Mamun Hossain - 100553073

Introduction:

The best way to describe what a distributed system is, is that its a computing environment where many different components are being used in tandem with other computers on what we call a network. Different devices work in unison that make it cohesive to make it work.

These devices and networks are what make a distributed system and they are crucial in our understanding of computer science, but that being said it would be next to impossible if one single computer were to do it alone, hence the name “distributed” plays a crucial role where it distributes its load across the network to different devices.

The way a distributed system works is that you would first need a way to operate through the internet, and what is ever evolving, the cloud system. The core process for a distribution is as follows; it starts with a simple need, which then creates a final product in the end. But that being said, a system just doesn’t have a stop and a finish, but several different points along the way that we call “nodes”. A system can have one node, but the bigger a system gets, the more nodes we end up needing. For example, cell phone networks are very intricate examples of a distributed system as it shares its capacity over several different devices and systems and networks. But it can be as simple as a client server system that most people use it for, that just have several computers connected to each other which has a main server that it uses for storage or data processing.

With that being said, a large reason to study distributed systems starts with something as simple as an OS, and touching on OS’s, we will discuss what we have learned in our previous years of study for it.

OS Course Summary

In our previous year, we had touched base on the Operating Systems course where we had learned and applied different ways an operating system is built, how it interacts with different commands, its architecture, and how it allocates itseslf within the commands given within the computers capacity. The ultimate goal was to understand load managing of the system and how they are able to utilize the computer parts that are at hand (eg. the CPU, RAM, HDD storage). This was called scheduling and using this knowledge we are able to apply this into our knowledge with distributed systems, and with this knowledge we will also learn how to manage the system sharing its resources and how they allocate on the processors, as this is the goal to Distributed OS. Some ways that we can implement them are the following: Dynamic and FIxed Partitioning, Simple and Virtual Memory Segmentation.

Comparison of functionalities between Centralized File system vs Distributed File System

Let us begin by talking about Centralized File Systems. We know that they are constructed in a simple manner and they are easy to understand. It works by using one or more client nodes that connect to a single server and uses a client server architecture to accomplish this. An example of this would be using a single central unit that coordinates all the other nodes in a system. The components in a centralized system is very simple, all it has are the server, the network, and the node. Although they are very easy to use, without the connected network they are useless. A use case where you would use a Centralized File System would be your personal computer.

A Distributed File System can contain thousands of nodes, and every node has its own decision. We are aware in the article that Yahoo! has roughly 4000 nodes in its system. This is a result of being able to allow multiple devices to coordinate on a server. Each node can fail independently but the network would still function as is, due to replication being able to save that node. The components in a DFS are a Node and the Network. Although a DFS helps overcome many of the issues we have with a DFS is that it can make it very difficult to work with different algorithms as there is no consistent clock frequency, creating all sorts of issues with the ordering of events. That being said, it is very useful for anything to do with social media or online games where a centralized file system would fail.

What are the components of the distributed file system in the article? Can you predict other components required that are not mentioned in the article?

In the article we can talk about three different things: The architecture, The File I/O Operations and the Replica Management. Within these, the following components are very important:

- Architecture:
    - NameNode
        - Hierarchy of files and directories that maintains the namespace and mappings to the DataNode
    - Image and Journal
        - The inodes and list of blocks that define the metadata of the name system
    - DataNodes
        - The node that sends the NameNode data reports via heartbeats and stores the application data
    - HDFS Client
        - The API or Library that exports the HDFS filesystem interface to the client that creates a new file
    - Checkpoint Node
        - A node that serves client requests on top of creating periodic checkpoints to protect file system metadata
    - BackupNode
        - Similar to CheckpointNode, can make periodic checkpoints but also adds recent images of the filesystem
    - Upgrades and Filesystem Snapshots:

        - Helps administrators to constantly save the state of the file system just in case if the need to upgrade comes across or any data loss occurs that it is possible to revert back to the namespace at the original point in time.
- File I/O Operations and Replica Management
    - Block Placement
        - A cluster topology to be able to spread the node across several racks so it needs to go through different switches to increase efficiency
    - Balancer
        - A tool that balances disk space usage on a HDFS cluster
    - Block Scanner
        - Uses the DataNode to run a scan to ensure that the replicas are stored with the checksums that are taken and to match the block data that was in the snapshot previously to prevent over replication
    - 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 about the 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.

How middleware transparencies are provisioned within the described image sharing system?

As we know, middleware is a layer between the software layer and the applications and objects. The way that the middleware is providing transparency is by using the blocks to cover up the failed nodes of the memory operation to allow seamless operations. The goal of the middleware is to simplify the connectivity between the nodes, the user to the application, and the components of the application, and with the replication and the components, we are able to prove that the HDFS system does this with ease.

As HDFS as a distributed system, why do we build distributed systems? What are the advantages and 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 having node failures but thousands of

nodes to be able to rely on in that case. Because they are also more flexible than a centralized server. Some advantages and disadvantages to a Distributed System are as follows:

Advantages:

- Reliability
    - A distributed system such as the HDFS can handle errors efficiently
- Low delay rates
    - It makes the user that is using the application have an easy experience
- Scalability
    - The ability to change the size of a system, and a distributed system can do that flawlessly
- Data sharing
    - The core feature of a distributed system is allowing easy data sharing with a system because of the way the nodes in a cluster are connected
- 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.
- Efficiency
    - A distributed system has high efficiency as it can work with several computers at a time to make life easier for users as they can save time.

