SYSC5704 – ELEMENTS OF COMPUTER SYSTEMS

ASSIGNMENT-6 REPORT

Submitted by

Aarthi Thirumavalavan

Student ID. 100958800

Master of Engineering. Electrical and Computer Engineering

Carleton University

6.2.

1. Your job is to cook 3 cakes as efficiently as possible. Assuming that you only have one oven large enough to hold one cake, one large bowl, one cake pan, and one mixer, come up with a schedule to make three cakes as quickly as possible. Identify the bottle necks in completing this task.

Solution : For this set of resources, we can pipeline the preparation. We assume that we do not have to reheat the oven for each cake.

Preheat Oven

Mix ingredients in bowl for Cake 1

Fill cake pan with contents of bowl and bake Cake 1. Mix ingredients for Cake 2 in bowl.

Finish baking Cake 1. Empty cake pan. Fill cake pan with bowl contents for Cake 2 and bake Cake 2. Mix ingredients in bowl for Cake 3.

Finish baking Cake 2. Empty cake pan. Fill cake pan with bowl contents for Cake 3 and bake Cake 3.

Finish baking Cake 3. Empty cake pan.

2. Assume now that you have three bowls, 3 cake pans and 3 mixers. How much faster is the process now that you have additional resources?

Solution : Now we have 3 bowls, 3 cake pans and 3 mixers. We will name them A, B and C.

Preheat Oven

Mix ingredients in bowl A for Cake 1

Fill cake pan A with contents of bowl A and bake for Cake 1. Mix ingredients for Cake 2 in bowl A.

Finish baking Cake 1. Empty cake pan A. Fill cake pan A with contents of bowl A for Cake 2. Mix ingredients in bowl A for Cake 3.

Finishing baking Cake 2. Empty cake pan A. Fill cake pan A with contents of bowl A for Cake 3.

Finish baking Cake 3. Empty cake pan A.

The point here is that we cannot carry out any of these items n parallel because we either have one person doing the work, or we have limited capacity in our oven.

6.4

1. How many cycles does it take for all instructions in a single iteration of the above loop to execute?

Solution : This is a straightforward computation. The first instruction is executed once, and the loop body is executed 998 times.
Version 1—17,965 cycles
Version 2—22,955 cycles
Version 3—20,959 cycles

2. When an instruction in a later iteration of a loop depends upon a data value produced in an earlier iteration of the same loop, we say that there is a loop carried dependence between iterations of the loop. Identify the loop-carried dependences in the above code. Identify the dependent program variable and assembly-level registers. You can ignore the loop induction variable `j`.

Solution: Array elements D[j] and D[j−1] will have loop carried dependencies. These will f3 in the current iteration and f1 in the next iteration.

3. Loop unrolling was described in Chapter 4. Apply loop unrolling to this loop and then consider running this code on a 2-node distributed memory message passing system. Assume that we are going to use message, where we introduce a new operation send (x, y) that sends to node x the value y, and an operation receive( ) that waits for the value being sent to it. Assume that send operations take a cycle to issue (i.e., later instructions on the same node can proceed on the next cycle), but take 10 cycles be received on the receiving node. Receive instructions stall execution on the node where they are executed until they receive a message. Produce a schedule for the two nodes assuming an unroll factor of 4 for the loop body (i.e., the loop body will appear 4 times). Compute the number of cycles it will take for the loop to run on the message passing system.

Solution : This is a very challenging problem and there are many possible implementations for the solution. The preferred solution will try to utilize the two nodes by unrolling the loop 4 times (this already gives you a substantial speedup by eliminating many loop increment, branch and load instructions. The loop body running on node 1 would look something like this (the code is not the most efficient code sequence):

```
DADDIU r2, r0, 996
L.D f1, –16(r1)
L.D f2, –8(r1)
```

loop:
ADD.D f3, f2, f1
ADD.D f4, f3, f2
Send (2, f3)
Send (2, f4)
S.D f3, 0(r1)
S.D f4, 8(r1)
Receive(f5)
ADD.D f6, f5, f4
ADD.D f1, f6, f5
Send (2, f6)
Send (2, f1)
S.D. f5, 16(r1)
S.D f6, 24(r1)
S.D f1 32(r1)
Receive(f2)
S.D f2 40(r1)
DADDIU r1, r1, 48
BNE r1, r2, loop
ADD.D f3, f2, f1
ADD.D f4, f3, f2
ADD.D f6, f5, f4
S.D f3, 0(r1)
S.D f4, 8(r1)
S.D f5, 16(r1)
The code on node 2 would look like this:
DADDIU r3, r0, 0
loop:
Receive (f7)
Receive (f8)
ADD.D f9, f8, f7
Send(1, f9)
Receive (f7)
Receive (f8)
ADD.D f9, f8, f7
Send(1, f9)
Receive (f7)
Receive (f8)
ADD.D f9, f8, f7
Send(1, f9)
Receive (f7)
Receive (f8)
ADD.D f9, f8, f7
Send(1, f9)
DADDIU r3, r3, 1
BNE r3, 83, loop

Basically Node 1 would compute 4 adds each loop iteration, and Node 2 would compute 4 adds. The loop takes 1463 cycles, which is much better than close to 18K. But the unrolled loop would run faster given the current send instruction latency.

4. The latency of the interconnect network plays a large role in the efficiency of message passing systems. How fast does the interconnect need to be in order to obtain any speedup from using the distributed system described in Exercise 6.4.3?

Solution : The loop network would need to respond within a single cycle to obtain a speedup. This illustrates why using distributed message passing is difficult when loops contain loop-carried dependencies.

6.20.
1. On average, how many requests are being processed at any given instant?

Solution :
So for a max transaction processing rate of 5000/sec, and we have 4 cores contributing, we would see an average latency of .8 ms if there was no queuing taking place. Thus, each core must have 1.25 transactions either executing or in some amount of completion on average.
So the answers are:

Latency	Max TP rate	Avg. # requests per core
1ms	5000/sec	1.25
2ms	5000/sec	2.5
1ms	10,000/sec	2.5
2ms	10,000/sec	5

2. If move to an 8-core system, ideally, what will happen to the system throughput (i.e., how many queries/second will the computer process)?

Solution :

We should be able to double the maximum transaction rate by doubling the number of cores.

3. Discuss why we rarely obtain this kind of speedup by simply increasing the number of cores.

Solution :

The reason this does not happen is due to memory contention on the shared memory system.