# Homework #3

**B1. You are trying to appreciate how important the principle of locality is in justifying the use of a cache memory, so you experiment with a computer having an L1 data cache and a main memory (you exclusively focus on data accesses). The latencies (in CPU cycles) of the different kinds of accesses are as follows: cache hit, 1 cycle; cache miss, 105 cycles; main memory access with cache disabled, 100 cycles.**

1. **When you run a program with an overall miss rate of 5%, what will the average memory access time (in CPU cycles) be?**

Average memory access time = (1-miss rate) \* hit time + miss rate \* miss time

= (1-5/100)\*1 + (5/100 \* 105)

= .95+0.05\*105 = 6.2 cycles

Average memory access time = **6.2 cycles**

1. **Next, you run a program specifically designed to produce completely random data addresses with no locality. Toward that end, you use an array of size 256 MB (all of it fits in the main memory). Accesses to random elements of this array are continuously made (using a uniform random number generator to generate the elements indices). If your data cache size is 64 KB, what will the average memory access time be?**

Average Memory access time is calculated as follows:

Hit rate= 64 Kbytes/256 Mbytes =64 Kbytes/256000 Kbytes = 0.00025

Average Memory Access time= 0.00025\*1+105\*(1-0.00025) =

= 0.00025 + 105\*0.99975

= **104.974 cycles**

1. **If you compare the result obtained in part (b) with the main memory access time when the cache is disabled, what can you conclude about the role of the principle of locality in justifying the use of cache memory?**

Average Memory Access time with Cache disabled=100\*100/100=100 clock cycles

While the average memory access with cache is 104 cycles, larger than when cache is disabled.

So cache memory will not only be useless and it will also be a burden.

1. **You observed that a cache hit produces a gain of 99 cycles (1 cycle vs. 100), but it produces a loss of 5 cycles in the case of a miss (105 cycles vs. 100). In the general case, we can express these two quantities as G (gain) and L (loss). Using these two quantities (G and L), identify the highest miss rate after which the cache use would be disadvantageous.**

Memory access time without cache is T­no;

Memory access time with cache is Tyes;

Miss rate is m;

Average access time (w/ cache) Tyes = (1-m)(Tno-G) + m(Tno + L)

The cache becomes useless when miss rate is high enough to make **Tno <= Tyes**

Here, Tno <= (1-m)(Tno-G) + m(Tno + L);

