**Nova Southeastern University**

**College of Engineering and Computing**

**CISC 665 Distributed Systems**

**Winter 2017**

**(January 9– May 7, 2017)**

**Assignment #2**

Due Date: March 30, 2017

Instructor: Dr. Greg Simco

Distributed Systems

Assignment Two

Initial Project Proposal

Bryce Tucker

Student #: BT438

Table of Contents

[Chapter 1: Introduction 5](#_Toc478636008)

[Problem Statement 5](#_Toc478636009)

[Goal Statement 5](#_Toc478636010)

[Server 5](#_Toc478636011)

[Data store 7](#_Toc478636012)

[Client 7](#_Toc478636013)

[Chapter 2: Review of Literature 8](#_Toc478636014)

[Purpose 8](#_Toc478636015)

[Cache architecture 8](#_Toc478636016)

[Cache Replacement Strategy 9](#_Toc478636017)

[Parallel document retrieval 10](#_Toc478636018)

[Summary 11](#_Toc478636019)

[Chapter 3: Approach 12](#_Toc478636020)

[System Implementation 12](#_Toc478636021)

[Client 15](#_Toc478636022)

[Client – server communication protocol 18](#_Toc478636023)

[Server 18](#_Toc478636024)

[Data Layer 23](#_Toc478636025)

[References: 24](#_Toc478636026)

# Chapter 1: Introduction

## Problem Statement

The problem is to design and develop a multi-tiered document vending application where users will be able to purchase and view documents contained in the system. Users will have a client program to use to view documents and will have access to documents based on cost and license availability. Each viewing of a document has an associated cost and each document has only a certain number of licenses for concurrent viewing available. This means a client may only view document content if they have sufficient funds and a license for the document is available at the time of the request for the document.

## Goal Statement

The goal is to design and develop the multi-tiered application and confirm that it meets the requirements, then to measure the effectiveness of the caching approach used by changing the initialization parameters for the cache.

### Server

A single server is implemented to provide the business functionality for the document vending application. This will include:

* Providing the client with a list of available documents along with the cost for each document and the number of available licenses.
* Validate payment received from client when a document is requested.
* Transfer the document to the client that requests the document if there is a license available.
* Receive notification when a document is closed in order to update the license count.
* Allow administrative accounts to make changes to the file contents of the document vending application.

#### Cache

The file server will contain a cache to reduce the cost of document delivery and will conform to the following design issues:

* Cache size
  + Cannot be infinite
  + Must make sense for the system
  + Must be re-configurable based on an initialization parameter
* Must support element query
  + Search for data in the cache
  + Return data if present
* Must support element creation
  + Adding of data to the cache
  + Place in unused space if available
* Must support element removal
  + Remove data from cache
  + Define a formula to remove elements
* Support cache consistency
  + Reads
  + Writes
* Cache service interactivity
  + Client interaction
  + Server interaction

### Data store

The data storage is a separate system layer and provides a central storage location for the documents contained in the system. This data layer will only interact with the server and will provide the persistent data storage along with data delivery and storage tasks needed to support the functionality of the other system components.

### Client

The client program will provide an interface for the user, transmit requests to the server and receive data from the server. The client will start with a random quantity of units to use to purchase documents from the server. The client will provide the following functions:

* Display a list of document names and costs
* Request a document
* Authorize payment for the document
* Receive confirmation message from the server
  + Payment received and license available
  + Insufficient payment
  + License not available
* Receive and display the document
* ‘Return’ the document when the document closes

# Chapter 2: Review of Literature

## Purpose

The review of literature pertaining to the implementation of a document vending application, in specific the implementation of a distributed file system with a cache was conducted. The purpose of the review was to establish a foundation of the current state of investigation into the type of implementation pertinent to the stated requirements of this assignment. Based on the literature, the advancement of ideas about implementing a more efficient solution to the given requirements could be built.

## Cache architecture

Rodriguez [2001] analyzed web cache architectures looking at hierarchical, distributed, and hybrid hierarchical and distributed architectures. Their conclusions show that the architectural implementation selected will depend on the constraints and goals in a scenario. The scenario in this assignment is for a single server that provides documents to clients, so this paper is not very applicable to this assignment. The ideas described of the benefits of distributed caches that enable clients to check for and obtain documents in parallel does lead towards possible solutions to the requirements of this assignment.

