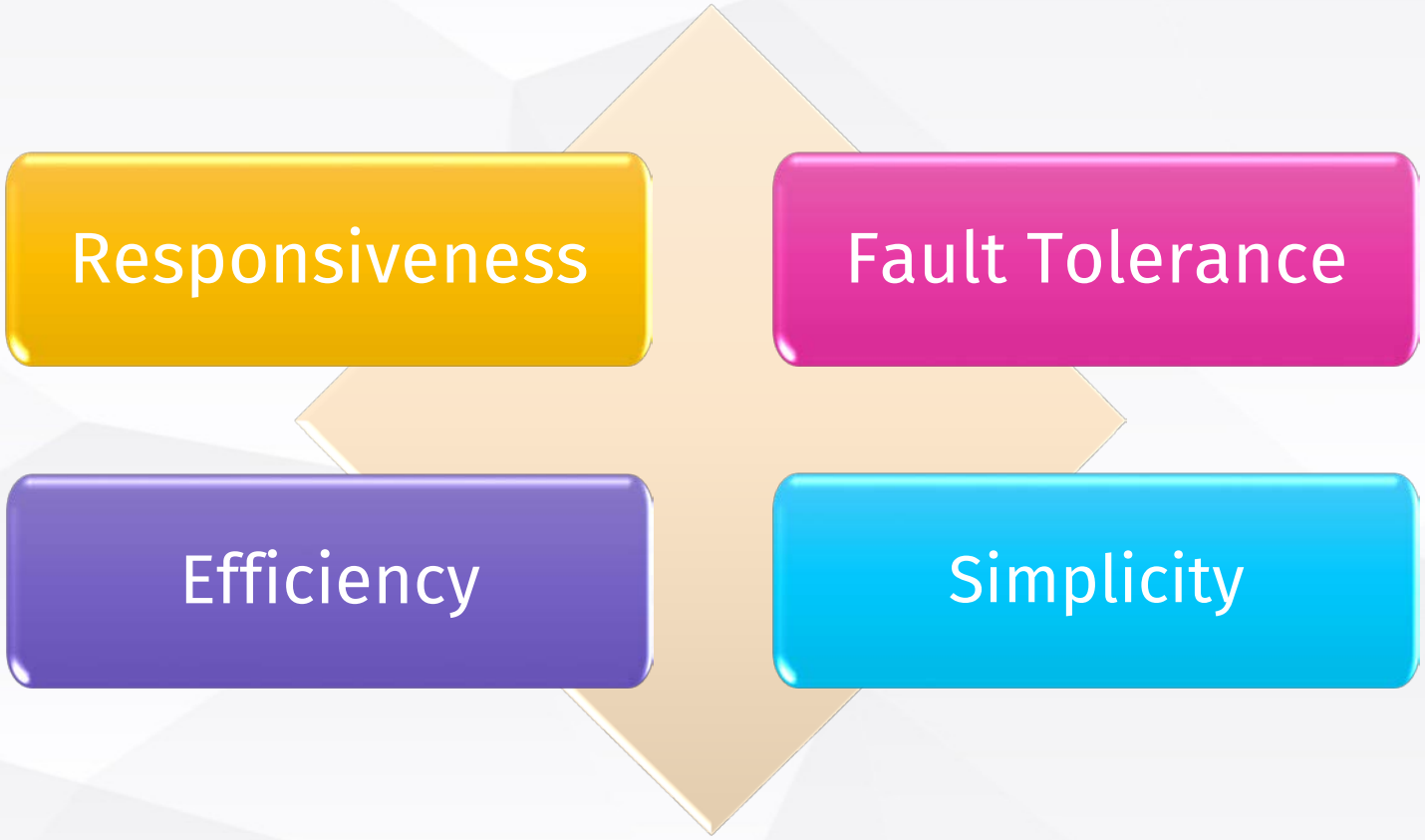
Concurrent or Parallel? (cont.)

