COMP137 Assignment 1

Sloan Kim

1. Consider a fast‐food restaurant such as In‐N‐Out or McDonald’s, and treat each member of the restaurant’s staff as a processing core.
2. Identify at least two ways in which the staff utilize task-parallelism.

Workers will divide their work based on task, pipelining

Someone will fry burger, someone will fry French fries, someone will wash lettuce, etc.

1. Identify at least two ways in which the staff utilize data-parallelism.

Workers will divide the load of the work.

Everyone will fry 30lb of burger, once finished, all will change to frying French fries.

1. Recall the example involving cache reads of a two-dimensional array (Parallel Hardware Slides, title of slide “Caches and programs”).

* How does a larger matrix and a larger cache affect the performance of the two pairs of nested loops?

Larger matrix increases cache miss but larger cache decreases cache miss

* What happens if MAX = 8 and the cache can store four lines?

4 times more cache misses occur for both fairs

* How many misses occur in the reads of A in the first pair of nested loops?

4 cache misses occur (A[0][0], A[2][0], A[4][0], A[6][0])

* How many misses occur in the second pair?

32 cache misses occur (A[0][0], A[2][0], A[4][0], A[6][0], A[0][1], A[2][1],….)

1. In the lecture on Week2 Day2, we looked at an example of pipelining using floating point addition. In our example, we made an assumption that each function unit took the same amount of time. Suppose that fetch and store took 2 nanoseconds (the other operations only take 1 nanosecond).
2. How long does the floating point addition take with these assumptions?

9 nanoseconds

1. How long will an unpipelined addition of 1000 pairs of floats take under these assumptions?

9000 nanoseconds

1. How long will a pipelined addition of 1000 pairs of floats take under these assumptions?
2. oseconds
3. Suppose that a fetch from a level 1 cache takes 2 nanoseconds, while a fetch from a level 2 cache takes 5 nanoseconds, and fetch from the main memory takes 50 nanoseconds. What happens to the pipeline when there is a level 1 miss and a level 2 hit on a fetch of one of the operands? What happens when there is a level 1 and level 2 miss?

When there is a L1 miss and a L2 hit, total time to fetch the operands will take 7 nanoseconds because it takes 2 nsec to fetch from L1 but miss, takes 5 more nsec to fetch from L2. The pipeline will take 13 nsec assuming other units take 1 nsec each.

When there is L1 miss and L2 miss, total time to fetch the operands will take 57 nsec.

