



**Faculty of Engineering & Applied Science**

**SOFE4790U – Distributed Systems**

**Homework - HDFS as a Distributed System**

**Due Date: 09/18/2022**

<b>First Name</b>	<b>Last Name</b>	<b>Student ID</b>
Abdul	Bhutta	100785884
Alexander	Campbell	100703650
Mamun	Hossain	100553073
Owen	Musselman	100657709

## **1.0 Objectives**

### **1.1 Describe what distributed systems are**

Distributed systems are an environment where the components are in different areas over a network. Where these machines will split the work and complete the work more efficiently than on centralized systems

### **1.2 Describe characteristics, and requirements of a distributed systems**

To be considered a distributed system, the system must have several key characteristics. The first and most important is that it is not centralized, it is a shared computer resource. It must also be able to run processes concurrently, for example, an application that is distributed across two computers where there are two components being run concurrently and coordinating on different computers. Distributed systems must also be reliable and scalable otherwise they will fall apart under large network loads. They must also be transparent, which can fall under many categories such as location, access, concurrency, etc. Transparency essentially boils down to hiding the complexities of the distributed system away from the user such as where their files are stored.

### **1.3 How this affects a company's hardware and software**

Using a distributed system heavily influences the architecture of a company's hardware and software. With hardware, this could mean there may be in-house distributed systems or running certain tasks via a cloud-based distributed system. Within software it mainly affects the level of interconnectivity between components, how many types of components, the number of actual running components, and how the components will communicate with each other such as during run-time or during setup.

### **1.4 Identify Key building blocks/ Components & Common Characteristics**

The clear building blocks in a distributed system are Availability; the system can perform its function at the specified time, Reliability; the system can deliver its services even if certain parts are compromised, efficiency; there is the least amount of wasted resources, and manageability; a system can be easily diagnosed and repaired.

### **1.5 Identify NFR & Basic Design Issues**

The non-functional requirements of a distributed system are as follows: Scalability, the system should accommodate fluctuating system traffic. Reliability and Availability, the system should be reliable and available at all times. Security, the system should be secure from unauthorized access. Maintainability, system needs to be easily maintained. Usability, the system should be easy to use. Performance, the system should have high performance.

The basic design issue with distributed systems is that security is difficult as the different components need to communicate with each other securely with proper permissions needing to be assigned to assure the system is secure. The reliability of the system needs to be maintained, but that is difficult as the system has multiple points of failure making it difficult to track down what went wrong and where it went wrong. The availability of the system can be difficult as routine backup images of the system need to be made to save the state of the system in case of a failure so that the system can recover and function again in a timely manner.

## **1.6 Examples of Distributed Systems**

- The Internet
- Telecommunication networks.
- GPS technologies.
- eCommerce technologies (PayPal, Amazon, etc)

## 2.0 Homework Questions

### 1. Reflect on your previous course on OS. Can you provide a few paragraphs of a summary of what you studied in the course?

In the operating system course, we learned different aspects of operating systems in terms of functionality and performance. The course was broken down into various components to help us understand operating system functions from hardware to software related. The purpose was to design and implement simple tasks of an operating system while understanding how each concept was applied. The ubuntu/Linux OS and C language was used to demonstrate how processes were created and deleted while learning how the communication between the processes worked. Synchronization with OS concepts was introduced such as mutex locks, semaphores, and deadlocks. CPU scheduling concepts were introduced such as FIFO, LIFO, Round Robin, Shortest Job First, Queues, and Shortest Remaining Time First. Process scheduling would decide which processes to end and which processes to execute next ensuring that the system functions accordingly. Various queues are used: Job ready queue stores all the processes in the system. Ready queue where the processes are ready and waiting to be executed. A device queue is where a process is blocked/halted while waiting for a device. Other schedulers are the short term, long term, and medium term. They are the CPU scheduler, Job scheduler, and swapper scheduler respectively. Context switches were also important as they can store and restore the CPU in the process control block (PCB), without context switching, multitasking would be a lot more difficult on systems. Lastly, file management concepts were introduced such as virtual memory, memory allocation, and page table structure. Furthermore, paging methods were discussed which allow the operating system to get processes that are located on the system's secondary storage (hard drive) making it faster to access data. Paging is used to map the virtual address to the physical address of the memory (Memory management unit). It is noteworthy that the page size is the same as the frame size, this is done to ensure that there is no external fragmentation, where the dynamically allocated memory leaves a section of memory unusable. Internal fragmentation is when a process is allocated to memory, but that process is smaller than the memory that was initially requested by the process. A summary of topics is provided below.