Disadvantages:

- 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.
- Overloading
    - Overloading can occur in a system if all the nodes are trying to send data at once.
- Security issue
    - Like any device, security issues can always occur but that being said, in a distributed system it is a major disadvantage because the nodes and connections are in open connections and with so many nodes its very difficult to keep track of sensitive information in each exact node.
- Data loss
    - Although node failure has a great failsafe in place, the risk of data loss can still occur in case the node does die midway, the 0.8 node that dies, it can also lead to losing important crucial messages which can make or break an organization if it is not set up properly.

What applications are distributed systems 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.

Why are distributed systems hard?

According to the video, the reason why distributed systems are so hard is because in a distributed system, there are a lot of issues such as memory access, partial failures, user errors, and latency. That being said, we can narrow it down to the 8 major issues: you can never make the latency turn to 0, the networks are not always stable, the bandwidth limitations, security issues with the network, flat topology isn't useful so we always have to transform our cluster topology, there are too many users, the expense is too high, and the networks do not always interact with each other in the way that we need. As a result, this can cause many issues to the system, the user application would be unpleasant in turn making things more difficult to use a distributed system. On top of these issues, distributed systems are constantly evolving, and there are always people working on these issues and trying to create fixes but human error can result in a much greater issue as well. Because of how difficult distributed systems are, it takes a lot of knowledge and the margin for error is significantly higher.

Generic Homework Questions:

How are transparencies provisioned within the described image sharing system?

The features for sharing HDFS shows the transparency when using a HAR file and how it can be used as the input to a MapReduce job. The way this would work is by maintaining the status quo in fairness regarding the sharing that needs to be enforced, but also when a user application has several hosts writing the data and the protection against applications that is also important.

What resources are managed and how?

The resources that are managed are the NameNodes and how Replication Management helps with this. We know that the NamdeNOde makes sure that each block has the accurate number of replicas, but the NameNode can also detect if a block has too many or too few replicas as well. The way that this is in two ways. One, if a block becomes under replicated, then it is put in a priority queue to receive a replicated block. When a block becomes over replicated,

then the NameNode will pick a replica to remove, but it will not reduce the number of racks. The way this is managed is that if a block has only one replica, it will

Compile a list of software requirements of the system described in the article?

The software requirements of the system in the article would be simply the Apache Hadoop 2.5.2 or higher application.

One definition of a system is: a set of components that work together to achieve common objectives.

The definition of the system in the article is to have a large network of nodes being able to send information out to other applications without having any delay or issues. In this case, the replication of nodes as needed was identified by the NameNode and the Block Scanner looking for corrupted blocks and fixing it accordingly.

How are the different components of the system "coordinated" to work together?

The way the different components worked together in the HDFS system is by making sure the files are separated into different blocks, which then are stored in to a DataNode. THen the DataNodes are put together into a cluster, as we've seen how the cluster topology works in the article. From there, they are linked to the NameNode, which then in turn redistributes the replicas across the cluster to where the user has a need for it.

How does the system described in the article provide the following requirements:

Resiliency in case of failures

For resiliency, we can see that when the DataNode undergoes Replication Management, it always makes sure that it has the right number of replicas, and when there isn't the NameNode knows the priority that it needs to take care of the under or over replicated node.

Scalability in case of expansion in various parameters

For scalability, we can see that when the DataNode is ran through the block scanner, it knows that if the read bandwidth needs to be adjusted to complete the verification that is required of the data system.

Maintainability in response to continuous change

We know for maintainability that the NameNode will always maintain the namespace tree and will map the topology for the DataNode as according to the user needs and what the administrator sets it as.

How does such benefits and challenges reflect on the user? How does these also reflect on the remaining functions of an OS?

The benefits that reflect on the user is ease of access for their application, having a large storage of data, being able to quickly detect any hardware failures and can quickly repair any nodes if needed when sending any types of issues, but some issues a user can face is issues with file modification as a lot of files aren't supported and the number of small number file storages are also very limited. This affects the remaining functions of an OS because when you can only have a large file storage it can end up putting a lot of load on the cluster rather than separating into multiple clusters and taking the load of the DataNode and being able to send things with lower latency.

How can/has the proposed system been extended? Do some further research on the internet.

There is an open source contribution that is called MapReduce, and this is actually solving a problem similar to the previous issue we have. Using MapReduce, we can increase the number of larger data to shorter jobs so instead of having one big cluster the many short jobs with big datas will also result in needing low response times. The MapReduce application will gather all the nodes to a single result and be able to send the information back to the master node and send it to the child nodes after.

Describe what distributed systems are

A distributed system is a system that allows resources to be shared by multiple devices over a network. An example of this is Yahoo!

Describe characteristics and requirements in distributed systems

The characteristics in a distributed system is resource sharing, scalability, transparency, fault tolerance, how concurrent the software is to each other and the openness of the software.

How this affects company's hardware and software

This affects the company's hardware as it would require many devices to be connected to a low latency network that would in turn be very expensive in the beginning to set up the server and have hundreds and thousands of computers. For the software aspect, the framework needs to be set up and to maintain that as well will also be expensive.

Identify Key building blocks/ Components and common characteristics

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

Identify NFR and Basic Design Issues

Some design issues can be that the software is very little in comparison to a PC but with the cloud network evolving at a rapid pace, this problem will soon be alleviated. Another problem can be privacy because although it is run on a secure private network, it can potentially be viewed without the owner's consent. The non functional requirements in distributed systems are as follows:

- Supportability
- Usability
- Lifetime Cost
- Scalability
- Reliability
- Performance
- Security
- Flexibility
- Reusability
- Maintainability