- A *concurrent* program has multiple logical threads of control. These threads may or may not run in parallel.
- A *parallel* program potentially runs more quickly than a sequential program by executing different parts of the computation **simultaneously** (in parallel). It may or may not have more than one logical thread of control.
- **Concurrency is a property of the code; parallelism is a property of the running program.**

Concurrent or Parallel? (cont.)

- Alternative way of thinking about it:
  - Concurrency is part of the problem domain.
    - Multiple events can **happen** at the same time.
  - Parallelism is an aspect of the solution domain.
    - Design a program such that computations occur **simultaneously**.

Concurrency: Beyond Multiple Cores



Why Worry about Concurrency?

- *“Concurrency is hard and I’ve only ever needed single-threaded programs. Why should I care about it?”*

- The days of waiting for faster hardware is (long) gone.
- To make software systems that perform efficiently, you need to incorporate concurrency into your system designs.

Why Worry about Concurrency? (cont.)

- *"Concurrency is hard and I've only ever needed single-threaded programs. Why should I care about it?"*

- Changing environment
  - multicore computers (including handheld devices)
  - use of computing **clusters** to solve problems on the rise
- Performance
  - Growth rates for chip speed are flat; You can't wait 18 months for a 2x speed-up anymore.
  - Instead, chips are becoming "wider": more cores, wider bus (more data at a time), more memory.

Why Worry about Concurrency? (cont.)

- Chips are not getting faster (in the same way they used to).
  - A single-threaded, single-process application is not going to see any significant performance gains from new hardware.
- Instead, software will only see performance gains from new hardware.
  - If it is designed to do more work in parallel.
  - As the number of processors available to it increases.
- The application's computations must be amenable to **parallelization**.
  - It must be possible to break its work into tasks that can run at the same time with no need to coordinate with each other.
- Concurrent programming is becoming hard to ignore.
  - Lots of application domains in which concurrency is the norm.

But …

- While concurrency is widespread, it is error prone.
- Programmers trained for single-threaded programming face unfamiliar problems.
  - Synchronization
  - Race conditions
  - Deadlocks
  - “Memory barriers”

Parallel Architecture

Bit-Level

- The history of computing moves from 8- to 16-, 32-, and now 64-bit architectures.

Instruction-Level

- Modern CPUs are highly parallel, using techniques like *pipelining*, *out-of-order execution*, and *speculative execution*.

Data-Level

- *Data-parallel* (SIMD, for "single instruction, multiple data") architectures are capable of performing the same operations on a large quantity of data in parallel.

Task-Level

- *Distributed-memory system*: each processor has its own local memory and where interprocessor communication is primarily via the network.

