Eric Tashji

Homework Assignment 5

Problem 1: N-cube networks are known for their ability to sustain broken links and still provide connectivity between the switches.

- How many links in 1-cube can fail and we still have full connectivity?

  Solution: A 1-cube has 2 (2^1) nodes, each with 1 link and 1 neighbor node. Therefore, **a max of 0 links can fail** and connectivity will be maintained because each node only has one connection to another node. If a single link fails, the two nodes will be disconnected from one another.

- How many links in 2-cube can fail and we still have full connectivity?

  Solution: A 2-cube has 4 (2^2) nodes, each with 2 links and 2 neighbor nodes. Therefore, only **one bidirectional link can fail**, and the 2-cube would still have full connectivity. This is because each node is connected to 2 other nodes, so if one fails, that node will still be connected to another node.

- How many links in 3-cube can fail and we still have full connectivity?

  Solution: A 3-cube has 8 (2^3) nodes, each with 3 links and 3 neighbor nodes. Therefore, a max of 2 bidirectional links can fail without losing connectivity. This is because if each node is connected to 3 other nodes, then **2 bidirectional links** can fail, and the node would still be connected to its third node.

- How many links in N-cube can fail and we still have full connectivity? Give your answer in terms of N.

  Solution: A N-cube has 2^N nodes, each with N links and N neighbor nodes. Therefore, a max of N-1 links can fail and connectivity will be maintained, because if each node is connected to N nodes, then **N-1 bidirectional links** can fail and connectivity will be maintained because the nodes would still be connected to at least 1 node.

Problem 2 on the next page.

Eric Tashji

Problem 2: Consider the following code, which multiplies two vectors that contain single precision floating point values:

```
For (i = 0; i < 100; i++) {
        C[i] = a[i] * b[i]
}
```

We define the Arithmetic Intensity of a code as the number of operations to run the program (for example in C) divided by the number of bytes accessed in the main memory. Arithmetic Intensity is a good measure of arithmetic operations versus the number of bytes transferred between memory and CPU so we can see whether the system is computation bound or memory bandwidth bound as we discussed in class. What is the arithmetic intensity of the above code?

Solution: The above code will perform 100 operations, as the for loop will execute 100 times and the for loop only contains 1 multiplication operation. Each time the loop iterates, it makes 3 memory accesses: 1 read for a[i], 1 read for b[i], and 1 write for C[i]. Each time it makes a memory access, it accesses 8 bytes. Therefore, if the loop iterates 100 times, we multiply the number of times the loop iterates times the number of access times the number of bytes accessed per access. Therefore, if we do 100 * 3 * 8, we get 2400 bytes. Then, we divide the number of operations by the number of bytes to find the Arithmetic Intensity. So, if we do 100/2400, we get 1/24, or .0417. **Therefore, the Arithmetic Intensity is .0417.**

Problem 3: Assume a GPU that runs 1.5 GHz with 16 SIMD processors, each having 16 single-precision FP units. This GPU is supported by a 100GB/s off-chip memory. Assume all memory latencies can be hidden. Assume each operation is an addition.

- Ignoring memory bandwidth, what is the peak SP FP operation in GFLOP/sec.?

  Solution: The peak SP FP operation in GFLOP/sec is equal to the clock speed times the number of SIMD processors times the number of SP FP units. If we do 1.5 * 16 * 16, we get 384 GFLOP/sec. **Therefore, the peak SP FP operation of this GPU is 384 GFLOP/sec.**

- Is this throughput sustainable given the bandwidth? Justify your answer.

  This throughput is not sustainable given the bandwidth. This is because an addition operation requires 2 4 byte operands for the input and a 4-byte operand for the output, equaling a total of 12 bytes. Therefore, it needs 12 * 384 or 4,608GB/sec, making the throughput unsustainable.

Eric Tashji

Problem 4: Assume a GPU architecture that contains 16 SIMD processors and has a clock speed of 4 GHz. Each SIMD can produce 24 single precision arithmetic results on its 8 lanes every cycle.

Now assume we run a program on this GPU that on average 80% of threads are active. Compute the throughput, in GFLOP/sec, for this code on this GPU.

Solution: We can calculate the throughput by multiplying the clock speed times the number of SIMD processors times the number of SP FP units. The SP FP are equivalent to the number of single-precision arithmetic results. However, because we are calculating throughputs where only 80% of threads are active, we then multiply that result by .8. So, if we do 4 * 16 * 24 * .8 we get 1228.8 GFLOPs/sec. **Therefore, the throughput is 1228.8 GFLOPs/sec.**