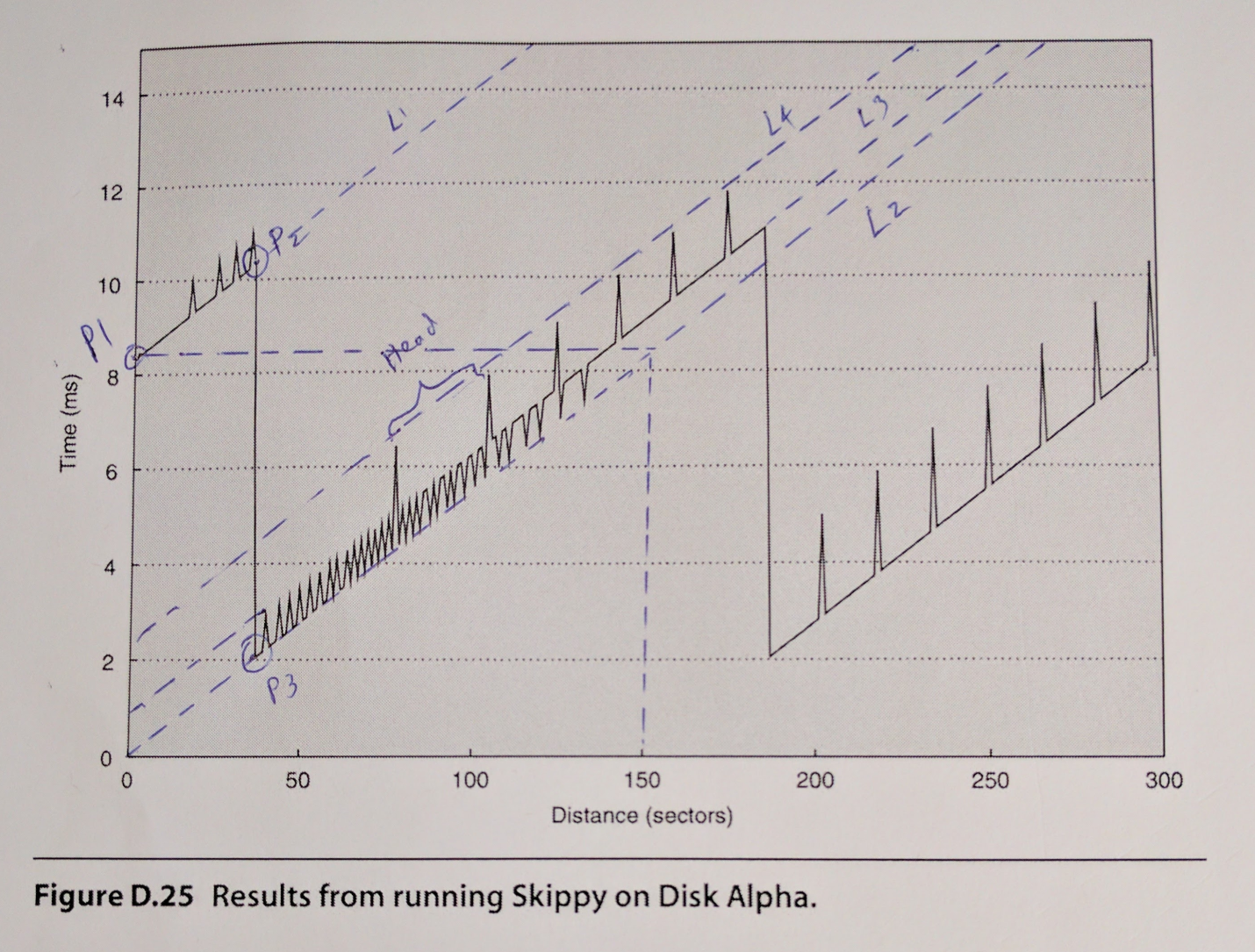
Here, miss rate becomes the following:

**M >= G / (G+L)**

For G=99 and L=5,

**m>= 99/104(~.95)** would make the cache useless and disadvantageous to use.

**D1. The results of running Skippy are shown for a mock disk (Disk Alpha) in Figure D.25.**



1. What is the minimal transfer time?

2.0 ms;

1. What is the rotational latency?

8.3 ms;

1. What is the head switch time?

0.7 ms;

