Homework 1 - Parallel Computing, Architectures, and

Performance Theory

Due Date: 4/21 @ 5pm

**Parallel Computing**

1. What is a parallel computer? Can a computer with only one processing core be considered a

parallel computer? Explain.

“A parallel computer is a computer system that uses multiple processing elements simultaneously in a cooperative manner to solve a computational problem.” Lecture 1 slide

A processing core can be considered as a parallel computer. Some of the processing core can have multiple pipeline stages to enable multiple instructions running concurrently.

2. What are the three components of parallelism?

Parallelism consists of concurrency, parallel hardware, and performance.

3. When executing the Linpack benchmark, is it possible for a parallel computer A to have a larger Rmax than a parallel computer B, but a smaller Nmax?

Yes, the Rmax is the maximal performance Linpack benchmark; and

the Nmax is the problem size needed to achieve Rmax.

Some computer is designed to run on a smaller data set so when the performance Rmax is obtained, the size Nmax is small; on the other hand, some computer is designed to run on larger data set, so when the Rmax is obtained, the size Nmax is larger.

4. What are some reasons why one would utilize parallel computing? Why is it important to

study parallel computing today more than any other time before?

There are 2 primary reasons: for a faster time to reach solution (response time) and to solve bigger computing problems (in same time)

Other factors might be to use machine resources effectively, and to overcome memory constraints.

It is important to study parallel computing today more than other time before because:

new computing architectures drive programming forward;

the technological convergence makes laptops and supercomputers more similar and cause diverse approaches to converge;

it is inevitable since most computing system is operating in parallel;

to understand the fundamental principles and design tradeoffs;

and it’s the future of computing.

Lecture 1 slide

**Parallel Architectures**

1. What is the difference between a shared memory parallel system and a distributed memory

parallel system? Give two advantages for each.

The different between a shared memory parallel system and a distributed memory parallel system is that:

A shared memory parallel system has multiple processors and a shared memory adderss space which **each processor can reference to any memory location**; and

A distributed memory parallel system has different nodes and each node has a local memory, **the processors must communicate to access non-local data**.

Advantages of a shared memory parallel system:

Widely compatibility with SMP hardware, the ease of programming when communication patterns are complex or vary dynamically during execution, lower communication overhead.

Advantages of a shared distributed parallel system:

The hardware can be simpler and more scalable; communication is explicit and simpler to understand.

Lecture 2 slide

2. What is a NUMA parallel architecture and why was it invented?

NUMA stands for non-uniform memory access, where memory is physically resident close to each processor to form a processing unit and has processing unit (node) connect by the network.

This architecture is more scalable since each processor would have its own memory, which reduces the communication (both hardware and software) required to maintain cache coherency; but cause performance dependent on data locality.

3. Why do we have multi-core processors? Are not single-core processors good enough? Why is

multi-core a disruptive technology from the point of view of parallel computing? (The high-

performance computing (HPC) community oftern refers to a new technology as "disruptive"

if it has the potential of causing a change in how HPC will be done going forward, di\_erent

from the status quo, often with the bene\_t of achieving much better performance.)

A multi-core processor has more than one processing units (cores) which allows them to run multiple instructions at the same time.

Manufacturers have difficulty increasing the single-core processor performance by increasing CPU clock frequency (thermal, power consumption problems, etc); in order to increase performance of a processor, they add multiple cores to increase performance.

The multi-core is a disruptive technology because it solves the inability to dissipate heat, high power consumption, and brings greater efficiencies; also multi-core processors allows multiple instructions to run at the same time.

4. For large-scale parallel systems, the interconnection network is key. Would you agree? Explain.

I do agree. Interconnection network is the core of the parallel architecture. It manages the message traffic across all of the processors. When the scale becomes larger, the communication overhead would be a problem. In order to optimize the communication performance, we have to decrease overhead, routing delay, channel occupancy, and contention delay.