The paper by Nelson [2006] describes a cache implementation that uses separate processes to contain cached data. The paper describes advantages that include:

* Each datum can respond actively, independently and in parallel to requests
* The total size of cached data may exceed the memory of a single compute node
* An idle datum may be easily removed by terminating its process
* Stale data may be replaced in parallel when multiple changes to the data store occur
* Replacement policies may be implemented independently from the cache itself
* A cached datum may dynamically migrate to the CPU that best responds to requests

From this paper, the implementation of the solution for this given assignment includes the use of processes to perform the caching requirement. By following the described implementation, the cache manager will spawn processes to cache documents instead of maintaining the documents itself. When a document request is made, the cache manager will examine a maintained list to determine if the requested document is cached in a process, the process id will then be provided so the client and the given process can interact directly for the client to download the document.  
Cache replacements will be performed by terminating the process and spawning a replacement process to contain a given document. The cache manager will control the number of cache processes based on a configuration value to meet the requirement that the cache not be infinite. The cache manager will control the replacement of cached documents by assigning a time to live (ttl) value to the process. Each time the process is accessed for document download, the ttl will be reset. If the document is not accessed before the ttl expires, or the cache manager does not act to renew the process, then the process will terminate, thus reducing the resources used by the cache.

## Cache Replacement Strategy

Podlipnig [2003] conducted a survey of web cache replacement strategies to establish classifications of those strategies. They propose the following classes:

* Recency-based strategies
* Frequency-based strategies
* Recency/frequency-based strategies
* Function-based strategies
* Randomized strategies

The general overview of these classes of strategies provided guidance on the selection of the most appropriate method for this assignment. The concurrent caching strategy has a default recency-based strategy based on a time to live value assigned to each cache datum. If the datum is not selected recently, in that ttl, then the cache datum is removed. Of course it is possible to implement a different replacement strategy as the cache manager has control over the cache datum processes it spawns.

The function-based strategy idea could account for the other factors of this assignment: document cost and number of licenses. Document cost and number of available licenses could guide the selection of documents to have in cache because higher costs or lower available licenses could influence the likelihood of a document being requested. For instance, if all the licenses for a document are being used, does it make sense to cache that document since it can’t be downloaded until a license is freed up?

In the interest of time and simplicity, the recency-based strategy will be used for cache replacement where the ttl will be an adjustable parameter.

## Parallel document retrieval

Janakiraman [2004] presented a technique for parallel retrieval of structured documents from multiple servers. This solution involves the segmentation of the documents in question to allow the document parts to be retrieved from multiple servers at the same time. Though their proposed solution does not directly relate to this assignment, the idea of breaking documents apart to allow for parallel operations is interesting.

Koo [2003] specifically showed that parallel download methods are not necessarily an improvement overall due to network traffic and the coordination of multiple servers.

For the purpose of this assignment, there is a single server that implements a cache of data stored in a data layer that is assumed to take longer to access. This scenario is different than either of the papers so the coordination is more easily accomplished. The network traffic is a potential issue, but as the initial implementation of this assignment is on a single computer, that is not a major consideration.

The design of the system in this assignment is to assume the possibility of deploying the parts of the system onto multiple servers, so it may be possible to capitalize on the benefits of the principles described in Janakiraman [2004] for parallelizing the download operations by having multiple cache processes contain parts of the document being requested.

## Summary

The review of literature applicable to this assignment was overall not a great match to the specific scenario of this assignment, but the principles in several instances do provide guidance in the development of potential solutions to the requirements. The ideas of breaking up the documents so that parallel operations can be performed by concurrent cache processes and cache replacement will be based on recency of access were brought together from looking at the various literature.

# Chapter 3: Approach

## System Implementation

The following sections describe the system implementation for this experiment. The key components under examination are a concurrent cache methodology as described in [1] and the implementation of a split file storage function where all documents are split into two parts, a initial part that is the same size for all documents and a remainder part that contains the remainder of the file which will be a different size depending on the size of the file.

The concurrent cache system will allow multiple processes to be active and responding to multiple document requests while the split file mechanism allows for a dual cache method where the first cache layer provides a larger number of documents because the size of the cached file parts is consistent and known. By having all the first level cache processes be the same size, the cache manager has a simplified way to manage the amount of resources used.