1. Consider a program with the following characteristics.

Ts = 20n^2 + C

Tp=20n^2/p + C + D

C = 100

D = 10 logp

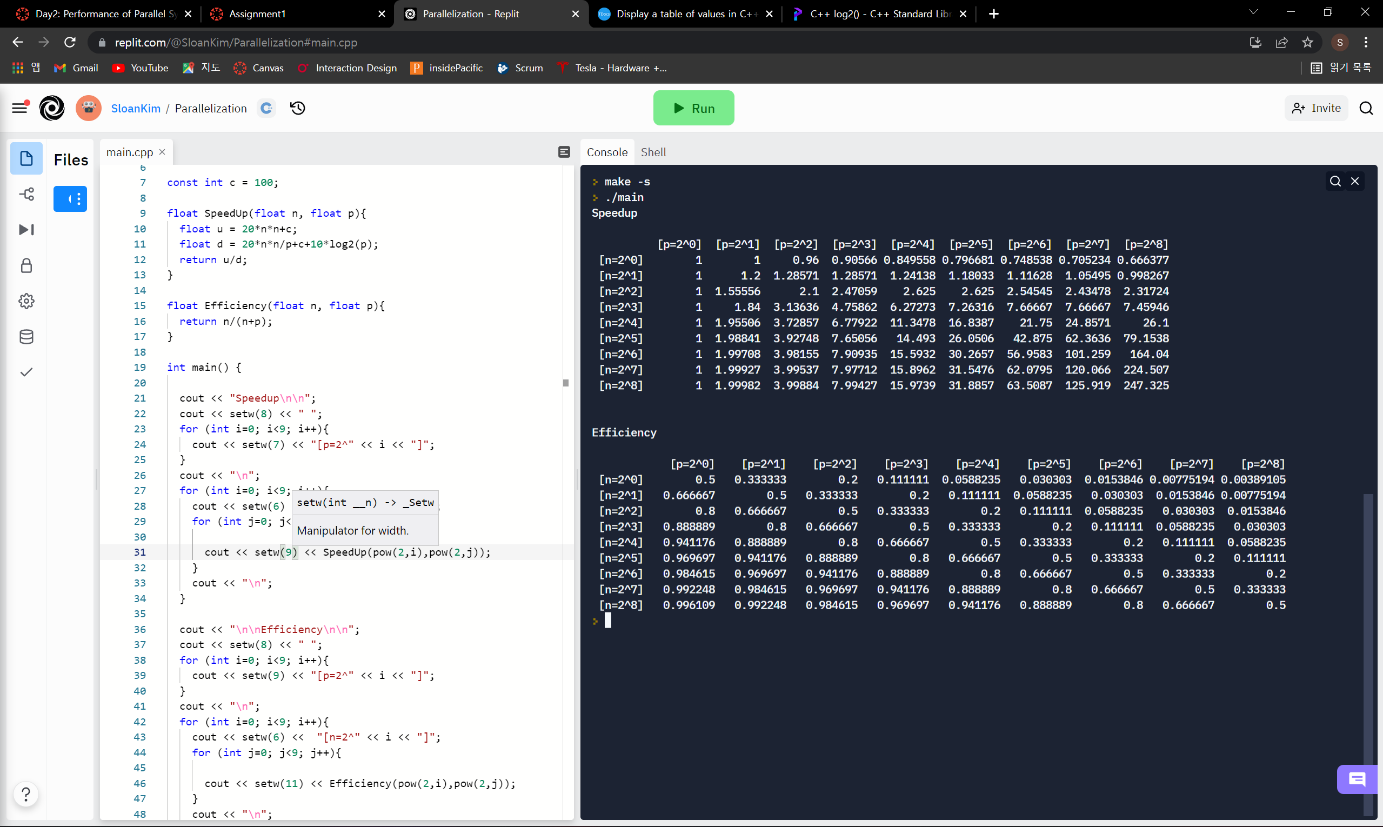
1. Does the term C result from non-parallelizable code or from parallelization overhead?

Non-parallelizable code

1. Does the term D result from non-parallelizable code or from parallelization overhead?

Parallelization overhead

1. Compute the speedup and efficiency of the program for n = 2^0, 2^1, …2^8 and p = 2^0, 2^1,…2^8. Display that data in a table (or tables). (Writing a short program should be more efficient than hand calculating 162 values.)



1. Is the program strongly scalable? Is it weakly scalable?

It is weakly scalable as the efficiency does not decrease when p and n are increasing in same rate.

1. Repeat questions 1(c) and 1(d) for C=100 and D = 100p

텍스트, 컴퓨터, 모니터, 스크린샷이(가) 표시된 사진

자동 생성된 설명

It is weakly scalable as well.

1. Consider the following serial algorithm to find the largest value in an array.

FindLagest Algorithm

1. Design a parallel algorithm to solve the same problem. Assume that a and n are in local memory for each core when the algorithm starts. Each core also has a local variables p and my\_id, where p is the number of cores and my\_id is the core’s unique identifier (0 ≤ my\_id ≤ p). Also assume that p is much smaller than n and that p is a power of 2.
2. One core divide array a in p parts
3. Depending on the my\_id, each core finds the largest number out of a section of array a, using serial algorithm given.
4. Store the largest number in each section into shared memory.
5. Fetch two numbers each in cores from shared memory.
6. Repeat [

each core compares two numbers

Store the larger number into shared memory

Fetch the two numbers each in cores from shared memory

] until one largest number is left

1. Indicate whether your solution is for a shared-memory or a distributed-memory system.

It is for a shared-memory system.