**Parallel Performance Theory**

1. What distinguishes Amdahl's Law" from Gustafson-Baris' Law" in respects to parallel speedup?

Under the Amdahl’s Law the speedup is determined by the degree of sequential execution time in the computation (not by number of processors); but the Gustafson-Baris’s Law take into account of the number of processors and can maintain or increase parallel efficiency as the problem scales.

2. For a given problem size, why does the efficiency go down as the number of processing elements increase? Is this always true?

The efficiency goes down as the number of processing elements increase because perfect efficiency is hard to achieve and some algorithm take more effort to do in parallel;

In order to maintain efficiency, one must balance the workload among all its processing elements.

It is not always true, when the data is easier to fit into more processors that can decrease number of I/O and hence increase the efficiency.

3. Suppose you are comparing 2 algorithms, A and B, for the same problem. Suppose that algorithm A has better strong scaling than algorithm B on a parallel machine.

(a) Will algorithm A always has better weak scaling than algorithm B on this machine?

Explain.

**No**,

Strong scaling is applied when the problem size stays fixed but the number of processing elements are increased, and

Weak scaling is applied when the problem size assigned to each processing elements stays constant and additional elements are used to solve a larger total problem.

If algorithm B is to run on a larger problem size (required a lot of memory or other system resources) than algorithm A, then the weak scaling of B might be higher.

(b) Is it possible that algorithm B will have better strong scaling than algorithm A on a different parallel machine?

**Yes**,

If on a different parallel machine is more suitable (has more system resources for B to run), then the strong scaling of B might exceed A.

4. Amdahl's Law is defined with respect to the fastest sequential execution time. Suppose the fastest sequential algorithm on machine A, SA, is not the fastest sequential algorithm on machine B, SB. Does this cause a problem when comparing speedups between the two machines? Explain.

**Yes**, it does cause a problem when comparing speedups between the two machines.

First of all, the question does not specify whether the algorithms are solving the same problem. If the problem solving on one of the machine is more scalable than the other, then to compare the machines with the fastest sequential algorithms is not fair.

Second of all, when we compare the two machines, we should setup a set of algorithms to test the machine against the set of algorithms to see the speedup.

It is difficult to compare machines in general.

5. Consider the problem of computing the dot product of two vectors, A and B, each of length N.

(a) Describe how you would parallelize this problem.

We would use map and reduce to parallelize this problem. We can a map pattern to get the product of Ai \* Bi and do a sum reduction to get the sum of all the Ai \* Bi.

(b) Assume that multiplying two numbers takes 4 units of time, adding two numbers takes 2 units of time, and communicating one number between two processing elements takes 50 units of time. What is the parallel runtime, speedup, and efficiency of your parallel algorithm when run on P processing elements? You can assume N and P are a power of 2. If you can, write your answer in terms of computation time and communication time components.

T1 = n \* (4 units (for multiplying Ai \* Bi) + 2 units (for added the product of i iteration to the total sum)) = 6n units.

Tp = 6n/p (the time for parallelized mult map) + (50 + 2)lg(p) (the time for communication and addiction)

Sp = 6n / (6n/p + 52lg(p))

Ep = 6n / (6n/p + 52lg(p)) / p(c) Calculate the speedup and efficiency assuming that the problem for P=1 is that of computing the dot product for two vectors of length 256. Use P=1, 4, 16, 64, and 256, and assume the same time costs as in b.

|  |  |  |
| --- | --- | --- |
| P | Speedup | Efficiency |
| 1 | 1 | 1 |
| 4 | 3.15 | 0.79 |
| 16 | 5.05 | 0.32 |
| 64 | 4.57 | 0.07 |
| 256 | 3.64 | 0.14 |

(d) When executing on 64 processors, how large would N have to be to achieve the same efficiency as achieved on 4 processors for N=256?

0.79 = 6n / (6n/64 + 52lg(64)) / 64

n = 12519