4005-Final-Review

Lecture 1 Introduction

Shared Memory Mutiprocessor System:

each process can access any memory of module. Two types

- Parallel Programming language. With special progarmming constructs and statements that allow shared variables to be declared. Like fortan D, C*
- Threads.

Each processor has the access to the whole memory using single memory address space.

MIMD and SIMD:

In a single processor, the instruction operate upon a single stream of data items, and call it a single instruction stream-single data stream. (SISD)

- MIMD: each processor has a separate program and one instruction stream is generated from each program for each processor. each instruction operates upon different data. **both shared memory and message-passing mutiprocessor so far descriped are in the MIMD classification.**
- SIMD: single data stream but there are mutiple data stream exist. The instruciton form the program are broadcast to more than one processor. each one execute the same instruction in synchronism but using different data. Developed because there are a number of application upon **arrays of data.** like sorting.
- MPMD: Within the MIMD classification, each process have its onw program to execute.
- SPMD: single program mutiple data. single source program is written and each processor will execute its personal copy of this program. although independency and not in synchronism.

Message Passing Muti-Processor:

There are Three ways to pass the message.

- Static Network message Passing.
- Node with a switch for internode message transfer
- wires

NetWork Criteria:

Cost: indicate the number of links in network

BandWidth: number of bits that can be transmitted in unit time.

Network Latency: time to make the message transfer through network

Communication Latency: total time to send message, including overhead and interface delay

Message Latency or Startup time: time to send a zero-length message, Essentially the software and hardware overhead in sending message and actual transmission time.

Diameter: minimum number of links between two farest nodes in the network, only shortest routes used, used to **determine the worst case delay.**

Interconnection Networks

Embedding: Embed a new network communication system into an original one.

Dilatio: used to indicate the quality of the embedding, which is the maximum number of links in the embedding network corresponding to the one link in the embedding network

Communication Methods

- Circuit Switch: Involves the establishing path and maintain all links in the path for message to pass.
- Packing Switching: message divide into packets and send the packet one node by one node.
- Virtual Cut Through: It can elimianted the storage latency compared with the packet one. it will immediately pass through without being stored in the nodal buffer.
- Wormhole Routing: Message dividied into smaller unit called flits (flow control digits) , only send the head to next node when links are available.

Point to Point Communication: provide the interconnection bandwidth.

Speed Up

Speed up factor: $\frac{t_s}{t_p}$, which is the execution time using one processor divide execution time using a mutiprocessor with n processors

The maximum Speed up is linear.

There is still some occasion due to suboptimial sequential algorithm.

Amdalh's Law: divide into sequential part and parallel part, and use the formula to calculate the time effiency.

Efficiency: time using one processor / time using mutiprocessor

Cost: execution time times the toatal number of processors used, hence we have the cost-optimal parallel algorithm.

Scalability: used to indicate with the increased data items with a low and bounded increase in computation step.

Lecture 2 Message Passing Computing and Programming

There are three ways to achieve the programming message-passing parallel

- Design a special parallel programming language
- extending the syntax words of existing sequential high-level language to handel it

- Using library of external procedures for message passing. necessary to say it is **explicit** what are process are to be executed.

Goals: A method of creating separate process for execution on different computers; sending and receiving message.

mutilple program and mutiple data stream: similar as `fork`

PVM

The programmer decompose the problem into separate part progam. each program is written in C or fortan, and complied to run on specific types of computers in the network

All PVM send routine is non-blocking while PVM recieve routines can be either blocking and non-blocking.

Communication

Send Communication Mode: Standard Mode Send, Buffer mode, synchronouns mode, ready mode.

Collective Commnucation Mode: broadcast

Lecture 8 Mutithreaded Programming

Heavy weight and light weight: `fork` is like a heavy weight and thread is the light weight.

Thread-safe Routine: simutaneously and always produce correct result.