1. What is the cylinder switch time?

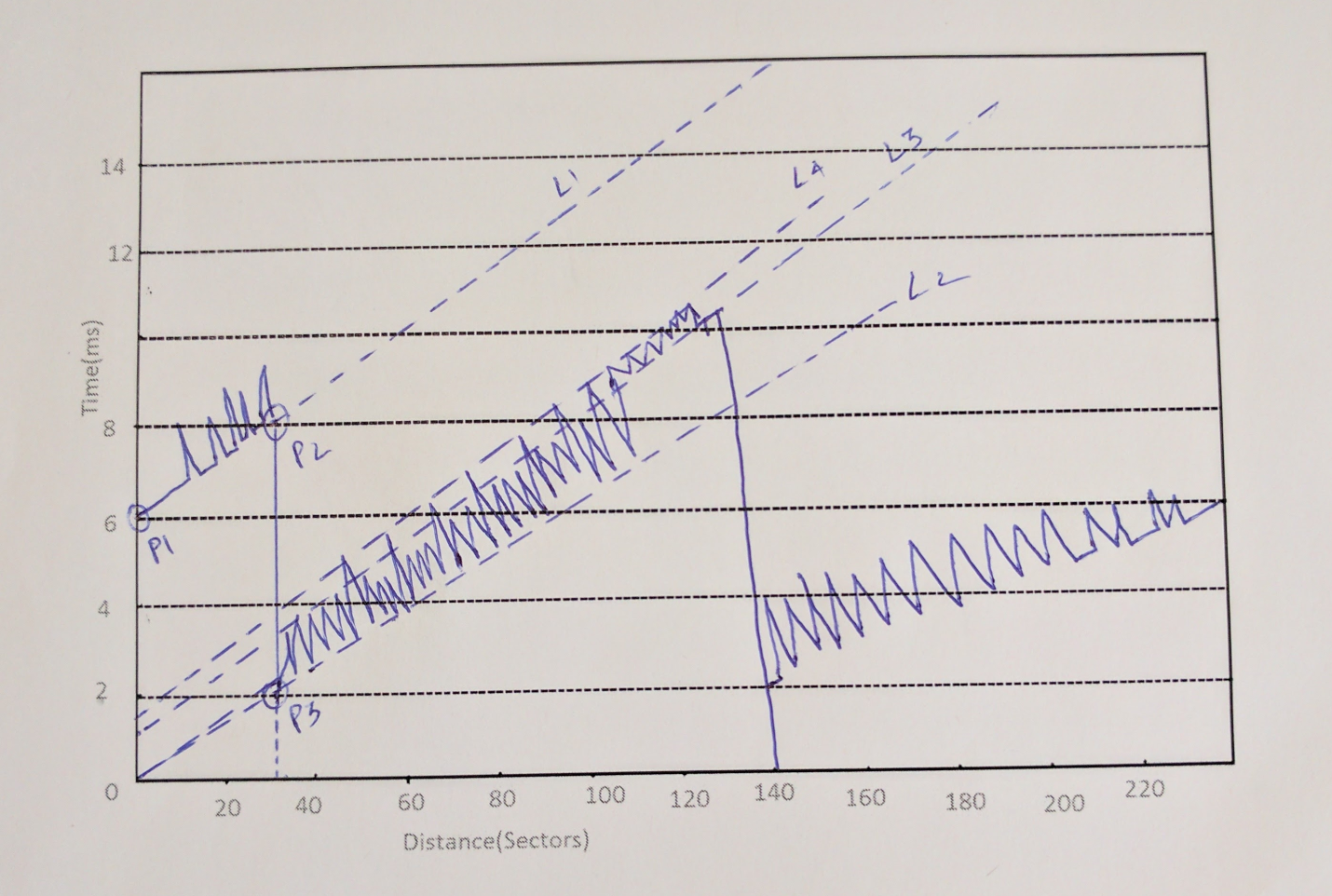
2 ms;

1. What is the number of disk heads?

15 disk heads;

**D2. Draw an approximation of the graph that would result from running Skippy on Disk Beta, a disk with the following parameters:**

1. Minimal transfer time, 2.0 ms
2. Rotational latency, 6.0 ms
3. Head switch time, 1.0 ms
4. Cylinder switch time, 1.5 ms
5. Number of disk heads, 4
6. Sectors per track, 100



