CS546 "Parallel and Distributed Processing"
Homework 1

1. **(7 points) What are the two main styles of parallelism? Explain.**

Ans: The two main styles of parallelism are

1) Partitioning: This type of parallelism focuses dividing the problem or sequences into independent steps and then parallelising them. This follows divide and rule policy.
    a. Data partitioning : In this the data of problem is divided into smaller chunks and performs operation on them. In these cases usually there is a need to assimilate and group the data back as a whole.
    b. Function partitioning: In this model the operations or functions are partitioned such that each partition or operation can independently work on the task it is assigned and can give back the process info back to main process.
2) Parallelism
    a. Concurrent: This model basically focusses on parallelising independent parts of a solutions concurrently.
    b. Pipelined: In this model the instructions or operations are pipeline in such a way that as soon as the dependency is met the instruction is executed and it doesn't have to wait for a full clock cycle to complete.

2. **(7 points) What are the two main types of locality? Explain.**

Ans: The two types of locality are

1) Temporal Locality: It is the concept that a resource that is referenced at one point will be referenced again sometime in the near future. It determines sensitivity to cache size. For a good Temporal Locality, the cache miss traffic should decrease fast when cache size increases. An example of Usage for Temporal Locality is a lower level cache as it is generally used to keep recently referenced data and data near recently referenced data, a specially designed, faster but smaller memory area, which can lead to potential performance increases.
2) Spatial Locality: It is the concept that a resource that is referenced then the likelihood of the next resource to be near the previous resource is higher. It determines sensitivity to line size. For a good Spatial Locality, the cache miss traffic should not increase much with increase in line size. An example of usage of Spatial Locality is at higher level cache where data-elements are brought into cache one cache-line at a time which means if one element is referenced, few neighbouring elements are also brought into the cache.

3. **(7 points) What are the three basic programming paradigms for parallel processing? Explain.**

Ans: The three basic programming paradigms for parallel processing are

1) Shared Memory: This programming model is an effective means of inter-process data passing. In this model, there is a global memory space which is shared by the processes where the processes can read and write asynchronously. Many parallel programming languages can exploit this shared memory support directly. The asynchronous access done on memories lead to race conditions and is handled using synchronizations techniques.

2) Distributed Memory: In this programming model, each processor has its own memory space where it reads and write. Each processor has the full bandwidth of their memory without inference from other processors. It is an extremely scalable model as there is no common bus. In this model each processor communicates with each other using messages. This model is most effective in cases where the processes can be divided into independent task with no communication with other task. The factors that needs to be considered for Distributed memory are

    a. Link Bandwidth: the number of buts that can be transmitted per unit of times.
    b. Message transfer through network

3) GPU/GPGPU(CUDA): It is a parallel computing model which handles computation for computer graphics. It has a number of computational resources like \
    a. Programmable Processors: It allows programmers to perform kernel on stream of data
    b. Rasterizer: create fragments and per vertex constants like texture, colour
    c. Texture Unit: read-only memory interface
    d. Framebuffer: write only memory interface.

4. **(7 points) Discuss the difference between shared address space machines and distributed address space machines. Discuss the advantages and disadvantages of both architectures.**

Ans: The difference between Shared Memory machine and Distributed Memory Machines are as follows:

Shared Memory machine

a. All processors see a global shared address space
b. Ability to access all memory from each processor – A write to a location is visible to the reads of other processors

Distributed Memory Machines

a. No global shared address space – Send and receive variants are the only method of communication between processors (much like networks of workstations today, i.e. clusters)
b. Suitability of the hardware depends on the problem to be solved as well as the programming model

Advantages of Shared memory are

e. User-friendly programming perspective to memory
f. Lower latency
g. Inter-process communication is fast and uniform.

Disadvantages of shared Memory are

a. Lack of scalability
b. Programmers need to synchronise to ensure correct access of global memories.

Advantages of Distributed Memory are

a. High Scalability

b. No lock Contention

Disadvantages of Distributed Memory are

a. Data Coherency problems.
b. Dependency on network bandwidth.

5. **(7 points) What is parallel I/O? Why do we need parallel I/O?**

Ans: Parallel I/O is performing multiple simultaneous input/output operations. One instance is performing parallel write operation by RAID array.

Parallel I/O is needed to

a) Increase I/O Bandwidth
b) Reduce time taken by I/O operations.
c) To reduce bottleneck created by I/O operations on the processors.
d) The I/O bandwidth is limited.
e) All HPC Applications do I/O for
   i) Reading initial conditions or datasets for processing
   ii) Writing numerical data from simulations
f) The relative speed of IO with memory and processor is very slow

6. **(7 points) Give an example of anti-dependence and give a corresponding solution to remove the dependence.**

