Parallel Programming Systems and Models

(Part 2)

Ioan Raicu
Center for Ultra-scale Computing and Information Security
Department of Electrical Engineering & Computer Science
Northwestern University

EECS 395 / EECS 495
Hot Topics in Distributed Systems: Data-Intensive Computing
February 11th, 2010

Parallel Algorithms

- Models
  - A way to structure a parallel algorithm by selecting decomposition and mapping techniques in a manner to minimize interactions.

Parallel Algorithms

- Models
  - Data-parallel
  - Task graph
  - Work pool
  - Master-slave
  - Pipeline
  - Hybrid

Parallel Algorithms

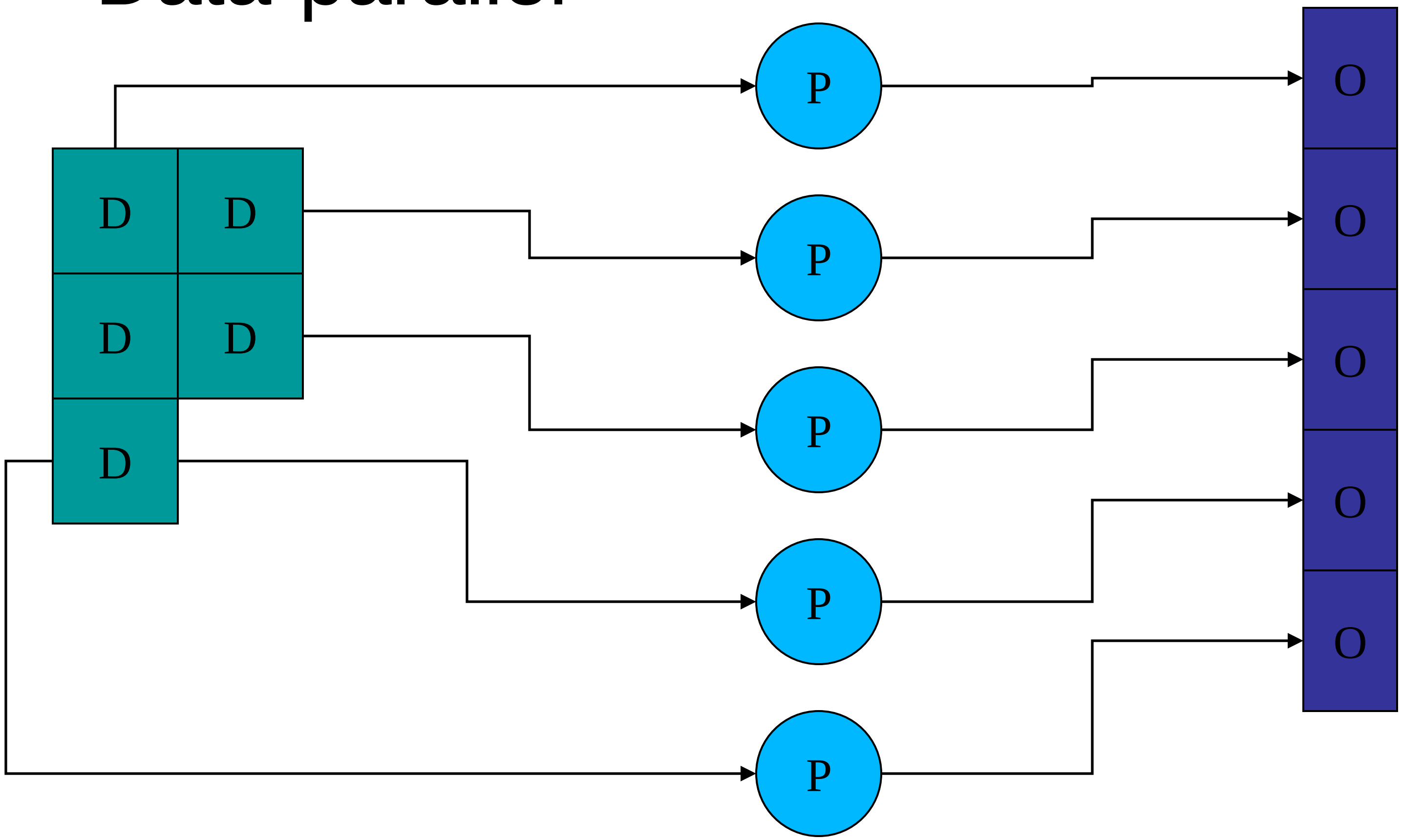
- Data-parallel
  - Mapping of Work
    - Static
    - Tasks -> Processes
  - Mapping of Data
    - Independent data items assigned to processes (Data Parallelism)