The cache manager must then more carefully manage the second level cache as those cache processes will use different amounts of resources depending on the size of the remainder files currently being stored. This means the number of cached files will be different though the amount of resources allocated to the first and second level cache processes will be the same. The second level cache will have a lower number of processes which means there will not be a one for one match between the initial parts of documents stored in the first level cache and the remainder of the document stored in the second level cache.

This implementation has the potential to provide a greater number of cache hits at the first level cache but may have a reduced benefit by having a miss on the second level cache for the remainder of the document. The multiple process approach to the cache implementation allows for a greater amount of parallelization of the transmission of documents to the client. The following scenarios of document requests are considered:

* Requested document is not cached: In this case neither part of the document (initial and remainder) is in a cache (first level or second level). This is a cache miss and results in the server needing to obtain the document from the data layer. The designed benefit of the split file and multiple cache process solution in this instance is the server can have the cache manager initiate a first level cache process to request the initial part of the requested document from the data layer and concurrently initiate a second level cache process to request the remainder of the document from the data layer.   
  Even though the data layer may not be able to deliver the files in parallel, there is still likely a performance improvement in the communication between the separate processes and separate threads in the data layer. Additionally, since the size of the initial part of the document is typically much smaller than the remainder of the document, the transfer of the initial part to the first level cache will complete first. That level cache process can then begin the data transfer to the client while the second level cache continues to obtain the remainder of the document.   
  This approach, in the event of a cache miss, can lower the length of time between the client requesting a document and the beginning of the delivery of the document data. Furthermore, if the client is designed to display the initial parts of a document to the user while continuing to obtain the remainder of the document data in the background, the latency for the user can be significantly reduced.
* Requested document has the initial part cached in the first level cache but the remainder of the document is not in the second level cache: In this case, only the initial part of the document is cached. This allows the cache manager to provide the client with the connection to the first level cache process that contains the initial part of the document immediately so the first part of the document can be transferred. Concurrently, the cache manager will initiate a second level cache process to request the remainder of the requested document from the data layer.   
  This methodology provides the ability for the client to obtain some of the document immediately while having to wait for the remainder of the document to be delivered. This can reduce the latency for the user to begin reading the document.
* Requested document has the remainder of the requested document in secondary cache but the initial part of the document is not in the first level cache: This case should not occur, as the cache manager will only populate the secondary cache with content that matches up with documents in the first level cache. When a first level cache process containing the initial part of a document expires (because the process time to live has expired) then the cache manager will be notified and will terminate a matching second level cache process if one exists.
* Requested document is cached in both the first and second level caches: In this case, there are cache processes that contain both the initial part of the document and the remainder of the document. Since there are two processes available it would be possible to have the client use multiple threads to communicate with the two processes to transfer the document parts. This may provide some performance improvement if the client has concurrent communication capabilities. In this case if the communication with the two processes must be performed serially, then there will be an increase in the latency due to the additional overhead of forming a data connection to the second level cache process to transfer the remainder of the document.

### Client

The client provides the user interface and will present options to the user in the form of a menu. Selecting the option executes the function. The functions are described here:

#### Start Connection

When the client application is started, there is an initial connection established to the document server to authenticate the user. The user provides credentials when starting the client application. Those credentials are used to determine the level of access the user has to the functionality of the server. In specific, the credentials determine if the user has access to administration functions: add documents and remove documents. Only users with specific credentials maintained on the server will have administrative access. All other user credentials will just have document download/read capabilities.

Once the initial connection has been confirmed, the client will present the appropriate menu of options to the user.

#### Functions

##### Show list of documents

The client has a menu option to retrieve and display a list of documents contained in the document server. The display of data will include the name of the document, the cost of the document, the total number of licenses for that document, and the number of currently available licenses for the document.

The documents will be numbered so the user will have a simple reference mechanism to reference the desired document.

##### Request a document

The client application will allow the user to request to view a document. The function will require the entry of a number that references the document from the listing.

###### Success

The success condition for the request of a document will be the display of the contents of the document. Success includes the availability of funds, the successful transaction of funds, and the availability of a license for the document to be viewed.

###### Lack of licenses

The client application will display a message that there are not enough licenses available to view the document in the case when the requested document does not have an available license at the initiation of the request for the document. This may occur even though an immediately prior execution of the display of the list of documents indicates available licenses, because other clients may have executed a request for the document in the time between the request for the documents list and the request for the document.

