# 6 Solutions

## Solution 6.1

### 6.1.1

| | | |
|---|---|---|
| **a.** | Video Game | Controller—Input, Human<br>Monitor—Output, Human<br>CDROM—Storage, Machine |
| **b.** | Handheld GPS | Keypad—Input, Human<br>Display—Output, Human<br>Satellite Interface—Input, Machine<br>Computer Interface—I/O, Machine |

### 6.1.2

| | | |
|---|---|---|
| **a.** | Video Game | Controller—0.0038 Mbit/sec<br>Monitor—800–8000 Mbit/sec<br>CDROM—88–220 Mbit/sec |
| **b.** | Handheld GPS | Keypad—0.0001 Mbit/sec<br>Display—800 Mbit/sec<br>Satellite Interface—10 Mbit/sec<br>Computer Interface—400–800 Mbit/sec |

### 6.1.3

| | | |
|---|---|---|
| **a.** | Video Game | Controller—Operation Rate<br>Monitor—Data Rate<br>CDROM—Data Rate for most applications |
| **b.** | Handheld GPS | Keypad—Operation Rate<br>Display—Data Rate<br>Satellite Interface—Data Rate<br>Computer Interface—Data Rate |

## Solution 6.2

### 6.2.1

| | |
|---|---|
| **a.** | 43848 |
| **b.** | 87480 |

### 6.2.2

| | |
|---|---|
| **a.** | 0.996168582375479 |
| **b.** | 0.998628257887517 |

**6.2.3** Availability approaches 1.0. With the emergence of inexpensive drives, having a nearly 0 replacement time *for hardware* is quite feasible. However, replacing file systems and other data can take significant time. Although a drive manufacturer will not include this time in their statistics, it is certainly a part of replacing a disk.

**6.2.4** MTTR becomes the dominant factor in determining availability. However, availability would be quite high if MTTF also grew measurably. If MTTF is 1000 times MTTR, it the specific value of MTTR is not significant.

## Solution 6.3

### 6.3.1

| | |
|---|---|
| **a.** | 15.196 ms |
| **b.** | 13.2 ms |

### 6.3.2

| | |
|---|---|
| **a.** | 15.225 |
| **b.** | 13.233 |

**6.3.3** The dominant factor for all disks seems to be the average seek time, although RPM would make a significant contribution as well. Interestingly, by doubling the block size, the RW time changes very little. Thus, block size does not seem to be critical.

## Solution 6.4

### 6.4.1

| | | |
|---|---|---|
| **a.** | Yes | A satellite database will process infrequent requests for bulk information. Thus, increasing the sector size will allow more data per read request. |
| **b.** | No | This depends substantially on which aspect of the video game is being discussed. However, response time is critical to gaming. Increasing sector size may reduce response time. |

### 6.4.2

| | | |
|---|---|---|
| **a.** | Yes | Increasing rotational speed will allow more data to be retrieved faster. For bulk data, this should improve performance. |
| **b.** | No | Increasing rotational speed will allow improved performance when retrieving graphical elements from disk. |

### 6.4.3

| | | |
|---|---|---|
| **a.** | No | A database system that is collecting data must have exceptionally high availability, or data loss is possible. |
| **b.** | Yes | Increasing disk performance in a non-critical application such as this may have benefits. |

## Solution 6.5

**6.5.1** There is no penalty for either seek time or for the disk rotating into position to access memory. In effect, if data transfer time remains constant, performance should increase. What is interesting is that disk data transfer rates have always outpaced improvements with disk alternatives. Flash is the first technology with potential to catch hard disk.

