Parallel and Scalable Architectures -1
Carlos Jaime Barrios Hernandez
@carlosjaimebh
Parallel Computing Architectures

Shared Memory

Distributed Memory

Homogeneous Architecture

Heterogeneous Architecture
Machine Models

• What is a machine model?
  – A abstraction describes the operation of a machine.
  – Allowing to associate a value (cost) to each machine operation.

• Why do we need models?
  – Make it easy to reason algorithms
  – Hide the machine implementation details so that general results that apply to a broad class of machines to be obtained.
  – Analyze the achievable complexity (time, space, etc) bounds
  – Analyze maximum parallelism
  – Models are directly related to algorithms.
Parallel Machine Model

The von Neumann computer

The multicomputer, an idealized parallel computer model. Each node consists of a von Neumann machine
RAM and PRAM Models

**RAM.** Random Access Machine

**PRAM.** Parallel Random Access Machine
RAM (Random Access Machine) model

- Memory consists of infinite array (memory cells).
- Each memory cell holds an infinitely large number.
- Instructions execute sequentially one at a time.
- All instructions take unit time
  - Load/store
  - Arithmetic
  - Logic
- Running time of an algorithm is the number of instructions executed.
- Memory requirement is the number of memory cells used in the algorithm.
RAM (random access machine) model

- The RAM model is the base of algorithm analysis for sequential algorithms although it is not perfect.
  - Memory not infinite
  - Not all memory access take the same time
  - Not all arithmetic operations take the same time
  - Instruction pipelining is not taken into consideration

- The RAM model (with asymptotic analysis) often gives relatively realistic results.
PRAM (Parallel RAM)

- A model developed for parallel machines
  - An unbounded collection of processors
  - Each processor has an infinite number of registers
  - An unbounded collection of shared memory cells.
  - All processors can access all memory cells in unit time (when there is no memory conflict).
  - All processors execute PRAM instructions synchronously
    - Somewhat like SIMD, except that different processors can run different instructions in the lock step.
    - Some processors may idle.
A model developed for parallel machines
- Each PRAM instruction executes in 3-phase cycles
  - Read from a share memory cell (if needed)
  - Computation
  - Write to a share memory cell (if needed)
  - Read $A[i-1]$, compute add 1, write $A[i]$
- The only way processors exchange data is through the shared memory.
PRAM (Parallel RAM)

- PRAM Provides an ideal model of a parallel machine (computer) for analyzing the efficiency of parallel algorithms.

- PRAM composed of
  - $P$ Unmodifiable programs, each composed of optionally labeled instructions
  - a single shared memory composed of a sequence of words, each capable of containing an arbitrary integer
  - $P$ accumulators, one associated with each program
    - A read-only input tape
    - A write-only output tape

Parallel time complexity: the number of synchronous steps in the algorithm

Space complexity: the number of share memory

Parallelism: the number of processors used
PRAM (A simple vision)

All processors can do things in a synchronous manner (with infinite shared Memory and infinite local memory), how many steps do it take to complete the task?
PRAM – further refinement

- PRAMs are further classified based on how the memory conflicts are resolved.
  - Read
    - Exclusive Read (ER) – all processors can only simultaneously read from distinct memory location (but not the same location).
    - What if two processors want to read from the same location?
  - Concurrent Read (CR) – all processors can simultaneously read from all memory locations.
PRAM – further refinement

- **Write**
  - Exclusive Write (EW) – all processors can only simultaneously write to distinct memory location (but not the same location).
  - Concurrent Write (CW) – all processors can simultaneously write to all memory locations.
    - Common CW: only allow same value to be written to the same location simultaneously.
    - Random CW: randomly pick a value
    - Priority CW: processors have priority, the value in the highest priority processor wins.
PRAM model variations

- EREW, CREW, CRCW (common), CRCW (random), CRCW (Priority)
  - Which model is closer to the practical SMP or multicore machines?

- Model A is computationally stronger than model B if and only if any algorithm written in B will run unchange in A.
  - EREW <= CREW <= CRCW (common) <= CRCW (random)

- And there are other models as BSP, LogP… (To see later)
PRAM algorithm example

- **SUM**: Add N numbers in memory $M[0, 1, ..., N-1]$

- Sequential SUM algorithm ($O(N)$ complexity)
  
  ```
  for (i=0; i<N; i++) sum = sum + M[i];
  ```

- PRAM SUM algorithm?
Which PRAM model?
Time complexity?
Space complexity?
Parallelism?
Speedup (.vs. sequential code)?
Parallel Addition

- Time complexity: $\log(n)$ steps
- Parallelism: $n/2$ processors
- Speed-up (vs sequential algorithm): $n/\log(n)$
Broadcast in PRAM

- EREW
  - double the number of processors that have the value in each steps
  - Log(P) steps

- CREW
  - Broadcaster sends value to shared memory
  - All processors read from shared memory
  - O(1) steps
Parallel search algorithm

- P processors PRAM with unsorted N numbers (P<=N)
- Does x exist in the N numbers?
- p_0 has x initially, p_0 must know the answer at the end.