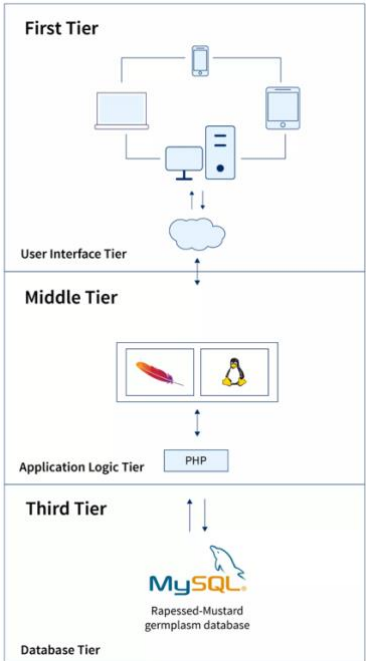
Topic	Concepts
Processes	Creation, Deletion, Inter-process communication
Threads	Creation, Deletion, Shared Memory
Synchronization	MUTEX locks, Semaphore, Deadlocks
CPU Scheduling	FIFO, Round Robin, Shortest Job first, Queue, Shortest Remaining Time First
File Management	Virtual Memory, Allocation, Paging, Page Table Structure

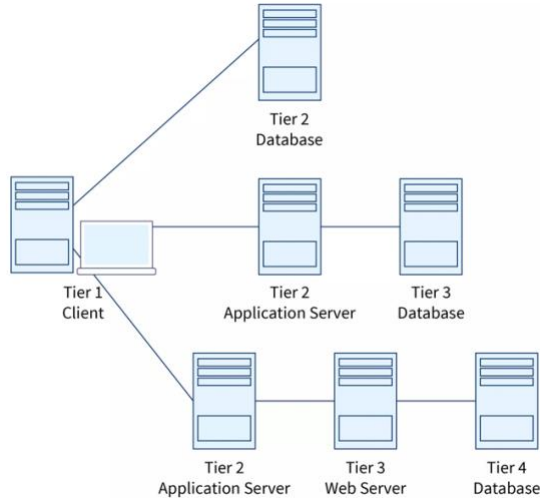
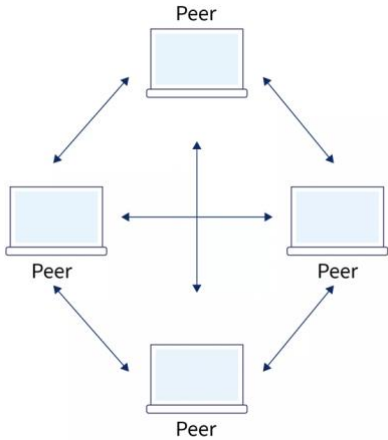
## 2. What are the main functions of a distributed OS?

A distributed operating system performs as a normal operating system but runs on multiple CPUs where the system has its CPU and memory while interacting with each other. The distributed operating system uses multiple CPUs for real-time applications to help process the data. The communication is done by means of WAN or LAN connections which allows the system to share resources like memory, storage, and computation resources. Distributed systems allow the users for resource sharing (This will allow more underpowered machines to utilize these resources when needed), scalability, performance, and fault tolerance.

## 3. How do you implement them?

Most distributed systems rely on network calls therefore a good network is required to construct a distributed system. We need to explore the four types of distributed system architectures to implement the system. The four systems are shown below.