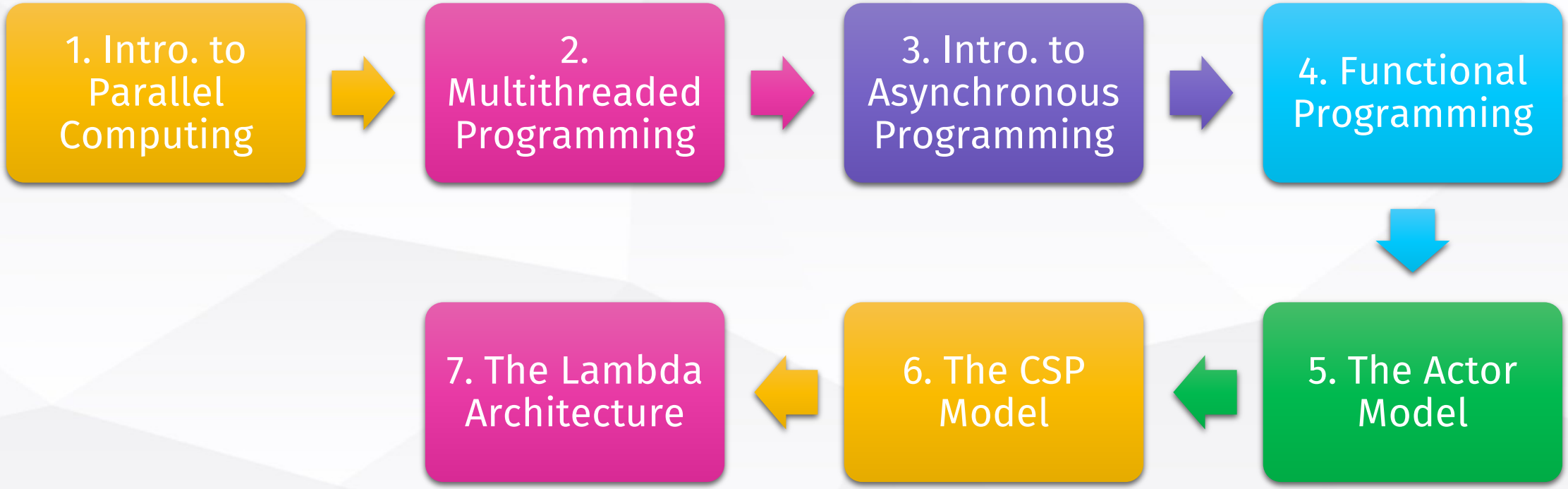
Software Architecture Design Choices

- When designing a modern application, we now have to ask
  - How many machines are involved?
  - What software components will be deployed on each machine?
  - For each component
    - Does it need concurrency?
    - If so, how will we achieve that concurrency?
      - multiple threads?
      - multiple processes?
      - both?

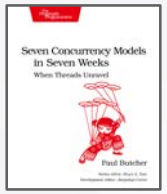
Case Study: Chrome

- Google made a splash in 2008 by announcing the creation of a new web browser that was
  - multi-process (one process per tab) and
  - multi-threaded (multiple threads handle loading of content within each tab)
- They documented their engineering choices via a comic book.
- Advantages: stability, speed and security.

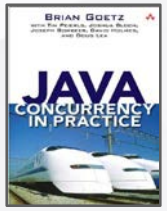
Course Topics



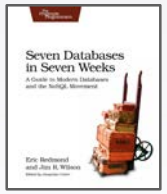
Bibliographies



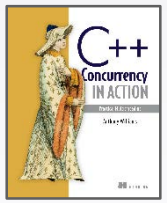
Seven Concurrency Models in Seven Weeks



Java Concurrency in Practice

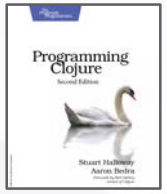


Seven Databases in Seven Weeks

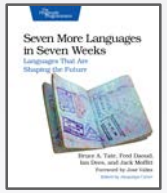


C++ Concurrency In Action

Seven Languages in Seven Weeks



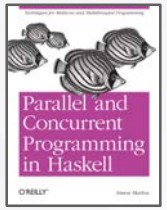
Programming Clojure (2nd Edition)



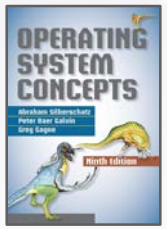
Seven More Languages in Seven Weeks

Programming Elixir 1.3

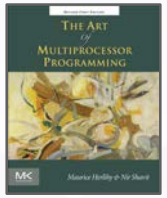
Bibliographies (cont.)



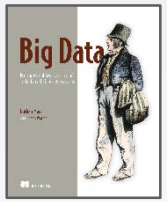
Parallel and Concurrent Programming in Haskell



Operating System Concepts (9th Edition)

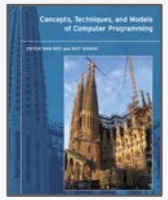


The Art of Multiprocessor Programming (Revised 1st Edition)

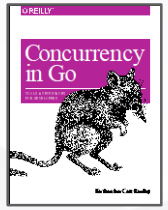


Big Data: Principles and Best Practices of Scalable Real-time Data Systems

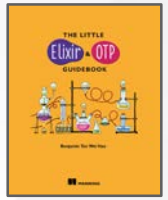
Big Data SMACK



Concepts, Techniques, and Models of Computer Programming



Concurrency in Go



The Little Elixir & OTP Guidebook

谢谢

Thank You