###### Lack of funds

Each client will have a certain quantity of funds available for use in purchasing the viewing of the documents in the document vending application. If the client does not have sufficient funds to purchase a view of the requested document, then the client will display a message to the user that there are insufficient funds available.

Since the cost of the documents can be different depending on if the document is cached or not, the cost must be determined by communicating with the server whenever a document is requested. The requirements of the assignment indicate a 50% reduction in cost if a document is in the cache. Since this implementation splits the document into two parts, then the cost savings will also be split. The cost reduction will be 25% if the first part of the documents is in cache level one and an additional 25% cost reduction will be applied if the second part of the document is in cache level two at the time of the request.

###### Other conditions

Other situations will generate messages to the user. For instance, if a document is removed by a client with administrator access between the time a client requests a list of documents and when the client requests a document, then an error message will need to be sent indicating that the document numbering has changed since the last time the list was requested.

#### Administrator functions

Administrator functions will only be available to users registered to have those capabilities.

##### Add document

The administrator can add a document to the server. This will involve uploading a document and entering the value for the number of licenses available. The cost of the document will be determined based on the size of the document. In alignment with the assignment requirements, the cost will be the size of the document divided by 1K rounded up.

##### Remove document

The administrator can select a document to be removed. This will remove the document from the server and data storage layer.

### Client – server communication protocol

All communication between system components will be done using tcp sockets as implemented by the java.net package. The server will establish a ServerSocket that will listen for client connections.

### Server

The server will listen for client connections and will handle the client requests based on what the request is.

#### User management

When a client first connects and needs to determine the user’s access level, the server will compare the provided credentials to the user list maintained in local memory. The user list is stored in the data layer, but the server retrieves the data on initialization. The server will immediately respond with the confirmation or denial that the user is an administrator. The client will then know which menu or options to present to the user.

#### List of documents

When the client requests a list of documents, the server responds with the information maintained in local memory of the list of documents obtained from the data layer. This list of documents is initially loaded from the data layer and is updated when administrator actions of additions or removals are made.

##### License management

The server also updates the list of documents with the number of licenses currently available. This changes as users request documents and return documents. The server monitors when a client has successfully requested a document and will monitor that client to determine when the client stops viewing the document.

If the client stops viewing the document, an update message will be sent to the server. The server will monitor the connection with the client so in the case where the client connection is severed, the server will assume the document is no longer being read and will increase the number of available licenses for that document.

##### Cost management

The server will also update the information about the available documents by adjusting the document cost depending on the cache management. When a document is fully cached, then the cost will be 50% of the document’s full cost. If the document is cached only in the level one cache, then the cost will be reduced by 25% of the full cost.

#### Document request

When a document is requested by a client, the server will check the cache status to see if the document exists, if there are licenses available, and if the client has funds to cover the current cost of the document.

##### License availability

The client can only read the document if there is a license available at the time of the request. The server will maintain a count of available licenses as the clients request documents. If there are no licenses available at the time of a request, then the server will respond with a denial message to the client’s request for a document.

##### Purchasing

The client can only read the document if the client has sufficient funds to cover the cost of the document being requested. The document cost is determined at the time of the request based on the document cost determined when it is added to the document vending application and whether or not the document is currently stored in the cache. The server will determine if the client has sufficient funds at the time of the request for the document.

##### Document transfer

Assuming the client is authorized to view the document based on the license availability and the purchasing, the server will initiate a transfer of the document to the client.

* If the document is stored in cache one and in cache two, then the server will provide the client with the addresses for the cache one process and the cache two process so the client can commence the document download from both processes simultaneously.
* If the document is stored in cache one but not in cache two, the server will immediately provide the client with the cache one address so the client can begin download of the available portion of the document. The server will then initiate a new cache two process to begin the retrieval of the second part of the requested document from the data layer. The server will provide the client with the address of the cache two process and download to the client will commence once the cache two process has obtained the document from the data layer.  
  The server will terminate a cache two process if necessary (see the cache management section for more description)
* If the document is not stored in either cache, then the server will initiate new cache one and cache two processes to retrieve the requested document parts from the data layer. The server will provide the client with the addresses of the processes and the download of content from the cache processes to the client will commence once the data has been obtained from the data layer.