Type	Description	Implementation
Client-Server Architecture	Mainly used for resource sharing where the server handles data and resources requested by the client. An example is Netflix.	 <p>The diagram illustrates a Client-Server Architecture. On the left, a laptop icon represents the 'Client Application'. On the right, a server tower icon represents the 'Server Application'. A double-headed arrow connects the two, indicating bidirectional communication.</p>
Three – Tier Architecture	The three-tier architecture has three layers: presentation, application (business layer), and data. The presentation tier is where the user will access the application to make a request. The application layer is used for business logic. The data tier is the database tier where the application data is stored. An example of a three-tier architecture is a typical business application.	 <p>The diagram illustrates a Three-Tier Architecture, divided into three horizontal sections:</p> <ul style="list-style-type: none"> <li><b>First Tier (User Interface Tier):</b> Contains icons for a laptop, a smartphone, and a tablet, representing user devices.</li> <li><b>Middle Tier (Application Logic Tier):</b> Contains icons for a server and a database, with a 'PHP' label below them.</li> <li><b>Third Tier (Database Tier):</b> Contains the 'MySQL' logo and the text 'Rapessed-Mustard germplasm database'.</li> </ul> <p>Arrows indicate the flow of data and requests between these tiers: from the First Tier to the Middle Tier, and from the Middle Tier to the Third Tier.</p>

<p>Multi – Tier Architecture</p>	<p>Used for when application needs to forward data or requests to multiple networks</p>	 <pre> graph LR     T1[Tier 1 Client] --- T2D[Tier 2 Database]     T1 --- T2AS1[Tier 2 Application Server]     T1 --- T2AS2[Tier 2 Application Server]     T2D --- T3D[Tier 3 Database]     T2AS1 --- T3AS[Tier 3 Application Server]     T2AS2 --- T3WS[Tier 3 Web Server]     T3D --- T4D[Tier 4 Database]     T3AS --- T3WS     T3WS --- T4D </pre>
<p>Peer – to – Peer Architecture</p>	<p>Each system is a node while acting as a client or server. An example of P2P is downloading torrents.</p>	 <pre> graph TD     P1[Peer] &lt;--&gt; P2[Peer]     P1 &lt;--&gt; P3[Peer]     P1 &lt;--&gt; P4[Peer]     P2 &lt;--&gt; P3     P2 &lt;--&gt; P4     P3 &lt;--&gt; P4 </pre>

#### 4. Compare the functionalities provided by a centralized file system to that of a distributed file system?

A centralized file system is where all the data is stored on one server or location while limiting the number of users accessing the data. The data is stored on a single database and accessed through the local area network. The data stored is secured since all the data is stored in one database and retrieving the data will take less time. A distributed file system is where the data is stored on multiple databases in various locations that are interconnected with each other. The distributed file system performs better as there are more systems which reduce the network's load and allows the network to be accessed from more networks. A centralized file system is cheaper than a distributed file system since all the data is in one location and on one system while the distributed system will have many nodes/systems at different locations. Although in a distributed system, if one of the databases fails, not all users will be impacted in a centralized file system, all

users will be impacted. Additionally, if a distributed system is not connected to the network, it will be useless, where this could happen to a centralized system, and it could potentially function as it was if it did not rely on network connections like a user's PC. The scalability of these two systems differs as well, where a distributed system is very scalable and able to add new resources on demand, whereas a centralized system is not as easy.

**5. What are the components of the distributed system listed in the article? Can you predict other components required that are not mentioned in the article?**

<b>Components</b>	<b>Description</b>
NameNode	A dedicated server to store metadata
Inode	store file property information like permissions, namespace and disk space quotas.
DataNodes	A dedicated server to store application data.
Image and Journal	The inodes and list of blocks that define the metadata of the name system
NameNode	Hierarchy of files and directories that maintains the namespace and mappings to the DataNode
TCP-Based Protocols	All servers are interconnected and communicating through a TCP based protocol
HDFS Client	Users access the files through the HDFS client, the client has a library that exports the HDFS interface. Once done the user can read, write, delete, and create directories.
CheckpointNode	It combines the existing checkpoint to create a new checkpoint and an empty journal.

BackupNode	A new feature integrated incase of a a failure to the NameNode which keeps a backup of the NameNode.
Block Placement	It used for large clusters to distribute the nodes across multiple racks.
Replication Management	It used to verify all block have the number of intended replicas.
Balancer	It is used to balance the disk space utilization on a HDFS cluster by moving DataNode(s) replica(s) from places with large utilization to less utilized places.
Block	A memory section where data is held.
Block scanner	A component run by each DataNode that will verify its records through a stored checksum
Inter-Cluster Data Copy	A tool called DistCp that interacts with the MapReduce to map the taks copies of the source data to a destination file system.

