Brock University
Faculty of Mathematics & Science
Department of Computer Science

COSC 3P93: Parallel Computing

4P78 Assignment 1

Prepared By:

Parker TenBroeck 7376726
pt21zs@brocku.ca

Instructor:
Robson De Grande

October 13, 2025

1 Introduction

1.1

1. What is Shared-memory Architecture

   Each processor in the system shares the same memory space

2. What are its advantages and distavantages

   Is typically easier to program for

   Is typically more flexable

   Can be more costly for hardware design

   Caches and other optimizations need to be employed to reduce latency

3. Can we have multiple processes of the same program, sharing data in a Shared-memory System.

   Yes, each individual processor in the system can access all memory in the system. For example on linix you can use the mmap function.

1.2

1. What is a distributed-memory architecture

   Many individual computers/nodes each with their own private memory (programs and data) which communicate through message passing in a connected network with eachother

2. What are its advantages and disadvantages

   Harder to program for

   Easier to scale to large number of processors/nodes

   Generally less costly

3. Can we have a multithraded program sharing data among its threads in a Distributed-memory system?

   Yes, if we have two threads running on the same node/processor it is possible to share data between them since they share the same address space.

2 Parallel Hardware

2.1

1. What is NUMA memory?

   Non Uniform Memory Access is a system of memory where multiple processors can all access the same address space but for each processor some memory is local and has lower latancy and other regions are remote and can have higher latency.

2. Is NUMA Architecture better than UMA Architectures?

   In some situtations where you can control which memory sections each processor accesses it can be faster to access memory in most situtations.

2.2

List and explain the two techniques to maintain Cache Coherence

1. Snooping Cache Coherence:

   Any signal transmitted on the bus is seen by all cores connected to it

2. Directory-based Cache Coherence:

   Using a directory of addresses to cache lines to store the status of each cache line only updating when a dirty line is accessed.

2.3

1. What is a Hypercube Connection

   Each node is connected with d wires where d is the dismention of the cube

2. How does a Hypercupe topology improve when compared against 2D Grid Mesh and Fully Connected Networks?

   hyercube topology has improved efficiency in routing compared to 2D grid mesh due to lower average distance between node.

   for fully conected networks it significantly reduces the number of wires and complexity needed for fully connected networks.

2.4

1. Define and explain the Bisection and diameter of an N-node Hypercube

   the bisection width of a N-node hypercube is $\frac{N}{2}$

   the diameter of an N-node hypercube is $log_2(N)$

2. Why are tese two metrics important

   The bisection width indicates the minimum number of connections that need to be severed to split the network, affecting fault tolerance, while the diameter measures the maximum distance a message must travel, influencing communication delay.

2.5

1. Because when it is commented n1 and n2 are shared in the same cache line aka false sharing is happening

2. Instead each thread modifying consecutive 8 byte values(on 64 bit platforms) which would result in false sharing. each thread is modifying the first value of 64 bit blocks which prevents false sharing. resulting in faster times.

3 Parallel Software

3.1

1. What is data-parallel computation

   Performing the same operation on multiple "sets" of data at the same time. like SIMD instructions

2. What is task parallel computation

   doing multiple tasks at the same time

3. Can we have programs that are both Data and Task parallel?

   yes. if your computer (it probably does) have SIMD instructions and you do task parallelization then boom.

3.2

Is user process locking required to control the order of access to guarentee sequential consistency.

No there are synchronization primitives like atomics which can guatentee sequential consistency. as well as concurrent datastructures which are lockless which can also do the same.

Additionally barriers can be used to have similar effects.

3.3

Why are locks important in parallel algorithms? Define and explain the major problem that locks bring to parallel programs

Locks are a tool which prevent data races in our programs when multiple parts of our program are trying to access the same data at the same time.

Locking can create sequential data sections which can result in a parallel program operating if it was single threaded or even having worse performance than a single threaded program.

Locking can also cause deadlocks which will completely halt the program and prevent it from completing.

3.4

How can one ensure Mutual Exclusion without locks?

Using atomics to perform simple operations or using more elaborate lock free data structures like channels to message pass for communication. also by not sharing data between threads.

4 Threads

4.1

Assume a producer-condumer thread based program.

1. How producers and consumers can communicate with eachother?

   Producers "produce" messages (application specific) that is sent to consumers through a channel

   Consumers "consume" messages sent to them by producers and perform certain actions based on them

2. Why are queues FIFOs one of the best data structures for organizing their communication?

   Because it allows multiple messages to be produced/sent without blocking while also maintaining the order in which they are sent allowing consumers to process them in order.

In the contect of threads

1. What are condition variables

   Condidtion variables are used in conjunction with mutexes or other locks to allow for multiple threads to wait for some condition to be true about the variable to proceed.

2. What are read-write locks

   Read-Write locks are locks which allow locking for two different kind of operations

   Reading, where many threads which hold a read lock can read from the data it protects but none can mutate it.

   Writing, where only a single owner can hold the lock which permits both reading and writing to the variable.

   These locking kinds are mutually exclusive, as in if readers currently hold the lock no writer does and viseversa.

3. Why are they useful

   Condition variables are useful when we want to wait for some condition to be true but have nothing else to do in the meanwhile. aka want to sleep

   Read-Write locks are useful in situations where read operations happen much more frequently than write operations. Say for instance a config file cached in memory that is read from, but only updated if the file is written to.

4. Write two short scriplets, and explain/describe them, to exemplify the use of condition variables and read-write locks

5 OpenMP

5.1

1. What are OpenMP pragmas

   The way we tell the compiler what and how parts of our code should be parallelized

2. What are OpenMP directives

   They are for atomic operations and critical sections

3. What are OpenMP clauses

   They define how variables are shared, specific conditions and tell openMP how to manage task execution.

5.2

1. running the function calculate in parallel with the number of threads suggested being `thread_count`
2. Define critical sections `odd` and `even` so the sum operations on the two respective variables are counted properly but importantly the critical sections don't interfrere with eachother.
3. run the outer for loop in a number of threads suggested by `thread_count` and have the inner loop run in parallel in chunks where the loop pilicy is determined at runtime or compile time.

5.3

Explain the following loop schedule policies in detail

1. Default schedule

   The compiler or runtime decides which policy to use

2. Static schedule

   The iterations are assigned to the threads while the loop is executing

3. dynamic schedule

   The iterations are also broken up into chunks of chunksize consecutive iterations

4. Guided schedule

   Each thread executes a chunk, when if finishes it requests another chunk, each requested chunk is smaller until the minimum size.

5. Runtime schedule

   Uses a value specified at runtime through an enviornment variable for what loop schedule policy should be used