Parallel Algorithms

- Data-parallel
  - Computation
    - Tasks process data, synchronize to get new data or exchange results, continue until all data processed
  - Load Balancing
    - Uniform partitioning of data
  - Synchronization
    - Minimal or barrier needed at end of a phase
  - Examples
    - Ray Tracing

Parallel Algorithms

- Data-parallel



Parallel Algorithms

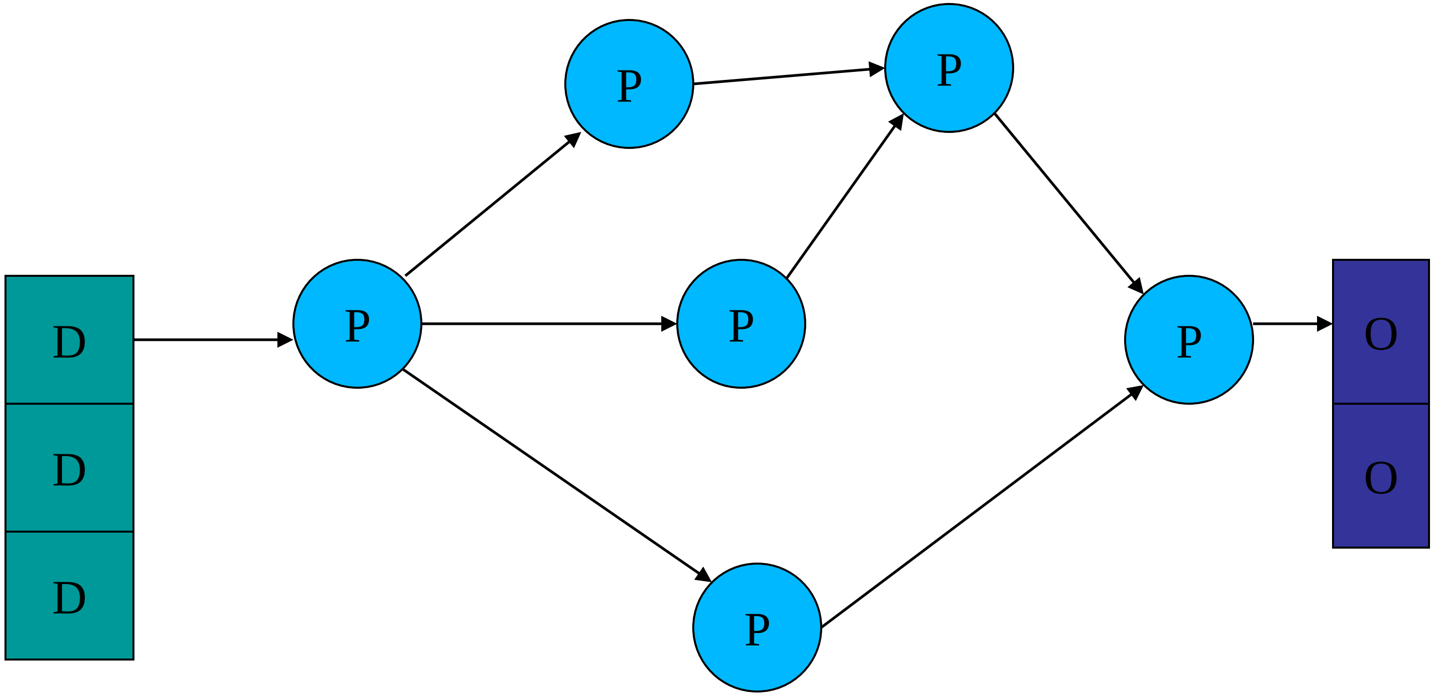
- Task graph
  - Mapping of Work
    - Static
    - Tasks are mapped to nodes in a data dependency task dependency graph (Task parallelism)
  - Mapping of Data
    - Data moves through graph (Source to Sink)

Parallel Algorithms

- Task graph
  - Computation
    - Each node processes input from previous node(s) and send output to next node(s) in the graph
  - Load Balancing
    - Assign more processes to a given task
    - Eliminate graph bottlenecks
  - Synchronization
    - Node data exchange
  - Examples
    - Parallel Quicksort, Divide and Conquer approaches
    - Scientific Applications that can be expressed in workflows (e.g. DAGs)