Ans: Anti-Dependency

1. B=A*A
2. A=5

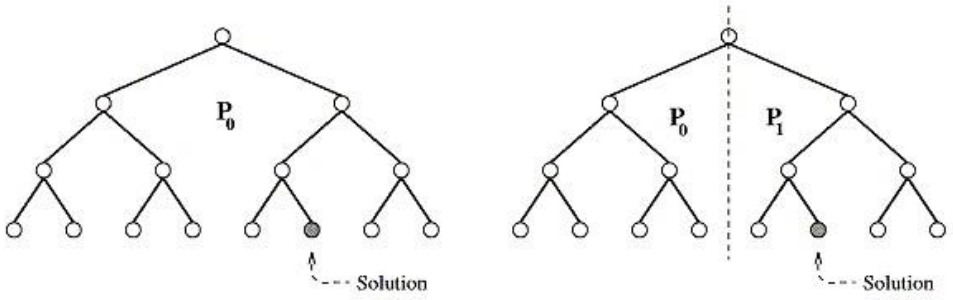
To remove we can

N Temp=A

1. B=Temp*Temp
2. A=5

Now 1 and 2 are independent but 1. Is truly dependent on N hence it cannot be removed.

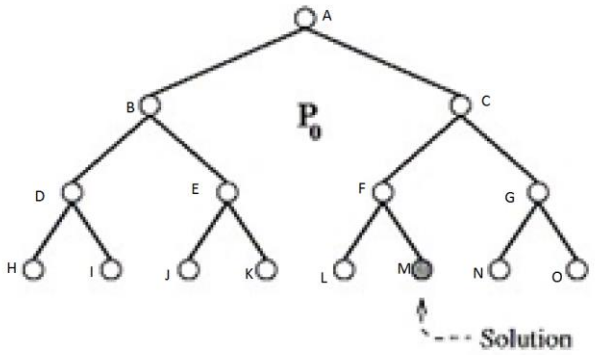
7. **(10 points) Consider the search tree shown in the following figure, in which the dark node represents the solution.**



(a) DFS with one processor (b) DFS with two processors

a) **If a sequential search of the tree is performed using the standard depth-first search (DFS) algorithm, how much time does it take to find the solution if traversing each arc of the tree takes one unit of time? Note: check this on how DFS works.**

Ans:



(a) DFS with one processor

In this case the processor traverses through

A -> B -> D -> H -> D -> I = 5

I -> D -> B -> E -> J -> E = 5

E -> K -> E -> B -> A -> C = 5

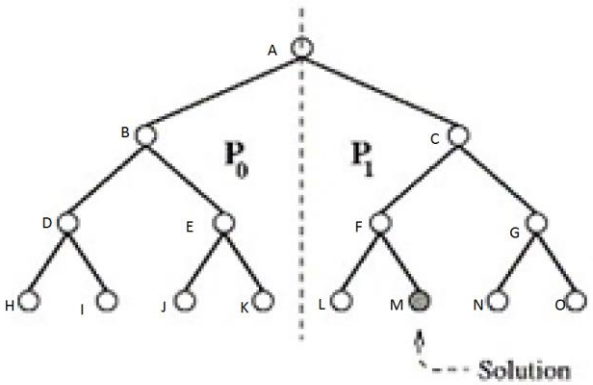
C-> F -> L -> F -> M = 4

Total = **19**

19 units of time

b) **Assume that the tree is partitioned between two processing elements that are assigned to do the search job, as shown in figure b. If both processing elements perform a DFS on their respective halves of the tree, how much time does it take for the solution to be found? What is the speedup? Is there a speedup anomaly? If so can you explain the anomaly?**

Ans:



(b) DFS with two processors

In this case

P_0	P_1
A -> B	A->C
B -> D	C -> F
D -> H	F -> L
H -> D	L -> F
D -> I	F -> M **(found)**

Here the P_1 finds the element M in 5 units of time. The speed up than a uniprocessor case is about 19-5=14 units faster. This happened as the data was partitions and given to multiple processors for parsing in parallel. This is a classic case of Data Parallelism.

8. **(10 points) Derive the formula for calculating the average access time for a word in a system with three levels of cache. Assume the following values for a theoretical system containing an L1, L2, and L3 cache.**

Location	Latency
L1	5 ns
L2	10ns
L3	35ns
Main Memory (RAM)	100 ns

If an application has following hit rates, what is the average memory access time for a memory word?

Location	Hit Rate
L1	45%
L2	75%
L3	90%
Main Memory (RAM)	100%

Ans: In order to find avg memory access time we have the formula :

Tavg = h*Tc +(1-h)*M

where h = hit rate
(1-h) = miss rate
Tc = time to access information from cache
M = miss penalty (time to access main memory)

Extending this logic to three level cache
h1: Hit Rate for L1 cache
h2: Hit Rate for L2 cache
h3: Hit Rate for L3 cache

t1: Time to access L1
t2: Time to access L2
t3: Time to access L3
t4: Time to access Main Memory

Tavg for L1= Hit Rate for L1 * Time to access L1 + Miss Rate of L1 (Avg Time to access L2)
Tavg for L2= Hit Rate for L2 * Time to access L2 + Miss Rate of L2 (Avg Time to access L3)
Tavg for L3= Hit Rate for L3 * Time to access L3 + Miss Rate of L3 (Time to access Main Memory)
Combining the three we get

Tavg = h1* t1 + (1-h1)*(h2*t2 + (1-h2)*(h3*t3 + (1-h3)*t4))
With the given values for t ie latencies