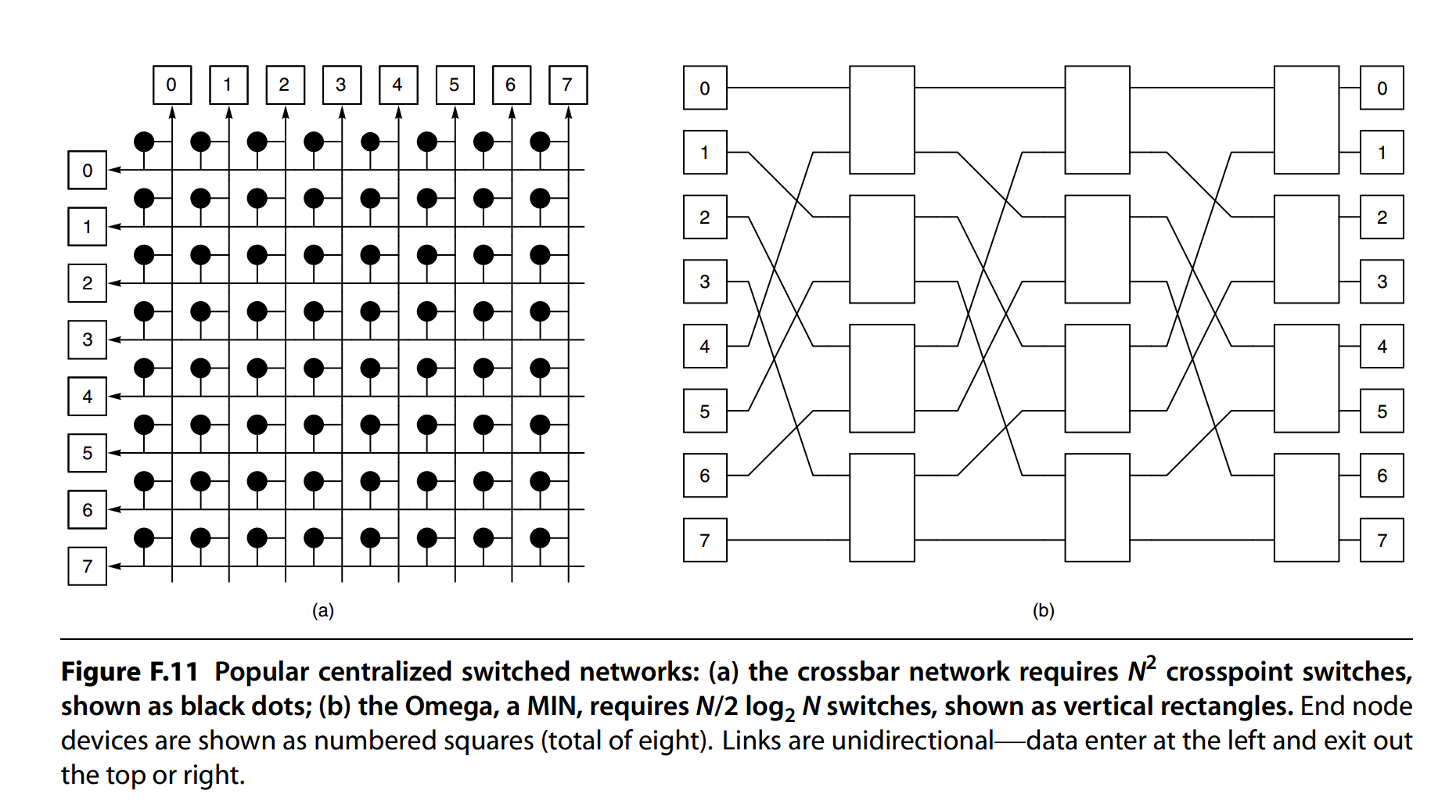
**D.19. You will begin by designing an I/O subsystem that is optimized only for capacity and performance (and not reliability), specifically IOPS. Discuss the RAID level and block size that will deliver the best performance.**

**Answer:**

Since reliability is not a concern, RAID 0 gives best capacity and performance. No space is wasted for redundancy to recover from failures and every disk can be used to handle a random I/O request.

For best performance we will use a block size of 16 KB because larger block size will write off the positioning costs, while smaller block sizes makes sure only necessary data is transferred.

**F.15. Compare the interconnection latency of a crossbar, Omega network, and fat tree with eight nodes. Use Figure F.11 on page F-31, Figure F.12 on page F-33, and Figure F.14 on page F-37. Assume that the fat tree is built entirely from two-input, two-output switches so that its hardware resources are more comparable to that of the Omega network. Assume that each switch costs a unit time delay. Assume that the fat tree randomly picks a path, so give the best case and worst case for each example. How long will it take to send a message from node 0 to node 6? How long will it take node 1 and node 7 to communicate?**



**Solution:**

1. **Crossbar network Figure F.11-a:**

Every node is connected to every other node in the network.

Time taken for path 1 to 7 is 1 unit time.

**Time taken for path 0 to 6:**

Best case: 1 unit time + constant time.