Parallel Algorithms

- Task graph



Parallel Algorithms

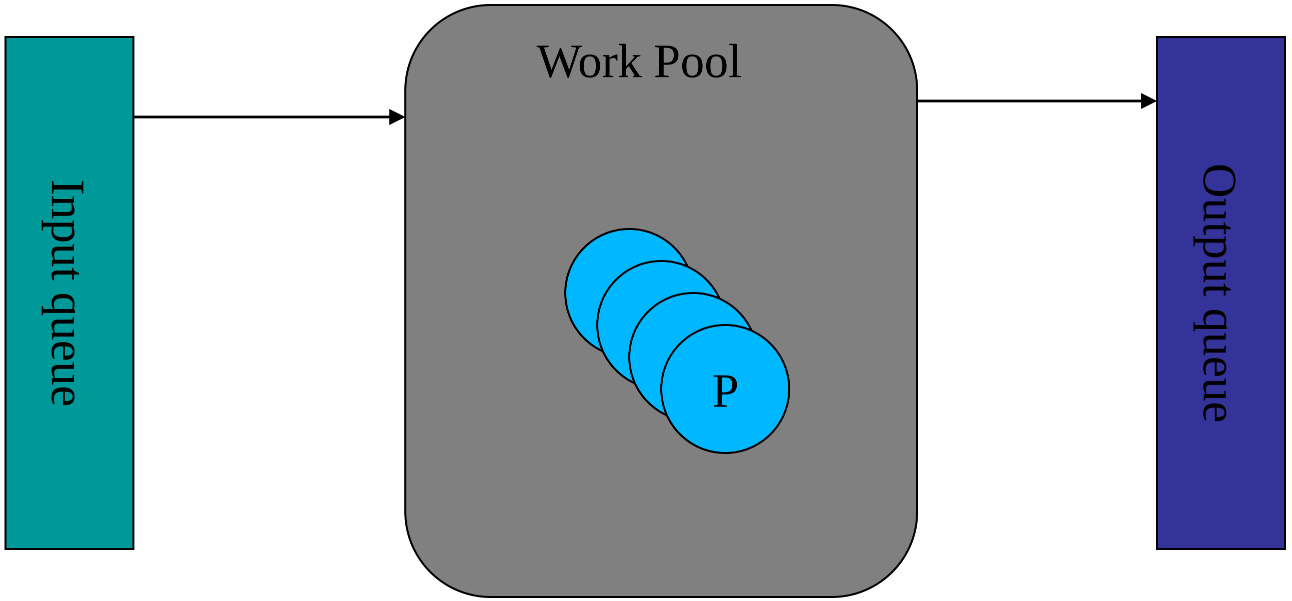
- Work pool
  - Mapping of Work/Data
    - No desired pre-mapping
    - Any task performed by any process
    - Pull-model oriented
  - Computation
    - Processes work as data becomes available (or requests arrive)

Parallel Algorithms

- Work pool
  - Load Balancing
    - Dynamic mapping of tasks to processes
  - Synchronization
    - Adding/removing work from input queue
  - Examples
    - Web Server
    - Bag-of-tasks

Parallel Algorithms

- Work pool



Parallel Algorithms

- Master-slave
  - Modification to Worker Pool Model
    - One or more Master processes generate and assign work to worker processes\
    - Push-model oriented
  - Load Balancing
    - A Master process can better distribute load to worker processes

