Logo, company name

Description automatically generated

**Faculty of Engineering & Applied Science**

**SOFE4790U – Distributed Systems**

**Homework: HDFS as a Distributed System**

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

|  |  |  |
| --- | --- | --- |
| **First Name** | **Last Name** | **Student ID** |
| Abdul | Bhutta | 100785884 |

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. Lastly, file management concepts were introduced such as virtual memory, memory allocation, paging, and page table structure. A summary of topics is provided below.

* Added 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. (Owen)
* 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. (Owen)

|  |  |
| --- | --- |
| **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 |

1. **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. Distributed systems allow the users for resource sharing, scalability, performance, and fault tolerance.

* 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. (Owen)

1. **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 systems to implement the system. The four systems are shown below.

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Implementation** |
| Client-Server Model | Mainly used for resource sharing where the server handles data and resources requested by the client. An example is Netflix. | Graphical user interface, text, application  Description automatically generated |
| Three – Tier | 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. | Diagram  Description automatically generated |
| Multi – Tier | Used for when application needs to forward data or requests to multiple networks | Diagram, schematic  Description automatically generated |
| Peer – to - Peer Architecture | Each system is a node while acting as a client or server. An example of P2P is downloading torrents. | A picture containing diagram  Description automatically generated |

1. **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 many nodes/systems at different locations. Although in a distributed system, if one of the databases fails, not all users will be impacted but 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

1. **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?**

|  |  |
| --- | --- |
| **Components** | **Description** |
| NameNode | A dedicated server to store metadata. |
| DataNodes | A dedicated server to store application data. |
| TCP-Based Protocols | All servers are interconnected and communicating through a TCP based protocol |
| HDFS Client | Users access the files through the HDFS client. |
| 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 | Used to balance the disk space utilization |
| 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. |
| 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. |
| Inode | store file property information like permissions, namespace and disk space quotas. |
| Image and Journal | The inodes and list of blocks that define the metadata of the name 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. (Mamun)

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

A middleware is a virtual middle layer between the applications and the hardware which runs the processes above the layer and all the resources are below the layer, this allows for resources and networks to be hidden away.

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

We build distribution systems because they are more reliable and efficient that a monolithic system. Horizontal scalability is cheaper than vertical scalability after a certain threshold which also allows for fault tolerance and faster response time from the systems. The advantages and disadvantages of a distributed system is shown below.

|  |  |
| --- | --- |
| **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. |
| 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 need to be secured. |
| 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. |  |

1. **What applications distributed systems are most suitable for?**

A few examples of distributed systems are Netflix, social network websites, online banking, and online gaming where resource sharing, availability, performance, and reliability is the main priority.

* 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. (Mamun)

1. **Why distributed systems are hard?**

Designing and implementing a distributed system is hard because the overall system is complex, and many failures can occur during the initial phase in hardware or software. The software written by the developer may not perform the intended way it was designed which may lead to failure. There may be a network failure or glitches which may cause the overall system to fail.

* 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. Due to all of 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 is also becoming more difficult as these distributed systems necessarily become more complicated. (Alex)