COMP 322: Parallel and Concurrent Programming

Lecture 37: Concurrent and Parallel Languages and Frameworks

Mack Joyner
mjoyner@rice.edu

http://comp322.rice.edu

COMP 322, Spring 2023 (M. Joyner)

What have we learned in this course?

- Functional programming for parallelism

- Lazy computation, streams

- Futures and promises

- Data-driven programming approach

- Computation graphs and their properties

- Map/Reduce programming model

- Data-parallel programming model

- Loop parallelism

- Locality control

- Handling concurrency while avoiding deadlock/livelock/starvation

- Barrier and point-to-point synchronization

- Actor programming model

Habanero

- Habanero-Java and Habanero-C
- Async/finish, futures/promises, loop parallelism, phasers, locality control, actors, isolation
- HJlib is a library implementation of these features
- Still developed and improved
- Python, Kotlin, Go, X10, MPI, Chapel, Java, C/C++
- There's also PCDP-Java
  - Coursera equivalent of COMP 322
- No streams

https://habanero.cc.gatech.edu/

X10

- Designed and developed at IBM

- One of the original "Next-generation" Asynchronous Partitioned Global Address Space projects

- Ancestor of Habanero Java

- Async, finish, loop parallelism, clocks (phasers), locality control

- No abstract metrics, data-driven execution, actors, streams

http://x10-lang.org/

Chapel

- Designed, implemented and maintained by Cray

- Partitioned Global Address Space

- Loop parallelism, task parallelism

- Locality control

- Distributed system execution

- Tasks, futures, promises

- No phasers, actors, abstract metrics, data-driven execution

https://chapel-lang.org/

Kotlin

- From the creators of IntelliJ

- Based on Java

- Multi-paradigm programming language

  - Functional, object-oriented

- Lots of support for functional programing

- More compact than Java

- Fully interoperable with Java

- Support for coroutines: very similar to asyncs and future tasks

- Low-level synchronization between tasks

- Channels

- No loop parallelism, phasers, abstract metrics, streams, locality control, actors

https://kotlinlang.org/

Go

- Multi-paradigm, object-oriented, concurrent language

- Goroutines (asyncs)

- Channels

- Concurrency control structures

  - Sending messages between coroutines

- No phasers, loop parallelism, futures/promises, abstract metrics, actors, locality control

https://go.dev/

Python/Ray

- Library based approach

- Aimed at data science, machine learning, data processing

- Futures and actors

- No task-level parallelism on shared memory

- No abstract metrics, phasers, loop parallelism
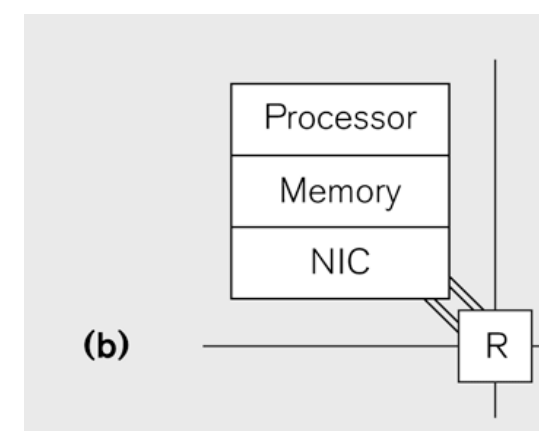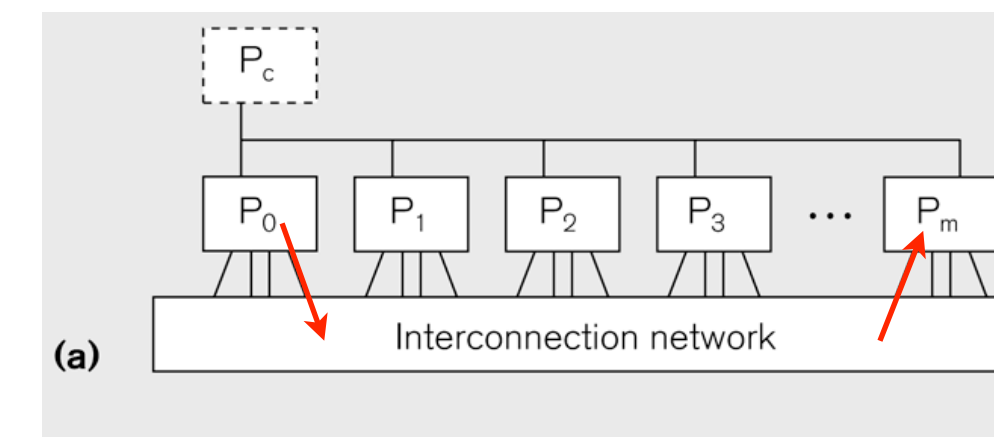
https://www.ray.io/

MPI

- Library framework

- Message-passing programming model

- Designed for distributed systems

- Implementations on top of several programming languages

  - C/C++

  - Java

  - Fortran

  - Julia, MATLAB, OCaml, Python, R

- Implementations for most modern supercomputers

- No tasking, futures/promises, abstract metrics, streams, phasers

Summary

- Concurrent and parallel programming is becoming pervasive

- Many languages and frameworks support some aspects

- Most of them do not support all aspects of concurrent and parallel programming

- It's possible to build additional features on top of a few basic ones

- You have learned most of the basic concepts in COMP 322