Introduction to Computer Architecture

Assignment 4

Due June 3, 2014

1. [75 = 15 x 5] Performance prediction for RAIDs

Concepts illustrated by this case study

- RAID Levels
- Queuing Theory
- Impact of Workloads
- Impact of Disk Layout

In this case study, you will explore how simple queuing theory can be used to predict the performance of the I/O system. You will investigate how both storage system configuration and the workload in.uence service time, disk utilization, and average response time.

The configuration of the storage system has a large impact on performance. Different RAID levels can be modeled using queuing theory in different ways. For example, a RAID 0 array containing N disks can be modeled as N separate systems of M/M/1 queues, assuming that requests are appropriately distributed across the N disks. The behavior of a RAID 1 array depends upon the workload: a read operation can be sent to either mirror, whereas a write operation must be sent to both disks. Therefore, for a read-only workload, a two-disk RAID 1 array can be modeled as an M/M/2 queue, whereas for a write-only workload, it can be modeled as an M/M/1 queue. The behavior of a RAID 4 array containing N disks also depends upon the workload: a read will be sent to a particular data disk, whereas writes must all update the parity disk, which becomes the bottleneck of the system. Therefore, for a read-only workload, RAID 4 can be modeled as $N - 1$ separate systems, whereas for a write-only workload, it can be modeled as one M/M/1 queue.

The layout of blocks within the storage system can have a significant impact on performance. Consider a single disk with a 40 GB capacity. If the workload randomly accesses 40 GB of data, then the layout of those blocks to the disk does not have much of an impact on performance. However, if the workload randomly accesses only half of the disk's capacity (i.e., 20 GB of data on that disk), then layout does matter: to reduce seek time, the 20 GB of data can be compacted within 20 GB of consecutive tracks instead of allocated uniformly distributed over the entire 40 GB capacity.

For this problem, we will use a rather simplistic model to estimate the service time of a disk. In this basic model, the average positioning and transfer time for a small random request is a linear function of the seek distance. For the 40 GB disk in this problem, assume that the service time is 5 ms * space utilization. Thus, if the entire 40 GB disk is used, then the average positioning and transfer time for a random request is 5 ms; if only the first 20 GB of the disk is used, then the average positioning and transfer time is 2.5 ms.

Throughout this case study, you can assume that the processor sends 167 small random disk requests per second and that these requests are exponentially distributed. You can assume that the size of the requests is equal to the block size of 8 KB. Each disk in the system has a capacity of 40 GB. Regardless of the storage system configuration, the workload accesses a total of 40 GB of data; you should allocate the 40 GB of data across the disks in the system in the most efficient manner.

Begin by assuming that the storage system consists of a single 40 GB disk.

a. Given this workload and storage system, what is the average service time?
b. On average, what is the utilization of the disk?
c. On average, how much time does each request spend waiting for the disk?
d. What is the mean number of requests in the queue?
e. Finally, what is the average response time for the disk requests?

2. [25 = 5 + 10 + 5 + 5] Single-chip multicore multiprocessor

Concepts illustrated by this case study

- Snooping Coherence Protocol Transitions
- Coherence Protocol Performance
- Coherence Protocol Optimizations
- Synchronization

The simple, bus-based multiprocessor illustrated in Figure 4.37 represents a commonly implemented symmetric shared-memory architecture. Each processor has a single, private cache with coherence maintained using the snooping coherence protocol of Figure 4.7. Each cache is direct-mapped, with four blocks each holding two words. To simplify the illustration, the cache-address tag contains the full address and each word shows only two hex characters, with the least significant word on the right. The coherence states are denoted M, S, and I for Modified, Shared, and Invalid.

For each part of this exercise, assume the initial cache and memory state as illustrated in Figure 4.37 (*enclosed*). Each part of this exercise specifies a sequence of one or more CPU operations of the form:

P#: <op> <address> [<-- <value>]

where P# designates the CPU (e.g., P0), <op> is the CPU operation (e.g., read or write), <address> denotes the memory address, and <value> indicates the new word to be assigned on a write operation.

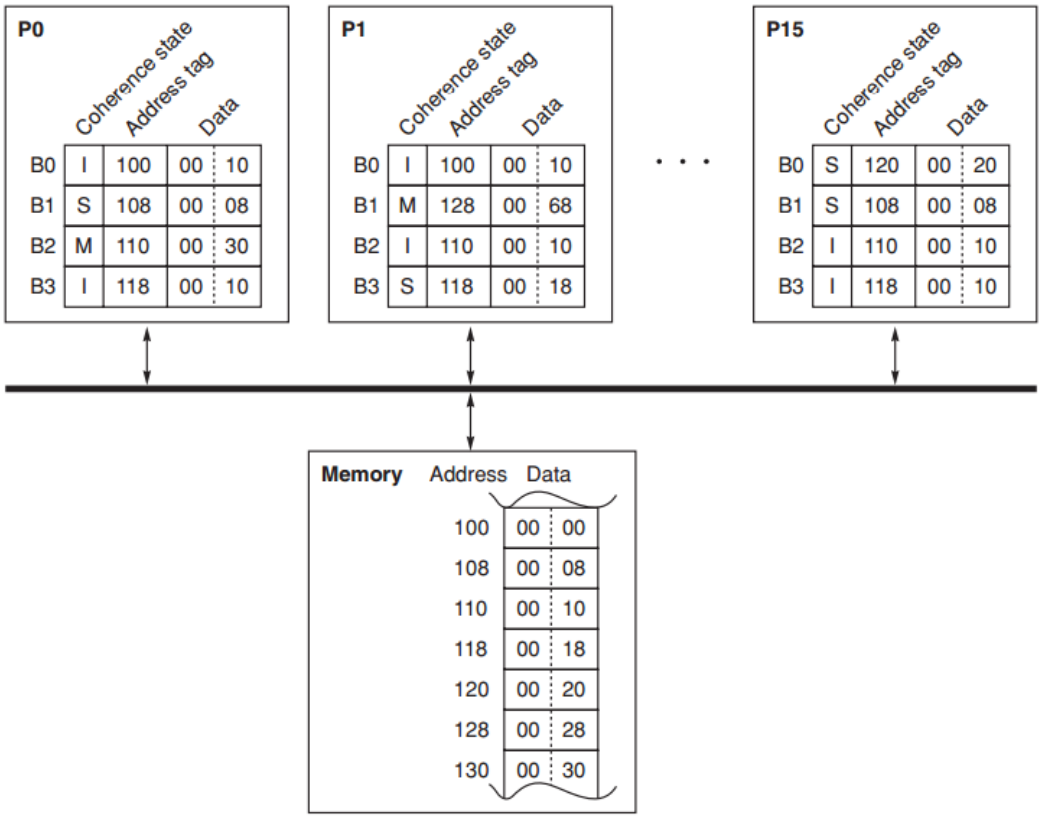

Figure 4.37 Bus-based snooping multiprocessor.

Treat each action below as independently applied to the initial state as given in Figure 4.37. What is the resulting state (i.e., coherence state, tags, and data) of the caches and memory after the given action? Show only the blocks that change, for example, P0.B0: (I, 120, 00 01) indicates that CPU P0's block B0 has the final state of I, tag of 120, and data words 00 and 01. Also, what value is returned by each read operation?

a. P0: read 120
b. P0: write 120 <-- 80
c. P15: write 120 <-- 80
d. P1: read 110