PRAM Algorithm:
- Step 1: Inform everyone what x is
- Step 2: every processor checks N/P numbers and sets a flag
- Step 3: Check if any flag is set to 1.
Parallel search algorithm

- **PRAM Algorithm:**
  - Step 1: Inform everyone what x is
  - Step 2: Every processor checks N/P numbers and sets a flag
  - Step 3: Check if any flag is set to 1.

- **EREW:** $O(\log(p))$ step 1, $O(N/P)$ step 2, and $O(\log(p))$ step 3.

- **CREW:** $O(1)$ step 1, $O(N/P)$ step 2, and $O(\log(p))$ step 3.

- **CRCW (common):** $O(1)$ step 1, $O(N/P)$ step 2, and $O(1)$ step 3.
Find Max of N items

- CRCW algorithm with $O(1)$ time using $N^2$ processors
  - Processor (r, 1) do $A[s] = 1$
  - Process (r,s) do if $(X[r] < X[s])$ $A[r] = 0$;
  - Process (r, 1) do: If $A[r] = 1$, $max = X[r]$;
PRAM matrix-vector product

- Given an $n \times n$ matrix $A$ and a column vector $X = (x[0], x[1], \ldots, x[n-1])$, $B = A \cdot X$

- Sequential code:
  
  ```
  For(i=0; i<n; i++)
  for(j=0; j<n; j++)
  B[i] += A[i][j] * X[j];
  ```

- CREW PRAM algorithm
  
  - Time to compute the product?
  - Time to compute the sum?
  - Number of processors needed?
  - Why CREW instead of EREW?
PRAM matrix multiplication

- CREW PRAM algorithm?
  - Time to compute the product?
  - Time to compute the sum?
  - Number of processors needed?
PRAM strengths

• Natural extension of RAM

• It is simple and easy to understand
  – Communication and synchronization issues are hided.

• Can be used as a benchmark
  – If an algorithm performs badly in the PRAM model, it will perform badly in reality.
  – A good PRAM program may not be practical though.

• It is useful to reason threaded algorithms for SMP/multicore machines.
PRAM weaknesses

• Model inaccuracies
  – Unbounded local memory (register)
  – All operations take unit time
  – Processors run in lock steps

• Unaccounted costs
  – Non-local memory access
  – Latency
  – Bandwidth
  – Memory access contention
PRAM variations

- **Bounded memory PRAM, PRAM(m)**
  - In a given step, only m memory accesses can be serviced.
  - Lemma: Assume \( m' < m \). Any problem that can be solved on a \( p \)-processor and \( m \)-cell PRAM in \( t \) steps can be solved on a \( \max(p,m') \)-processor \( m' \)-cell PRAM in \( O(tm/m') \) steps.

- **Bounded number of processors PRAM**
  - Lemma: Any problem that can be solved by a \( p \) processor PRAM in \( t \) steps can be solved by a \( p' \) processor PRAM in \( t = O(tp/p') \) steps.
    - E.g. Matrix multiplication PRAM algorithm with time complexity \( O(\log(N)) \) on \( N^3 \) processors → on \( P \) processors, the problem can be solved in \( O(\log(N)N^3/P) \).

- **LPRAM**
  - L units to access global memory
  - Lemma: Any algorithm that runs in a \( p \) processor PRAM can run in LPRAM with a loss of a factor of \( L \).
PRAM summary

• The RAM model is widely used.

• PRAM is simple and easy to understand
  – This model never reaches beyond the algorithm community.
  – It is getting more important as threaded programming becomes more popular.

• The BSP (bulk synchronous parallel) model is another try after PRAM.
  – Asynchronously progress
  – Model latency and limited bandwidth
Remember...Computing Elements

Microkernel

Multi-Processor Computing System

Threads Interface

Hardware

Operating System

Programming paradigms

Applications

Processor

Thread

Process
Multi Processor Computing Framework

Programming Models

Runtime Optimization

Tools Debug/Profiling
MPC Execution Model
(An Example)
MPC Execution Model

(An example of 4 Tasks and 4 threads)

2 MPI tasks + OpenMP parallel region w/ 4 threads (on 2 cores)

MPC Execution Model
(An Example with GPUs)
Parallel Communication

Serial Communication

Parallel Communication
A General Communication - processing Model
In Parallel...
Interconnection Networks Topologies

(a) Static

(b) Dynamic
Flynn’s Taxonomy

*Proposed by M. Flynn in 1966*
Some Glosary

- Any scalable system is a distributed system.

- **Parallel computing** uses **Scalable Systems**
  - Many instructions are carried out simultaneously-concurrently.
  - High Performance Computing (HPC) implies **Parallel Computing**

- **Scalable Systems** may be describe in terms of **Scalable Architectures**.

- **Scalable Architectures** (hardware point of view) have the characteristics of **Scalable Systems**.
  - Concurrency, Distribution

- **Scalable Architectures** support **Parallel Computing**.

- Of course, **Parallel Computing** implies parallelism.

- Obviously, **Parallel Computing** demands **Parallel Machines**.
Conclusions

- Abstractions of Parallel Architectures are addressed to understand execution models (to see in detail after parallel programming execution models).
  - Related with Algorithms
  - Related with Implementation Mechanisms

- Scalability is a condition to understand in some aspects
  - Hardware
  - Data
  - Processing
Thank you! @SC3UIS