MPI and Ray
Lecture 21, cs262a

Ion Stoica & Ali Ghodsi
UC Berkeley

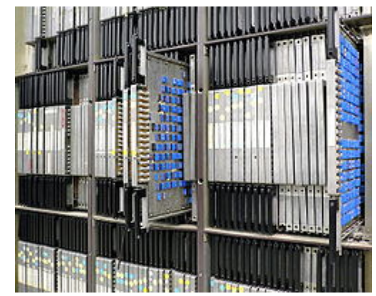
November 9, 2020

“*A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard*”, William Gropp Ewing Lusk and Nathan Doss and Anthony Skjellum (https://ucbrise.github.io/cs262a-fall2020/)

“*Ray: A Distributed Framework for Emerging AI Applications*”, Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan and Ion Stoica (https://ucbrise.github.io/cs262a-fall2020/)

High Performance Computing

1960: UNIVAC, first supercomputer



1975: ILLIAC IV

- First massively parallel computer
- SIMD (Single Instruction Multiple Data)
  - 1CPU and 64 processors
    (original design was for 4 CPUs and 256 FPUs)

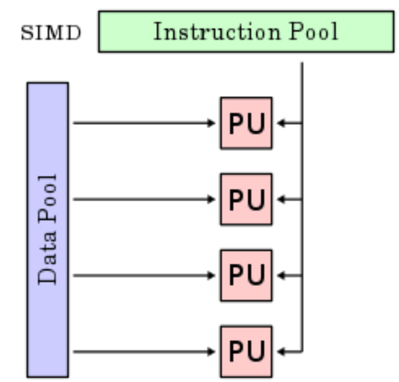
1976: Cray 1

- First supercomputer with a vector processing unit

SIMD

SIMD: Single Instruction Multiple Data

- Multiple Processor Units (PUs), each running the **same** instruction to process **different** data



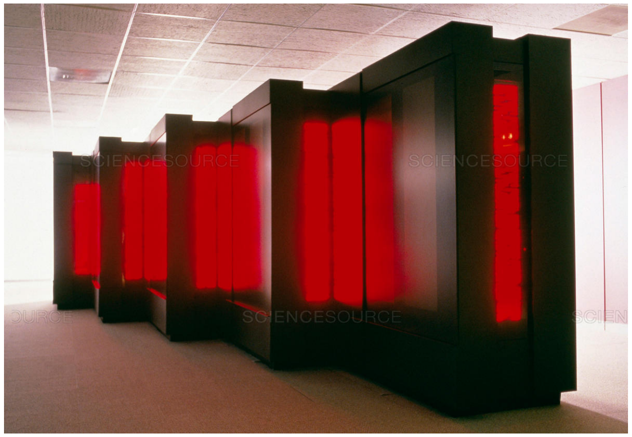
The rise of multi-processor supercomputers

Despite Cray's argument:

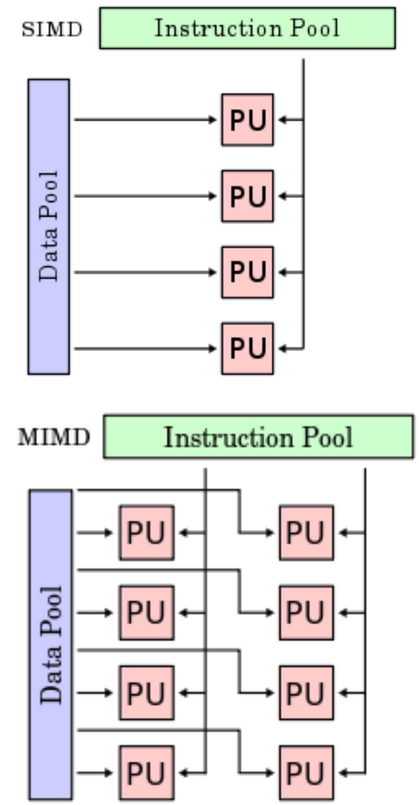
- "If you were plowing a field, which would you rather use? Two strong oxen or 1024 chickens?"

1995: CM-5

- MIMD (Multiple Instructions Multiple Data) architecture
- 32 SPARC processors and 128 vector processing units



SIMD vs MIMD



SIMD: Single Instruction Multiple Data

- Multiple Processor Units (PUs), each running the **same** instruction to process **different** data

MIMD: Multiple Instructions Multiple Data

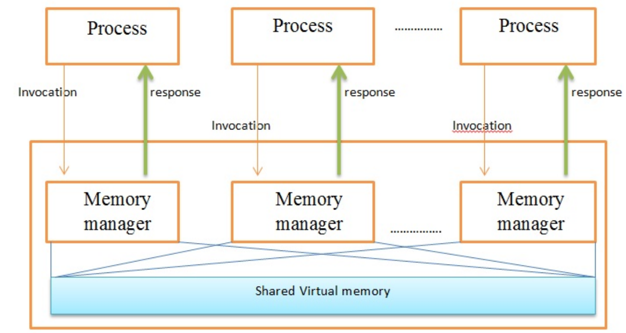
- Multiple Processor Units (PUs), each running **different** instruction to process **different** data

All supercomputers today are MIMD

How do you program MIMD computers?