Tavg = h1* 5 + (1-h1)*(h2*10 + (1-h2)*(h3*35 + (1-h3)*100))

Now with the Hit Rates for application given

Tavg = .45* 5 + (1-.45)*(.75*10 + (1-.75)*(.9*35 + (1-.9)*100))
= 17.875

9. **(7 points) Explain how a queue, implemented in hardware in the CPU, could be used to improve the performance of a write-through cache.**

Ans: The write-through implementation of cache solves the data inconsistency problem there in the writing in cache but it forces every write to go to main memory hence it uses up the bandwidth between cache and memory. So to improve this we include a **write buffer** which queues pending writes to the main memory and permits CPU to continue. This saves the bandwidth. Also write buffers free the cache to service read requests while the write is taking place

10. **(7 points) Recall the example involving cache reads of a two-dimensional array (figure below). How does a larger matrix and a larger cache affect the performance of the two pairs of nested loops? What happens if MAX = 8 and the cache can store four lines? How many misses occur in the reads of A in the first pair of nested loops? How many misses occur in the second pair?**

Cache Line	Elements of A			
0	A[0][0]	A[0][1]	A[0][2]	A[0][3]
1	A[1][0]	A[1][1]	A[1][2]	A[1][3]
2	A[2][0]	A[2][1]	A[2][2]	A[2][3]
3	A[3][0]	A[3][1]	A[3][2]	A[3][3]

Ans: Cache can contain 4 elements per cache-line [as per original problem] and also assuming no element was present in the cache initialy the first miss happens at A[0][0] after which the system

will read 4 elements into the cache A[0][1], A[0][2],A[0][3] after that when it reads A[0][1], A[0][2],A[0][3] these elements are in cache hence it reads from there and then when it needs A[0][4] again a cache miss happens(2 misses). System now brings A[0][4], A[0][5], A[0][6], A[0][7]. Now it reads till end of inner loop till A[0][7] hence for each new outer loop there are 2 misses. As There only 4 lines hence after every 2 outer loops the 3rd and 4th need to clear the earlier cache tables.

Total misses for 1st pair of loops= 8*2=16 misses

For The second pair of loops, as cache contains last two rows of elements of A it goes to a cache miss for A[0][0] and it brings A[0][1], A[0][2],A[0][3] into cache in line 1. Now it needs A[1][0] which is again not in cache hence it removed line 2 from cache and replaces it with A[1][1], A[1][2],A[1][3]. Now it needs A[2][0] which got removed by the previous two misses hence again there is a cache miss (3 misses).and it brings A[3][1], A[3][2],A[3][3] into line 3 of cache. Again when it access A[3][0] it was removed by previous misses hence there is a cache miss again (4 misses). This will continue for all as when it accesses A[4][0], A[5][0], A[6][0] and A[7][0], it will replace the whole grid with their adjacent values of A hence the next outer loop of A[0][1] till A[7][1] all will miss. Hence

Total Misses in 2nd Pair of loops is 64.

11. **(7 points) Explain why the performance of a hardware multi-threaded processing core might degrade if it had large caches and it ran many threads.**

Ans: The performance of multi-threaded processing core will degrade if it ran many threads as the system will take extra time to schedule and de-schedule cores and will hence add to the overall run-time. Also in case of Multi Threads the threads can interfere with each other when sharing hardware such as cache as it gives raise to lock contentions plus cache coherency which increase a lot of overhead.

12. **(10 points)**
    a) **A planar mesh is just like a toroidal mesh, except that it doesn't have the "wraparound" links. What is the bisection width of a square planar mesh?**
    b) **A three-dimensional mesh is similar to a planar mesh, except that it also has depth. What is the bisection width of a three-dimensional mesh?**

Ans: Bisection width is the amount of communication that can take place between two groups after dividing the topology in 2 equal halfs.

a) In case of 2d planar mesh of 16 processors if we remove all the middle links(4) there can be no communication between the halves ie 4 that is the bisection width. If p is processors such that $p=q^2$ then bisection width = $\sqrt{p}$
b) In case of 3d mesh is let p processors such that $p=q^3$ then bisection width = $\sqrt[3]{p}$

13. **(7 points) Modify the trapezoidal rule so that it will correctly estimate the integral even if `comm_sz` doesn't evenly divide n (you can still assume that `n >= comm_sz`).**

```
double Trap(double leftEndPoint,double rightEndPoint,int count){
    double estimate=0.0;
    double xInIntervalI=leftEndPoint;
    double h=0.0;
    int i=0;
    for(i=0;i<count-1;i++){
        h=getHeightForInterval(leftEndPoint,rightEndPoint,i);
        estimate=h*(f(xInIntervalI)+f(xInIntervalI+h))/2;
    }
    return estimate;
}
```

Note: We encourage collaboration between you and your classmates. Discuss various approaches and techniques to better understand the questions. However, we do NOT allow copying solutions or code. This is considered as cheating and falls under IIT code of honor. Penalties will be enforced. Please make sure you write your own solutions.

GOOD LUCK!