Some predicted components that could have been used are things such as a YARN in a MapReduce Use case, as it is an integral part of Hadoop and as Yahoo! is reliant on the API, it would be able to discuss how its framework would be beneficial to applications that wouldn't need to worry about overloading the nodes. Another component is discussing more rack awareness when talking about replica storage so we are aware of its reliability and its bandwidth speeds, but the Rack Awareness algorithm would help deal with that as it can prove to be very difficult.

## **6. How are middleware transparencies provisioned within the described image-sharing system?**

Middleware is a virtual layer that lies between software (applications) and hardware. This allows for resources and networks to be hidden, or make it appear that the system is on a singular system but is distributed throughout many computers. Middleware makes development an easier task due to these portions being hidden/masked and makes the system easier to manage/maintain.



**7. An HDFS is a distributed system, why do we build distributed systems? What are the advantages? What are the disadvantages?**

The reason we build distributed systems is that it brings in many difficulties that we may have faced in the past. We can now have easier testing, smaller deployable units, and being able to alleviate the need to be able to do all this on one single computer. Having the option to run a system where the components are on different computers but using the same network allows us to be able to use larger platforms seamlessly. Before it would be very expensive to have to work with a monolith server but now because of node failures but thousands of nodes are able to rely on it in that case because they are also more flexible than a centralized server. For example, when traffic is high more resources can be assigned to the system to accommodate the increased traffic.

Advantages	Disadvantages
Availability as the system is always available to the user.	A distributed system is complex and hard to deploy.
Scalability through horizontal scalability which allows the adding of new servers to the existing resources to meet the requirements	The data may be lost within the network as it moves from node to node. Communication Errors: In the case where messages are passed from system to system, there may be a specific sequence that the messages must arrive in, so checks must be put in place to assure that messages arrive in order.
Performance – The system allows the users to run queries faster due to the database being close to the users since the databases are at various locations.	It is difficult to manage security as the systems/node and the network needs to be secured. The nodes and connections are in open connections and with so many nodes it's very difficult to keep track of sensitive information in each exact node.
Greater efficiency: Traffic is sent to various locations rather than one centralized system.	Cost: The initial cost of setting up a distributed system is a high investment and can end up being very expensive as it is using many different software and hardware devices, as well as the cost of maintenance can rack up very quickly.

Cheaper (over time): implementing distributed systems has an expensive initial investment cost, but the cost of investment over time more than makes up for the initial cost while compared to centralized stems.	Overloading can occur in a system if all the nodes are trying to send data at once.
Reliability: A distributed system can continue to function even if some nodes fail. The overall efficiency may decrease when these failures occur, but the system will still function.	
Failure: The way a distributed system handles failure is fantastic because as we know something like Yahoo! has 4000 nodes and if one node fails it will use the block scanner to replace it as it is needed within a few moments, and statistically speaking 0.8 nodes fail a day so a distributed system is great for that.	

## 8. What applications distributed systems are most suitable for?

Distributed systems are most suitable for things such as multimedia, social networking, and multiplayer video games. For all the things we have previously mentioned, things such as social networking would desperately need a distributed system as it can be running something like a user application that would run on a server and allow users to interact with each other sending information through the nodes on different devices. Multimedia apps such as Spotify that can be run on either your phone or your computer or through your car or other devices would also need to use a distributed system, and the same applies to multiplayer video games.

## 9. Why distributed systems are hard?

The biggest issues within distributed systems are latency, memory access, and partial failures. These are generally difficult because of 8 main issues: networks are unreliable, latency is not zero, there are bandwidth limitations, networks aren't always secure, topology is constantly changing, there are many users and administrators, it's expensive to transport data, and networks are often not homogenous. This can lead to many communication failures between computers within the system. All these distributed systems can be extremely unreliable because of the many things that can go wrong when the computers miscommunicate such as dropped messages or spoofed messages. In a distributed system we also require consistency, however, due to latency it is hard to maintain linearizability. On top of the technological challenges, there is a level of uncertainty introduced due to the people working on these increasingly complex systems. Mental models and understanding are also becoming more difficult as these distributed systems necessarily become more complicated.