### 6.5.2

| | | |
|---|---|---|
| **a.** | No | Databases are huge and Flash is expensive. The performance gain is not worth the expense. |
| **b.** | Yes | Anything that improves performance is of benefit to gaming. |

### 6.5.3

| | | |
|---|---|---|
| **a.** | Maybe | Decreasing download time is highly beneficial to database downloads. However, the data rate for some satellites may be so low that no gain would result. |
| **b.** | Yes | |

## Solution 6.6

**6.6.1** Note that some of the specified Flash memories are controller limited. This is to convince you to think about the system rather than simply the Flash memory.

| | |
|---|---|
| **a.** | 28.7 ms |
| **b.** | 32.5 ms |

**6.6.2** Note that some of the specified Flash memories are controller limited. This is to convince you to think about the system rather than simply the Flash memory.

| | |
|---|---|
| **a.** | 14.35 ms |
| **b.** | 16.25 ms |

**6.6.3** On initial thought, this may seem unexpected. However, as the Flash memory array grows, delays in propagation through the decode logic and delays propagating decoded addresses to the Flash array account for longer access times.

## Solution 6.7

### 6.7.1

| | |
|---|---|
| **a.** | Asynchronous. Mouse inputs are relatively infrequent in comparison to other inputs. The mouse device is electrically distant from the CPU. |
| **b.** | Synchronous. The memory controller is electrically close to the CPU and throughput to memory must be high. |

**6.7.2** For all devices in the table, problems with long, synchronous buses are the same. Specifically, long synchronous buses typically use parallel cables that are subject to noise and clock skew. The longer a parallel bus is, the more susceptible it is to environmental noise. Balanced cables can prevent some of these issues, but not without significant expense. Clock skew is also a problem with the clock at the end of a long bus being delayed due to transmission distance or distorted due to noise and transmission issues. If a bus is electrically long, then an asynchronous bus is usually best.

**6.7.3** The only real drawback to an asynchronous bus is the time required to transmit bulk data. Usually, asynchronous buses are serial. Thus, for large data sets, transmission can be quite high. If a device is time sensitive, then an asynchronous bus may not be the right choice. There are certainly exceptions to this rule-of-thumb such as FireWire, an asynchronous bus that has excellent timing properties.

## Solution 6.8

### 6.8.1

| | |
|---|---|
| **a.** | USB or FireWire due to hot swap capabilities and access to the drive. |
| **b.** | USB due to distance from the CPU and low bandwidth requirements. FireWire would not be as appropriate due to its daisy chaining implementation. |

### 6.8.2

| Bus Type | Protocol |
|---|---|
| PCI | Uses a single, parallel data bus with control lines for each device. Individual devices do not have controllers, but send requests and receive commands from the bus controller through their control lines. Although the data bus is shared among all devices, control lines belong to a single device on the bus. |
| USB | Similar to the PCI bus except that data and control information is communicated serially from the bus controller. |
| FireWire | Uses a daisy chain approach. A controller exists in each device that generates requests for the device and processes requests from devices after it on the bus. Devices relay requests from other devices along the daisy chain until they reach the main bus controller. |
| SATA | As the name implies, Serial ATA uses a serial, point-to-point connection between a controller and device. Although both SATA and USB are serial connections, point-to-point implies that unlike USB, data lines are not shared by multiple connections. Like USB and FireWire, SATA devices are hot swappable. |

### 6.8.3

| Bus Type | Drawbacks |
|---|---|
| PCI | The parallel bus use to transmit data limits the length of the bus. Having a fixed number of control lines limits the number of devices on the bus. The tradeoff is speed. PCI buses are not useful for peripherals that are physically distant from the computer. |
| USB | Serial communication implies longer communication distances, but the serial nature of the communication limits communication speed. USB buses are useful for peripherals with relatively low data rates that must be physically distant from the computer. |
| FireWire | Daisy chaining allows adding theoretically unlimited numbers of devices. However, when one device in the daisy chain dies, all devices further along the chain cannot communicate with the controller. The multiplexed nature of communication on FireWire makes it faster than USB. |
| SATA | The high-speed nature of SATA connections limits the length of the connection between the controller and devices. The distance is longer than PCI, but shorter than FireWire or USB. Because SATA connections are point-to-point, SATA is not as extensible as either USB or FireWire. |

## Solution 6.9

**6.9.1** A polled device is checked by devices that communicate with it. When the devices requires attention or is available, the polling process communicates with it.

| | |
|---|---|
| **a.** | No. Signals from the controller must be handed immediately for satisfactory interaction. |
| **b.** | Yes |