Parallel Algorithms

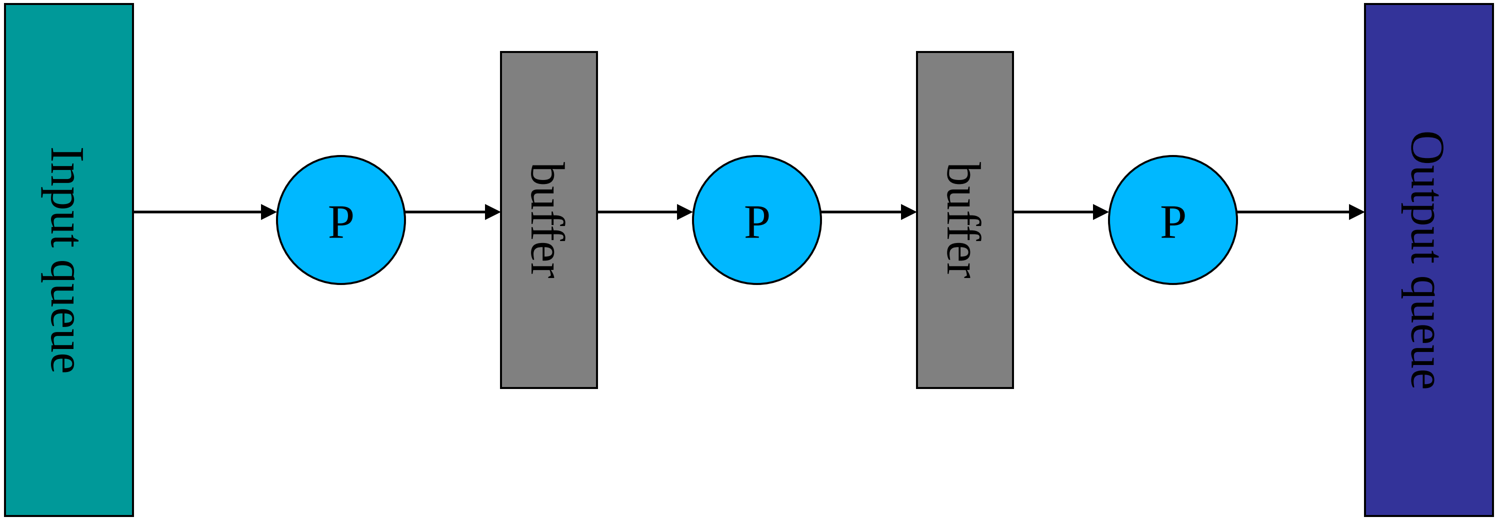
- Pipeline
  - Mapping of work
    - Processes are assigned tasks that correspond to stages in the pipeline
    - Static
  - Mapping of Data
    - Data processed in FIFO order
      - Stream parallelism

Parallel Algorithms

- Pipeline
  - Computation
    - Data is passed through a succession of processes, each of which will perform some task on it
  - Load Balancing
    - Insure all stages of the pipeline are balanced (contain the same amount of work)
  - Synchronization
    - Producer/Consumer buffers between stages
  - Ex: Processor pipeline, graphics pipeline