Distributed Shared Memory

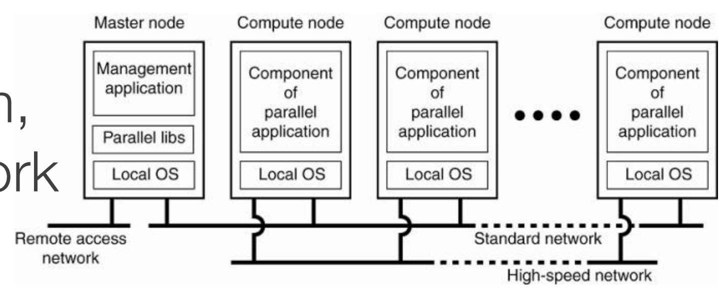
- Provide the abstraction of a single machine with multiple processors



Distributed shared memory

Message passing

- Provide the abstraction of a distributed system, i.e., multiple computers connected by a network



How do you program these supercomputers?

Distributed Shared Memory:

- Great abstraction in theory
- But bad in practice; couldn’t implement the abstraction:
  - Cannot hide access performance: local memory vs remote memory
    - Very hard to reason about the application performance
  - Extremely hard to make failures transparent
  - Data consistency (memory coherence) very hard

Lots of research, but little practical success!

MPI – Message Programming Interface

Draft for MPI presented at Supercomputing '93 conference in November 1993

The standard for programming supercomputers today

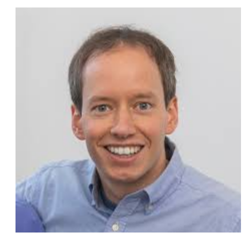
Ray: How everything started?

CS 294: Big Data Systems (Fall 2015)

One class project: data parallel training

- Asked two ML students (Robert and Philipp) to use Spark for this!

Robert Nishihara



Philipp Moritz

Published as a conference paper at ICLR 2016

SPARKNET: TRAINING DEEP NETWORKS IN SPARK

Philipp Moritz,* **Robert Nishihara**,* **Ion Stoica**, **Michael I. Jordan**
Electrical Engineering and Computer Science
University of California
Berkeley, CA 94720, USA
{pcmoritz,rkn,istoica,jordan}@eecs.berkeley.edu

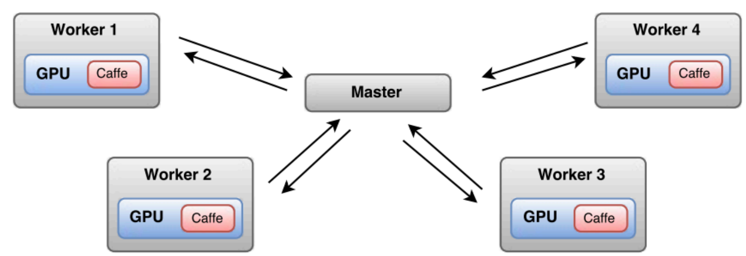


Figure 1: This figure depicts the SparkNet architecture.

A few challenges

Hard to use GPUs from Spark (because JVMs)

Soon started to focus on more complex workloads, e.g., reinforcement learning (RL)

- Spark's BSP model too constraining

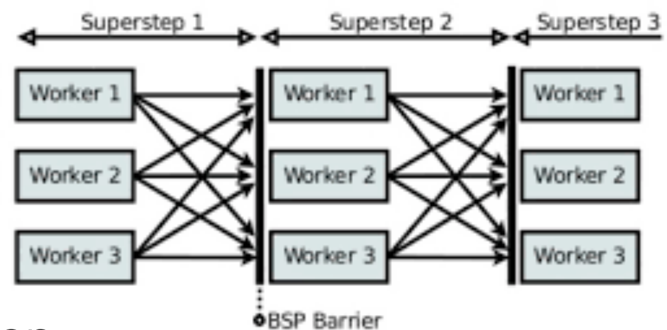
Python emerging as Lingua Franca for ML

Ray development started in 2016 (open sourced in 2017)

More context

SIMD → MIMD

Spark → Ray (other actor frameworks)



- Spark: Bulk Synchronous Processing (BSP)
  - Single stream of instructions
  - Each instruction operates on a different partition
- Ray: Explicit parallelism (more flexible but harder to program)
  - Each remote function/actor can execute different code on different data

But keep in mind that all are Turing complete so you can always emulate one with another ;-)