ECE 4750 PSET 3

Tim Yao (ty252)

Nov 6, 2015

Worked with Gautam Ramaswamy, Gaurab Bhattacharya, and Sacheth Hegde.

1 Tree Network Topologies

Part	Topology	B_c	b (bits/cycle)	γ_{max}	Θ_{term} (bits/cycle)
Part 1.A	**Baseline Tree Topology**	2	32	2	16
Part 1.B	**Fat-Tree Topology**	8	32	1	32

Figure 1: Ideal Terminal Throughput for Tree Topologies

Part	Topology	H_D	H_r	t_r (cycles)	H_c	t_c (cycle)	L/b (cycle)	t_0 (cycle)
Part 1.A	**Baseline Tree Topology**	4	3	2	2	1	3	11
Part 1.B	**Fat-Tree Topology**	4	3	3.83	2	1	3	16.5

Figure 2: Zero-Load Latency for Tree Topologies

1.a Performance of Baseline Tree Topology

To calculate the minimum bisection channel count, I first found the minimum bisection cut. This happens to be the cut on the channels that connect the two level-2 routers. This cut cuts 2 channels, so the B_c is 2.
To calculate the max channel, there are 3 possibilities. It can be the channels connecting between the nodes and the level-1 routers, level-1 and level-2 routers, and between the two level-2 routers. Because we are dealing with uniform random routing, all channels of one level has the same load.
The channel load for channels between nodes and level-1 routers are 1. In uniform random routing, the amount traffic sent from and to a node is always equal to 1.
The channel load for channels between level-1 and level-2 routers are 1.5. To calculate this, I first look at the amount of traffic that is sent from one node to all other nodes that require crossing the this channel. For example, all traffic that go from node 0 to 2,3,4,5,6,7 need to cross this channel. Each unit of traffic is 1/8, so for each node that resides under (in the sense of a tree) that channel is 6/8. Because there are 2 nodes (ie. node 0 and 1) under that channel, the max channel load is 2*6/8 = 12/8 = 3/2.
The channel load for channels between level-2 routers are 2. We use the same idea from the previous calculation to calculate this max load. Each node on the left side needs to send to 4 nodes on the right side, therefore, each node contributes 4*1/8 = 1/2 unit of traffic. There are 4 nodes, so the max load is 4*1/2 = 2. From this, we see that the max channel load is between the two level 2 routers. And using the ideal terminal throughput equation, $\Theta_{term}=B_c/\gamma_{max}$, I calculated that Θ_{term}=32/2=16.

The network diameter is 4 router hops. One example is the path going from node 0 to node 7. This minimum path takes 4 router hops.
The average number of router hops is calculated using the following method. The average number of router hops for going from 0 to itself and all other nodes is the same from 1 to itself and all other nodes, from 2 to itself and all other nodes, and etc. Therefore, we can calculate the overall average number of router hops by simply analyzing the traffic going from 1 node to itself and all other nodes. If we pick 0, then the analysis is as following:

0->0: 1
0->1: 1
0->2: 3
0->3: 3
0->4: 4
0->5: 4
0->6: 4
0->7: 4
The average is then: (1+1+3+3+4+4+4+4)/8=24/8=3 hops.
Because all of the routers are radix-2, the average per-hop router latency is 2.
The method to calculate the average channel hop count is the same as calculating the average router hop count.
0->0: 0
0->1: 0
0->2: 2
0->3: 2
0->4: 3
0->5: 3
0->6: 3
0->7: 3
The average is then: (2+2+3+3+3+3)/8=16/8=2 hops.
The average per-hop channel latency is 1 cycle as stated in the assumption.
Because the message is 96 bits and the channel width is 32 bits, the serialization latency is 96/32=3 cycles.
Now, we can calculate the zero-load latency.
t_0=$H_r t_r$+$H_c t_c$+L/b
t_0=3*2+2*1+3=11 cycles

1.b Performance of Fat-Tree Topology

We use the same approach as in the previous section to calculate the max channel load and ideal terminal throughput for the baseline tree topology. Again, the channels connecting nodes to level-1 routers have a load of 1. The total load on all channels between level-1 and level-2 routers are the same (because this is a uniforma random traffic pattern), but now there are twice the number of channels. Therefore, the channel load is 6/8=3/4. The total load on all channels between the two level-2 routers are also the same, but because there are now 4 times the number of channels, the channel load is 1/2. Therefore, the max channel load is on the channels between nodes and level-1 routers. The max channel load is 1 and the ideal terminal throughput is 32/1=32.