**6.9.2** Interrupt-driven communication involves devices raising interrupts when they require attention and the CPU processing those interrupts as appropriate. While polling requires a process to periodically examine the state of a device, interrupts are raised by the device and occur when the device is ready to communicate. When the CPU is ready to communicate with the device, the handler associated with the interrupt runs and then returns control to the main process.

| | |
|---|---|
| **a.** | Inputs from the controller generate interrupts handled by the controller driver. |
| **b.** | Polling is okay |

**6.9.3** Basically, each interface is designed in a similar way with memory locations identified for inputs and outputs associated with devices.

| | |
|---|---|
| **a.** | The video game controller is an input only device. It has 4 buttons, a joystick, and a rocker. Each button can be in either an on or off position. The joystick generates nine 1 byte values that indicate relative position as a vector from the origin at the center of the control. Finally, the rocker state is expressed as 4 bits, one bit for each of the four directions. |
| **b.** | A computer monitor is an output only device that requires memory based on the number of pixels available for output. The monitor I am sitting at now is 2560x1600 pixels. Each pixel requires a memory word to set its color. The amount of memory required suggests why video cards tend to have their own, onboard memory. |

### 6.9.4

| | |
|---|---|
| **a.** | The video game controller is an input only device. Thus, a collection of commands should be defined that either poll inputs or are called to process interrupts. The commands simply convey the same information as memory mapping, but return values for command invocations. |
| **b.** | A computer monitor is an output only device A single command can be implemented that sends an image to be displayed to the interface card. Alternatively, it could send a pointer to the image rather than the image itself. |

**6.9.5** Absolutely. A graphics card is an excellent example. A memory map can be used to store information that is to be displayed. Then, a command used to actually display the information. Similar techniques would work for other devices from the table.

## Solution 6.10

**6.10.1** Low-priority interrupts are disabled to prevent them from interrupting the handing of the current interrupt which is higher priority. The status register is saved to assure that any lower priority interrupts that have been detected are handled with the status register is restored following handing of the current interrupt.

**6.10.2** Lower numbers have higher interrupt priorities

| | | | |
|---|---|---|---|
| **a.** | Power Down: 2 | Overheat: 1 | Ethernet Controller Data: 3 |
| **b.** | Overheat: 1 | Reboot: 2 | Mouse Controller: 3 |

**6.10.3**

| | |
|---|---|
| Power Down Interrupt | Jump to an emergency power down sequence and begin execution |
| Ethernet Controller Data Interrupt | Save the current program state. Jump to the Ethernet controller code and handle data input. Restore the program state and continue execution |
| Overheat Interrupt | Jump to an emergency power down sequence and begin execution |
| Mouse Controller Interrupt | Save the current program state. Jump to the mouse controller code and handle input. Restore the program state and continue execution |
| Reboot Interrupt | Jump to address 0 and reinitialize the system |

**6.10.4** If the enable bit of the cause register is not set then interrupts are all disabled and no interrupts will be handled. Zeroing all bits in the mask would have the same affect.

**6.10.5** Hardware support for saving and restoring program state prior to interrupt-handling would help substantially. Specifically, when an interrupt is handled that does not terminate execution, the running program must return to the point where the interrupt occurred. Handling this in the operating system is certainly feasible, but the only solution requires storing information on a stack or some other dedicated memory area. In some case, registers are dedicated to this task. Providing hardware support removes the burden from the operating system and program state need not be pulled from the CPU and put in memory.

This is essentially the same has handling a function call, except that some interrupts do not allow the interrupted program to resume execution. Like an interrupt, a function must store program state information before jumping to its code. There are sophisticated activation record management protocols and frequently supporting hardware for many CPUs.

**6.10.6** Priority interrupts can still be implemented by the interrupt handler in roughly the same manner. Higher priority interrupts are handled first and lower priority interrupts are disabled when a higher priority interrupt is being handled. Even though each interrupt causes a jump to its own vector, the interrupt system implementation must still handle interrupt signals.

Both approaches have roughly the same capabilities.

## Solution 6.11

**6.11.1** Yes. The CPU initiates the data transfer, but once the data transfer starts, the device and memory communicate directly with no intervention from the CPU.