#### Cache System

The cache system is a central requirement for this assignment. The use of a cache should improve the retrieval times for the requested documents by the clients.

##### Implementation approach

The cache will be implemented using multiple processes executed using commands available in the java.lang library.

###### Processes

When a document is requested, the process will be started with the document information necessary to have the process obtain the document from the data layer. The server will maintain a listing of the active processes, the addresses for the processes and the document the process has cached. The server will also maintain information on the time to live for each active process.

###### Multi-layer

The caching will be multiple layer in that the layer one cache will contain the first part of a file while the layer two cache will contain the remaining portion of the file. The size in bytes of the first part of documents will be standardized so the server can determine the number of processes to have active based on the amount of available storage space allotted to the cache level one.

Cache level two will also be allotted some amount of space, but will be managed differently due to the remaining parts of the documents being different sizes.

###### Document splitting

Documents stored in the vending application will be split into two parts, a beginning part and a remainder. The size of the beginning part will be based on a parameter in the server setup file. The size of the remainder part of each document will depend on the overall size of the document. The goal of document splitting is to allow for greater parallelization of operations on each document.

##### Cache management

##### Requests

Each cache process handles the transfer of data when a request is made. The server will direct the client to the correct process when a request is made and the cache process exists.

##### Additions

The server spawns a new cache process when there is a request for a document that does not have a cache process in place.

##### Removals

The server removes cached documents by terminating a cache process. Alternatively, the cache process is allowed to terminate based on the time to live value expiring and the server not renewing the process.

#### Admin functions

The following are the administrative functions the server supports.

##### Add a document

When an administrative client executes this command, the client will transmit the document to the server along with a value for the number of licenses the document has. The server will transfer the file to the data layer and will update the list stored in the data layer for the list of documents stored in the data layer. The server will then provide the new document information in subsequent document list requests.

##### Remove a document

When an administrative client executes a command to remove a document from the application, the server will:

* Determine if any clients have the document open and send close commands to those clients.
* Remove the document from the cache if it exists in either cache level by adjusting the time to live values to 0 and allowing the processes to terminate.
* Will work with the data layer to remove the document.
* Will update the document metadata list in the data layer and in local memory.
* Will provide confirmation to the administrative client that the document has been removed.

The document will then not be included in subsequent requests for a list of documents.

### Data Layer

The data layer is a separate application that operates to store the documents and provide those documents upon request to the server. The data layer will use a process with multiple threads to handle the communication with the server. The communication will be conducted with TCP/IP Sockets in Java.

Requests from the server will come in to a main process that is listening on a specific port. The request will be handled based on the type of request it is.

#### Request document list

When a request for the list of documents is received, the main process will reply with the stored list of document metadata. The document metadata is updated during add and remove operations described below.

#### Request document

When a document is requested, the main process will create a thread to handle the communication of the document through the established socket connection.

#### Add document

The main process will create a thread to handle the communication of the document being transferred into the data storage. The file will be stored and the document metadata will be updated.

#### Remove document

The main process will execute the removal of the document from storage and will update the document metadata file.

# References:

Koo, S. G. M., Rosenberg, C., Xu, D. (2003). Analysis of Parallel Downloading for Large File Distribution. *Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems*, 128-135. doi: <https://doi.org/10.1109/FTDCS.2003.1204324>

Janakiraman, R., Xu, L. (2004). Efficient and Flexible Parallel Retrieval Using Priority Encoded Transmission. *NOSSDAV ’04 Proceedings of the 14th international workshop on Network and operating systems support for digital audio and video*, 48-53. doi: <https://doi.org/10.1145/1005847.1005859>

Podlipnig, S., Böszörmenyi, L. (2003). A Survey of Web Cache Replacement Strategies. *ACM Computing Surveys Volume 35 Issue 4*, 374-398. doi: <https://doi.org/10.1145/954339.954341>

Rodriguez, P., Spanner, C., Biersack, E. W. (2001). Analysis of Web Caching Architectures: Hierarchical and Distributed Caching. *IEEE/ACM Transactions on Networking (TON) Volume 9 Issue 4*, 404-418. doi: <https://doi.org/10.1109/90.944339>

Nelson, J. (2006). Concurrent Caching. *Proceedings of the 2006 ACM SIGPLAN workshop on Erlang*, 32-38. doi: <https://doi.org/10.1145/1159789.1159797>