**Time taken for path 1 to 7:**

Since it is not mentioned if it is unidirectional or bidirectional, we need to consider the 2 cross points that should be changed considering unidirectional communication so time taken for unidirectional communication will be : 1 unit time + constant time.

For bidirectional communication it will be: 2 unit time + constant time.

1. **Omega Network Figure F.11-b:**

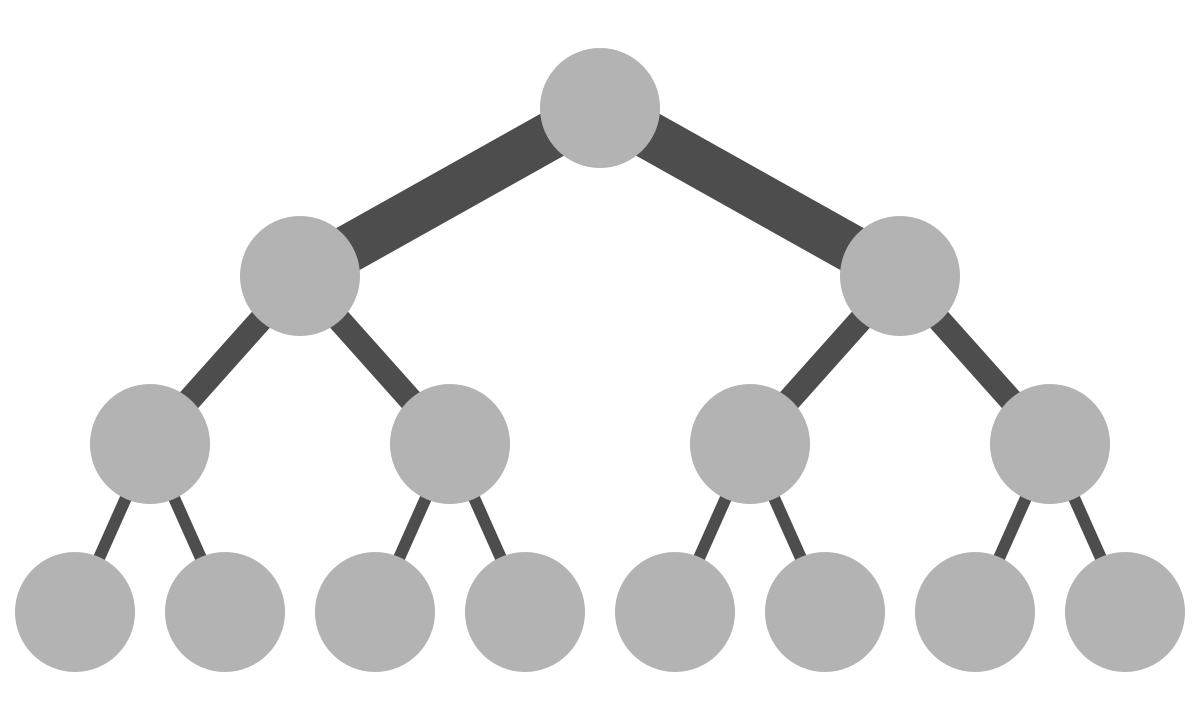
**Time taken for path 0 to 6:**

Best case: 3 unit time + constant time.

**Time taken for path 1 to 7:**

Since it is not mentioned if it is unidirectional or bidirectional, we need to consider the 6 cross points that should be changed considering unidirectional communication so time taken will be 6 unit time + constant time.

1. **Fat Tree Network:**

****

Fat tree network is a network of bidirectional multistage interconnection network with non-blocking properties.

**Time taken for path 0 to 6:**

Best case: Node 0 and 6 will be in the same switch. Time taken = 1 unit time + constant time

Worst case: Node 0 and 6 are in different places. Time taken = 5 unit time + constant time

**Time taken for path 1 to 7:**

Best case: Node 1 to Node 7 had unidirectional communication, Time taken = 2 unit time + constant time

Worst case: Time Taken =10 unit time + constant time