### 6.11.2

| | |
|---|---|
| **a.** | Yes. If the CPU is processing graphical data that is to be displayed, allowing the graphics card to access that data without going through the CPU can prevent substantial delays. |
| **b.** | Yes. If the CPU is processing sound data that is to be output by the sound card in real time, allowing the sound card to access data without going through the CPU can have extensive benefit. |

DMA is useful when individual transactions with the CPU may involve large amounts of data. A frame handled by a graphics card may be huge, but is treated as one display action. Conversely, input from a mouse is tiny.

### 6.11.3

| | |
|---|---|
| **a.** | No. The graphics card does not write back to system memory. |
| **b.** | No. The sound card does not write back to system memory. |

Basically, any device that writes to memory directly can cause the data in memory to differ from what is stored in cache.

**6.11.4** Virtual memory swaps memory pages in and out of physical memory based on locations being addressed. If a page is not in memory when an address associated with it is accessed, the page must be loaded, potentially displacing another page. Virtual memory works because of the principle of locality. Specifically, when memory is accessed, the likelihood of the next access being nearby is high. Thus, pulling a page from disk to memory due to a memory access not only retrieves the memory be accessed, but likely the next memory element being access.

Any of the devices listed in the table could cause potential problems if it causes virtual memory to thrash, continuously swapping in and out pages from physical memory. This would happen if the locality principle is violated by the device. Careful design and sufficient physical memory will almost always solve this problem.

## Solution 6.12

### 6.12.1

| | |
|---|---|
| **a.** | Yes. |
| **b.** | Yes. |

### 6.12.2

| | |
|---|---|
| **a.** | No. Web data is usually small, but requires significant numbers of transactions. |
| **b.** | Yes. Sound data is large with relatively infrequent transactions. |

**6.12.3** See the previous problem for explanations.

| | |
|---|---|
| **a.** | Yes. |
| **b.** | No. |

**6.12.4** Polling would be more inappropriate for applications were numbers of transactions handled is a good performance metric. When data throughput dominates numbers of transactions, then polling could potentially be a reasonable approach.

The selection of command-driven or memory-mapped I/O is more difficult. In most situations, a mixture of the two approaches is the most pragmatic approach. Specifically, use commands to handle interactions and memory to exchange data. For transaction dominated I/O, command-driven I/O will likely be sufficient.

## Solution 6.13

### 6.13.1

| | |
|---|---|
| **a.** | Large numbers of small, concurrent transactions |
| **b.** | Large, concurrent data reads and writes |

**6.13.2** Standard benchmarks help when trying to compare and contrast different systems. Ranking systems with benchmarks is generally not useful. However, understanding tradeoffs certainly is.

**6.13.3** It does not make much sense to evaluate an I/O system outside the system where it will be used. Although benchmarks help simulate the environment of a system, nothing replaces live data in a live system.

CPUs are particularly difficult to evaluate outside of the system where they are used. Again, benchmarks can help with this, but frequently Amdahl's Law makes spending resources on improving CPU speed have diminishing returns.

## Solution 6.14

**6.14.1** Striping forces I/O to occur on multiple disks concurrently rather than on a single disk.

| | |
|---|---|
| **a.** | No. The bottleneck in such systems is network throughput, not disk I/O |
| **b.** | Yes. Sound editing requires access to large amounts of data in real time. |

**6.14.2** The MTBF is calculated as MTTF + MTTR, with MTTF as the dominating factor. For the RAID 1 system with redundancy to fail, both disks must fail. The probability of both disks failing is the product of a single disk failing. The result is a substantially increased MTBF.

In all applications, decreasing the likelihood of data loss is good. However, online database and video services are particularly sensitive to resource availability. When such systems are offline, revenue loss is immediate and customers lose confidence in the service.

**6.14.3** RAID 1 maintains two complete copies of a dataset while RAID 3 maintains error correction data only. The tradeoff is storage cost. RAID 1 requires 2 times the actual storage capacity while RAID 3 requires substantially less. This must be viewed both in terms of the cost of disks, but also power and other resources required to keep the disk array running.