Parallel Algorithms

- Pipeline



Common Parallel Programming Models

- Message-Passing
- Shared Address Space

Common Parallel Programming Models

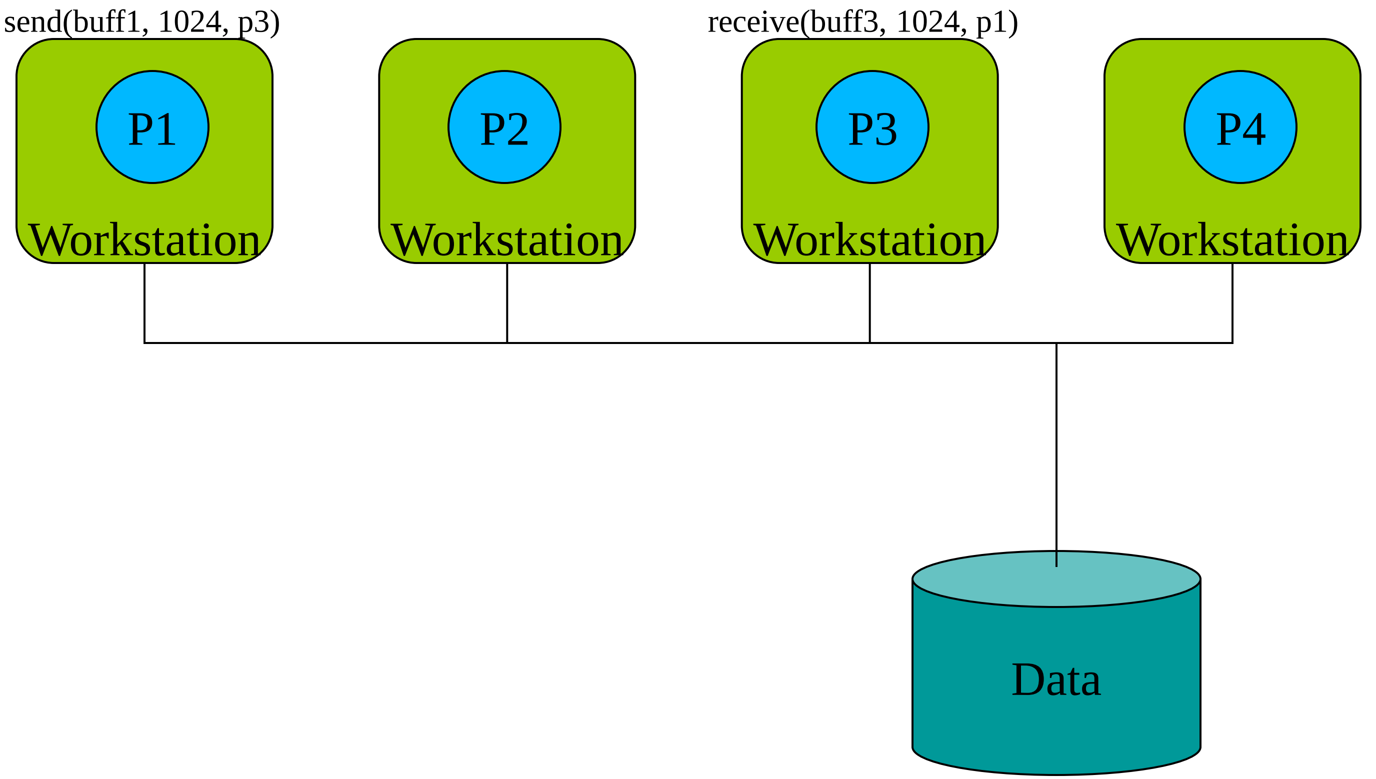
- Message-Passing
  - Most widely used for programming parallel computers (clusters of workstations)
  - Key attributes:
    - Partitioned address space
    - Explicit parallelization
  - Process interactions
    - Send and receive data

Common Parallel Programming Models

- Message-Passing
  - Communications
    - Sending and receiving messages
    - Primitives
      - send(buff, size, destination)
      - receive(buff, size, source)
      - Blocking vs non-blocking
      - Buffered vs non-buffered
    - Message Passing Interface (MPI)
      - Popular message passing library
      - ~125 functions

Common Parallel Programming Models

- Message-Passing



Common Parallel Programming Models

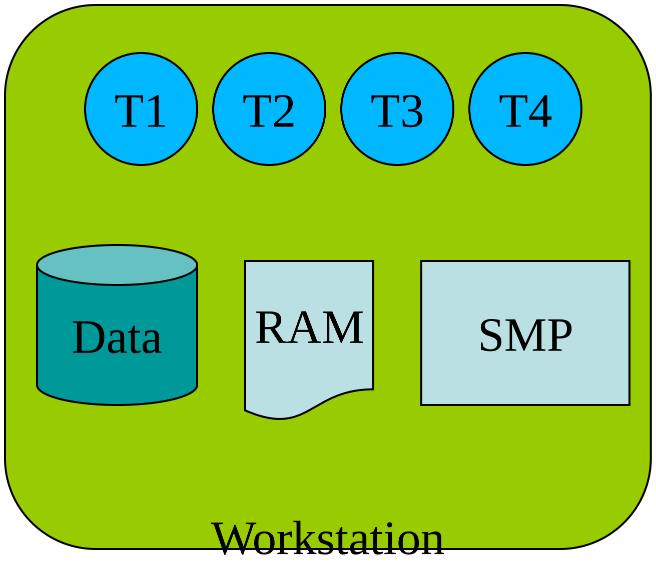
- Shared Address Space
  - Mostly used for programming SMP machines (multicore chips)
  - Key attributes
    - Shared address space
      - Threads
      - Shmget/shmat UNIX operations
    - Implicit parallelization
  - Process/Thread communication
    - Memory reads/stores

Common Parallel Programming Models

- Shared Address Space
  - Communication
    - Read/write memory
      - EX: x++;
  - Posix Thread API
    - Popular thread API
    - Operations
      - Creation/deletion of threads
      - Synchronization (mutexes, semaphores)
      - Thread management

Common Parallel Programming Models

- Shared Address Space



Parallel Programming Pitfalls

- Synchronization
  - Deadlock
  - Livelock
  - Fairness
- Efficiency
  - Maximize parallelism
- Reliability
  - Correctness
  - Debugging

Questions

?