Critical Section: critiacal sections is also known mutual exclusion

semaphore: is the mecanism simiar like lock. P operation and V operation.

Monitor: can monitor the data to avoid the data race.

conditional variable: like `wait`

Several Alternatives for programming:

- Using a new programming language
- modify an existing sequential code
- using library routines with sequential language
- Using a sequential programming language and ask complier to convert it into parallel code
- UNIX process

Create concurrent process: Using Fork-Join method.

Unix Heavy Weight Process: Unix system call fork to create a new process, and the child is the exact copy of that. which we called it is a heavy weight.

Compared with thread, It share the memory space and global variables between routines.

Spinlock 是内核中提供的一种比较常见的锁机制，*自旋锁是“原地等待”的方式解决资源冲突的，即，一个线程获取了一个自旋锁后，另外一个线程期望获取该自旋锁，获取不到，只能够原地“打转”（忙等待）。由于自旋锁的这个忙等待的特性，注定了它使用场景上的限制 —— *自旋锁不应该被长时间的持有（消耗 CPU 资源）。

Dependency Analysis

Use Berstein's Conditions to check there is the data race or the data dependent.

Ij i sthe set of memory locations read by the process Pi
Oj is teh set of memory locarions altered by process Pj

then I1 && O2 is empty and I2 && O1 is empty also O1 && O2 is empty represent the two prcocess can concurrently execute it.

Lecture 3 Embarrassingly Parallel Computations

deal with the computation that can be divided into a number of completely independent parts, each parts can be executed by a separate processor. Such as Mandbrot Sort.

There are two examples for this chapter:

- The first one is Monte Carlo Methods, which is used to calculate the size of the circle. In this case, the data is independent so we can apply the parallel computation just in a naive way.
- The second one is Parallel Random Number Generation, whichi is used to generate the random number.

Lecture 4 Partitioning and Divide-Conquer

This Lecture is the strategies of paralle computatation that it can deal with the problem that need to divide into several parts and conquer.

Bucket Sort:

all to all routine: is a all reduce routine.

Lecture 5 Pipline

In the pipline technique, the problem will divide into a series of tasks that we have to be completed one after the other.

Three Types of pipeline computation;

1. If more than one intance of the complete problem is to be executed
2. If a series of data items must be processed, each requiring mutltiple data operations
3. If information to start the next process can be passed forward before the process has completed all its internal operations

Leture 6 Sychronous Computation

counter-based has two phases:

- Arrival phase
- Departure phase

Lecture 7 Loading Balancing

Static load balancing:

- Round robin algorithm: passes out the tasks in sequential order of processes coming bakc to the first when all process have been gievn a task.
- Randomized algorithm: select processes at random to take tasks
- Recursive bisection: recursively divide problem into subproblem of equal computational effort while minimizing message passing
- simulated annneling -- an optimization technique
- Genetic algorithm: another optimization technique

Static Load Balancing evaluation:

- very difficult to estimate accurately the execution time of various parts of programm without actually executing the parts,
- communication delays that vary under different circumastances.
- some problems have an indeterminate number of steps to reach the solution.

Dynamic Load Balancing:

- dependent on the execution part they are being execute.
- Does incur an additional communication overhead but still effective.

Two types of dynamic load balacing: centralized and decentralized

Task transfer mechanism:

- request task when it has few or no tasks to perfom, this is called recier-initiated method, however,
- the send-initialed method is about the process with heavy load passes out of some of its tasks to others that are willing to accept them.
- Round robin algorithm and random polling algorithm

Line structure: The master process feeds the queue with tasks at one end, and the tasks are shifted down the queue. High-priority or alrger tasks could be placed in the queue first.

Tree structure: tasks passed form node into one of the two nodes below it when node buffer empty

Termination

Distributed Termination conditions:

- Application-specific local termination conditions exist throughout the collection of processes at time t.
- there are no messages in transit between prcocess at time t.

Ring Termination condition: if terminated send it to next until all the tasks are terminated.