In the previous applications, large online services like database and video services would definitely benefit from RAID 3. Video and sound editing may also benefit from RAID 3, but these applications are not as sensitive to availability issues as online services.

## Solution 6.15

### 6.15.1

| | |
|---|---|
| **a.** | 513C |
| **b.** | 8404 |

### 6.15.2

| | |
|---|---|
| **a.** | BB83 |
| **b.** | FC4C |

**6.15.3** RAID 4 is more efficient because it requires fewer reads to generate the next parity word value. Specifically, RAID 3 accesses every disk for every data write no matter which disk is being written to. For smaller writes where data is located on a single disk, RAID 4 will be more efficient.

RAID 3 has no inherent advantages to RAID 4.

**6.15.4** RAID 5 distributes parity blocks throughout the disk array rather than on a single disk. This eliminates the parity disk as a bottleneck during disk access. For applications with high numbers of concurrent reads and writes, RAID 5 will be more efficient. For lower volume, RAID 5 will not significantly outperform RAID 4.

**6.15.5** As the number of disks grows by 1, the number of accesses required to calculate a parity word in RAID 3 also grows by 1. In contrast, RAID 4 and 5 continue to access only existing values of data being stored. Thus, as the number of disks grows, RAID 3 performance will continue to degrade while RAID 4 and 5 will remain constant.

There is no performance advantage for RAID 4 or 5 over RAID three for small numbers of disks. For 2 disks, there is no difference.

## Solution 6.16

### 6.16.1

| | |
|---|---|
| **a.** | 13333 |
| **b.** | 26667 |

### 6.16.2

| | 16 Disks | | 8 Disks | | 4 Disks | | 2 Disks | |
|---|---|---|---|---|---|---|---|---|
| | IOPS | Bottleneck? | IOPS | Bottleneck? | IOPS | Bottleneck? | IOPS | Bottleneck? |
| **a.** | 14000 | No | 7000 | Yes | 3500 | Yes | 1750 | Yes |
| **b.** | 28000 | No | 14000 | No | 7000 | Yes | 3500 | Yes |

### 6.16.3

| | PCI Bus | | DIMM | | Front Side Bus | |
|---|---|---|---|---|---|---|
| | IOPS | Bottleneck? | IOPS | Bottleneck? | IOPS | Bottleneck? |
| **a.** | 15625 | No | 41687.5 | NO | 82812.5 | No |
| **b.** | 31250 | No | 83375 | No | 165625 | No |

**6.16.4** The assumptions made in approximating I/O performance are extensive. From the approximation of I/O commands generated by the executing system through sequential and random I/O events handled by disks, the approximations are extensive. By benchmarking in a full system, or executing actual application an engineer can see actual numbers that are far more accurate than approximate calculations.

## Solution 6.17

**6.17.1** Runtime characteristics vary substantially from application to application. All three applications perform some kind of transaction processing, but

those transactions may be different in nature. A Web server processes numerous transactions typically involving small amounts of data. Thus, transaction throughput is critical. A database server is similar, but the data transferred may be much larger. A bioinformatics data server will deal with huge data sets where transactions processed is not nearly as critical as data throughput.

When identifying the runtime characteristics of the application, you are implicitly identifying characteristics for evaluation. For a web server, transactions per second is a critical metric. For the bioinformatics data server, data throughput is critical. For a database server, you will want to balance both criteria.

**6.17.2** It is relatively easy to use online resources to identify potential servers. You may also find advertisements in periodicals from your professional societies or trade journals. You should be able to identify one or more candidates using the criteria identified in 6.17.1. If your reasons for selecting the server don't follow from the criteria in 6.17.1, something is not right.

**6.17.3** In problem 6.16, we used characteristics of a Sun Fire x4150 to attempt to predict its performance. You can use the same data and characteristics here. Remember that the Sun Fire x4150 has multiple configurations. You should consider this when you perform your evaluation.

Find similar measurements for the server that you have selected. Most of this data should be available online. If not, contact the company providing the server and see if such data is available.

