Daniel Loranger

CS-230 Week 6

*Draw It or Lose It is loosely similar to the 1980s television game Win, Lose or Draw, where teams compete to guess what is being drawn. Rather than a player drawing images on an easel to help team members guess the puzzle (a phrase, title, or thing), the application will render images from a large library of stock drawings as clues. A game consists of four rounds of play lasting one minute each. Drawings are rendered at a steady rate and are fully complete at the 30-second mark. If the team does not guess the puzzle before time expires, the remaining teams have an opportunity to offer one guess each to solve the puzzle with a 15-second time limit.*

*Draw It or Lose It is a multi-user game application that must have the ability to run multiple instances of the game at once, each having players and teams. As you continue to develop this application, you must consider****how storage and memory are managed in each platform****and how management differs.*

*Prompt*

*For the purposes of this assignment, assume that The Gaming Room will have 200 high-definition image files to choose from, each one approximately 8 megabytes in size.*

*Memory Management: Memory refers to what resources are required for transferring files and how the speed and performance of the system are impacted. In the context of the software, Draw It or Lose It, consider how the game application will need to render and display pictures at a fixed rapid rate to meet your client’s requirements. Consider the user’s experience and what is required to have the application run rapidly and effectively on all operating platforms. Address the following in your short paper:*

***What considerations and specific approaches would it take to ensure that memory is effectively managed in the software application, Draw It or Lose It?***

client device Considerations: One of the key performance parameters related to the software application is a steady and predictable frame rate when rendering the image as it plays a key role in the game play. With the varying content needed in this frame of the application being a variable image, it would make sense for the client side application to preload the entire image to be rendered into fast access memory before it is required to start rendering. Once the image is loaded into fast access memory, it would be best practice to ensure it cannot be paged back to slow access memory. With modern smart devices having multiple gigabits of memory, freezing 10’s of MB is not expected to pose any performance issues. Once the image has finished rendering on the screen, the locked memory can be released back to normal operation.

All other predicted memory items needed in the client device application are expected to perform well using traditional memory structures including multi-layer caching, paging swaps and slow access memory such as a rotational hard disk.

Server device considerations: On the server side, the memory will be needed to be versatile to handle all API interactions, where overall random data access performance should be maximized. Depending on the decision to serve images real time, or store them locally on the client will also play a role here as all images might be best stored in memory for constant access to the client devices. A strategy to reduce the static memory contents could be a subset of the images are selectively used for all active games, such as 10 random images are served to requests in 15 second increments. This would be transparent to the end users while allowing reduced content in the fast access memory of the server.

An additional consideration here would be high access of the same content during heavy game play loads so redundant copies of the images might make sense where each core of the processor accesses a different physical ram device with a dedicated copy of the image to help reduce bottle necking at the ram controller chip.

*Storage Management: Storage refers to how files and permanent discs are stored. In this particular instance, a large library of images files is required for Draw It or Lose It. In the context of the game application, consider all aspects of the game that will need to be stored and address the following in your short paper:*

***What considerations and specific approaches would you take to determine how much storage is needed and how to manage storage for your client’s application, Draw It or Lose It?***

The storage needed is directly dependent on the architecture implemented for handing the images. It can be assumed outside of the actual images, the overall footprint of the application on the client side will largely remain consistent (within a few percent of the baseline compiled size). This baseline compiled code size will then be offset based on the image storage. If the images are intended to be stored locally on the client device, then the storage would ideally include space allocations for ALL images related to the game play, but could also be reduced if required to enough images to play a handful of games, with images already played being marked for replacement as a background task between the client and server.

The storage on the server device will need to keep copies of the images, along with server side code installations which should represent a fairly static storage load. The server however will be responsible for storing all of the game play information, player details, history logs, etc. that will be continuously growing and need accessed quickly. For the server, it would be recommended to optimize the storage to be utilized for database type access to minimize the amount of thrashing required to handle the random data access that can be predicted. Some examples would be user login checking where a user’s name needs to be checked against their password, this could represent 2 databases with several million entries in each database. Seeking from a text file would not be practical nor efficient.

Additionally on the server side, is the criticality of the data being stored. A drive crash could ruin the company’s reputation if it occurred during a key event such as an e-sports tournament. To prevent this, the storage implemented must include redundant fault tolerant disk systems that can detect a failed drive and retrieve the information from a back-up source. Ideally this would be done as local redundant as well as remote redundant (example, recoverable from a disaster that wipes out the entire local storage system).

*Comparison: Now that you have identified the considerations and approaches, differentiate between memory and storage management. Address the following in your short paper:*

***What are the differences in how memory and storage are used in terms of the game application functionality?***

Memory (typically denoted as “RAM”) should be optimized and sized accordingly for real-time performance to ensure consistently smooth operation with handing real-time priority requests. Typically, there is no drawback to having more than you need and it is an inexpensive resource when implementing computer systems, but will be limited by physical devices such as a smart phone or tablet where the company cannot control what hardware is used.

Storage is generally only considered if there is a concern about the amount of file data that needs to be held on the client device such as a phone or tablet where 10GB of storage could pose a real problem. On the client side, it should always be considered a semi-scarce resource and footprint minimized to have the least impact on the users end device.

Storage on the server side however, is generally considered an unlimited resource as modern systems can easily introduce additional disk drives either locally or remotely where a server may have access to multiple TeraBytes or PetaBytes of storage.