Because the layout topology and channel width of the fat-tree is the same as the baseline, the network diameter, average router hop count, average channel hop count, average per-hop channel latency, and serialization latency are the same.
The only difference here is the average per-hop router latency due to different radix routers. To calculate this, we take a similar approach to calculating the average hop counts. We use a weighted average for each node-pair due to the different latencies of different routers.
0->0: 3
0->1: 3
0->2: 3+5+3 = 11
0->3: 3+5+3 = 11
0->4: 3+5+5+3 = 16
0->5: 3+5+5+3 = 16
0->6: 3+5+5+3 = 16
0->7: 3+5+5+3 = 16
The average is: (3+3+11+11+16+16+16+16)/24=92/24=23/6=3.833
From this, we can calculate the zero-load latency.
t_0=3*23/6+2*1+3=16.5 cycles

1.c Integrating Processors, Memories, and Networks

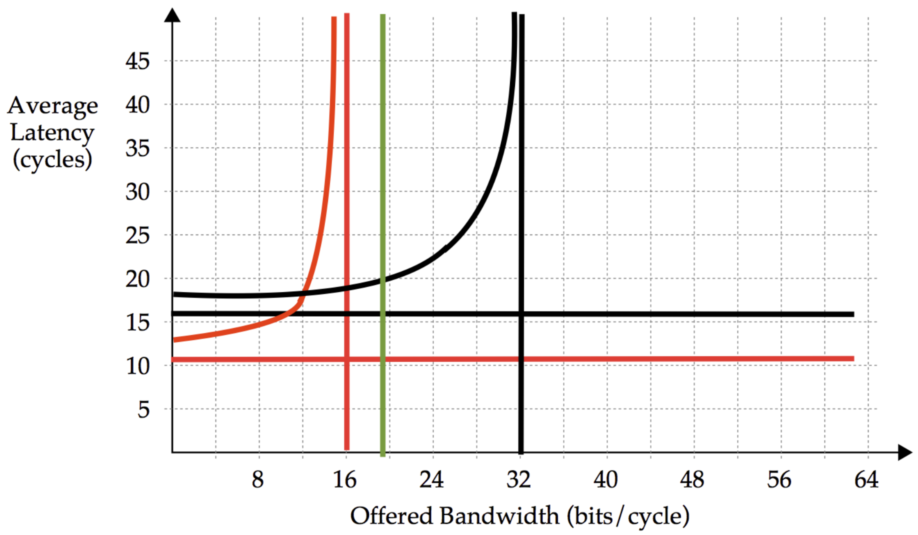

Figure 3: Average Latency versus Offered Bandwidth Curve

The red lines represent the baseline tree, the black lines represent the fat-tree, and the green line represents where the application will be located on the latency vs offered bandwidth curve.

As shown by the graph, the fat-tree will perform better on this application because the expected bandwidth of application is higher than the ideal terminal throughput of the baseline design but within the range of the fat-tree design. While the baseline design has a lower zero-load latency than the fat-tree design, it has a much lower ideal throughput than the fat-tree design. This means that if we used the baseline design, we will have to stall the processors and wait until the network is free enough to handle to the memory requests. This stalling can have a huge negative impact on the performance and completely negate the benefits of the lower zero-load latency of the baseline design. Therefore, the fat-tree will perform much better because we are operating well under its saturation point. This means that we can still achieve low latency while sustaining full expected throughput.

I do not think that this conclusion will apply across all applications, but it will apply to most. The reason is that the fat-tree design has a much higher (50%) latency than the baseline design. The purpose of a cache is to lower the memory access latency, so low latency is very important, especially for the L1 cache. The only time that fat-tree is a better design is when the program contains more than 15% of load and store word instructions. The 10% to 15% is roughly where the latency vs offered bandwidth curves of the 2 designs intersect. Therefore, for applications that does not require frequent memory accesses, the baseline design will perform better due to the lower latency. If our programs contain over 15% of load and store instructions, then the fat-tree is a better design because we will not hit network saturation until over 30% of load and store instructions. This means that we are able to sustain a decently low latency for longer. From the many assembly programs that I have seen throughout the course (VVadd, Array incrementer, etc), the amount of load and store instructions per program (or per iteration) is usually 20% to 40%. Therefore, I believe that if one were to choose only one implementation, the fat-tree will be the better choice as it will avoid the need to stall in more programs. However, as I stated earlier, the fat-tree will not always perform better than the baseline design, especially for programs with few memory accesses.