It's a reasonably simple task to use a spreadsheet to evaluate numerous configurations and systems simultaneously. If you design your spreadsheet carefully, you can simply enter a table of data and make comparisons quickly. This is exactly what you will do in industry when evaluating systems.

**6.17.4** Although analytic analysis is useful when comparing systems, nothing beats hands-on evaluation. There are a number of test suites available that will serve your needs here. Virtually all of them will be available online. Look for benchmarks that generate transactions for the web server, generate large data transfers for the bioinformatics server, and a combination of the two for the database server.

## Solution 6.18

### 6.18.1

| | |
|---|---|
| **a.** | 8.76 |
| **b.** | 7.008 |

### 6.18.2

| | 7 years | 10 years |
|---|---|---|
| a. | 31.536 | 227.76 |
| b. | 21.024 | 151.84 |

**6.18.3** Average failure rates of the drives with longer longevity for 7 and 10 years are:

| | 7 years | 10 years |
|---|---|---|
| a. | 12.264 | 36.792 |
| b. | 8.176 | 24.528 |

It is not surprising that with failure rates starting to double 3 years later, we have to replace far fewer disks in the second situation than the first. The ratio of the number of drives replaced in the first scenario to the number replaced in the second should give us the multiple that we want:

| | 7 years | 10 years |
|---|---|---|
| a. | 2.57 | 6.19 |
| b. | 2.57 | 6.19 |

## Solution 6.19

**6.19.1** In all cases, no. The objective of the customer is not known. Thus, improving any performance metric by nearly doubling the cost may or may not have an price impact on the company.

**6.19.2** As a search engine provider paid by ad hits, throughput is critical. Most HTTP traffic is small, so the network is not as great a bottleneck as it would be for large data transfers. RAID 0 may be an effective solution. However, RAID 1 will almost certainly not be an effective solution. Increased availability makes our product more attractive, but a 1.6 cost multiple is most likely too high.

RAID 0 is going to increase throughput by 70%, meaning the potential exists to serve 1.7 times as many ads. The cost of this gain is 0.6 of the original price. 1.7 times as many ads for 1.6 times the original cost may justify the upgrade cost.

**6.19.3** This problem is not as simple as it would seem at first glance. As an online backup provider, availability is critical. Thus, using RAID 1 where failure

rate decreases for a 1.6 times cost increase might be worthwhile. However, online backup is more appealing when services are provided quickly making RAID 0 appealing. Remember Amdahl's law. Will increasing throughput in the disk array for long data reads and writes result in performance improvements for the system? The network will be our throughput bottleneck, not disk access. RAID 0 will not help much.

RAID 1 has more potential for increased revenue by making the disk array available more. For our original configuration, we are losing between 12 and 19 disks per 1000 to 1500 every 7 years. If the system lifetime is 7 years, the RAID 1 upgrade will almost certainly not pay for itself even though it addresses the most critical property of our system. Over 10 years, we lose between 30 and 50 drives. If repair times are small, then even over a 10 year span the RAID 1 solution will not be cost effective.

## Solution 6.20

**6.20.1** The approach to solving this problem is relatively simple once parameters of a bioinformatics simulation are understood. Simulations tend to run days or months. Thus, losing simulation data or having a system failure during simulation are catastrophic events. Availability is therefore a critical evaluation parameter. Additionally, the disk array will be accessed by 1000 parallel processors. Throughput will be a major concern.

The primary role of the power constraint in this problem is to prevent simply maximizing all parameters in the disk array. Adding additional disks and controllers without justification will increase power consumption unnecessarily.

**6.20.2** Remember that your system must provide both backup and archiving. Thus, you will need multiple copies of your data and may be required to move those copies offsite. This makes none of the solutions optimal.

RAID or a second backup array provides high speed backup, but does not provide archival capabilities. Magnetic tape allows archiving, but can be exceptionally slow when comparing to disk backups. Online backup automatically achieves archiving, but can be even slower than disks.

**6.20.3** Your benchmarks must evaluate backup throughput. Most other parameters that govern selection of a system are relatively well understood—portability and cost being the primary issues to be evaluated.