Faculty of Engineering and Applied Science

SOFE 4790U Computer Networks

Group 17 CRN 44425

HDFS as a Distributed System

Name	**Student #**
Alexander Campbell	**100703650**

Intro Objectives

A distributed system is one in which a network of connected computers spreads out tasks that have been sent through the use of services that distribute the resources of the computers for various computing needs.

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.

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 actually communicate with each other such as during run-time or during setup.

Non-functional requirements are requirements that aren't rigidly defined such as functional requirements. These can be requirements that define what the designer considers a "quality" distributed system such as responsiveness, scalability, robustness, availability, and transparency. However, of all of the requirements, transparency is generally the most important. As stated earlier, transparency essentially allows you to use a distributed system as a centralized system and hide the complexity away from the user. A good distributed system will feel no different than a centralized one.

OS Course Summary

In the past operating systems course, we learned about how operating systems function, how they are built, and how they handle the computer's resources. The main function of a distributed OS is to distribute various tasks to the lower level hardware. This is done by managing the system's resources and allocating them based on the current tasks and what processors are available. This is called process scheduling and it is the basis of a distributed operating system.

How does a centralized file system compare to a distributed file system?

The largest difference in functionality between a centralized system and distributed system would be the reliability and scalability of the systems. A distributed system is inherently more reliable due to its fault tolerant nature, having a set of computers rather than just one makes it so that if there are failures they are not as noticeable as on a centralized system. In terms of scalability, as mentioned in the HDFS article, distributing computation across many computers can ensure that we are always economical in our usage of resources no matter what growth occurs. Another key difference is the maintenance aspect. In a centralized system the maintenance is much easier as it's on one machine, when you introduce many machines you require more overhead to ensure the transparency of the system to the user with load balancing, parallel programming, etc.

What components make up a distributed system?

In this article we can split the components mentioned within the article up into two groups, HDFS-specific components and general distributed system components:

HDFS-specific components:

- NameNode: maintains namespace tree and mappings to DataNode
- Inode: representations of files and directories stored on NameNodes. Record attributes such as permissions, access times, NameNode, etc.
- DataNode: servers that store application data
- Image: metadata of name system made up of inodes and lists of blocks
- HDFS Client: the interface in which a user can access the HDFS filesystem, allowing operations such as reading from or writing to files
- Checkpoint Node: this is an alternative role of the NameNode specified at start time, it will periodically existing checkpoint and journal to create a new checkpoint and an empty journal. This helps to protect the system's metadata
- Backup Node: similar to the checkpoint node, however also maintains an updated version of the image of the filesystem namespace
- Balancer: tool used for balancing disk space within an HDFS cluster by moving DataNode replicas to lower utilized areas from areas of high utilization
- Block scanner: is a component run by each DataNode that will verify its records through a stored checksum

General distributed system components:

- Block: portion of memory in which data is stored

Predicted components:

- HDFS Privileged Client: this client would be used by system administrators for maintaining rather unlike the regular client which is used for data interfacing
- Load balancer: some sort of technology that is used to balance the HDFS clients accessing the data so the computation is spread out throughout the system

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

Middleware is a virtual layer in between the hardware and software. It allows for an abstracted form of communication that provides transparency to the user by masking the process to the operating system interface. The goal of middleware is to make it appear to the user that the operating system is on one system when it is in fact distributed across several computers.

Why do we build distributed systems? What are the advantages? What are the disadvantages?

We build distributed systems to make up for where centralized systems fall behind. For example centralized systems have lower availability, and reliability while compared to distributed systems. In essence, distributed systems are more flexible than centralized systems. Additionally, distributed systems are built as they have a greater ability to scale with fluctuating traffic. For example, when traffic is high more resources can be assigned to the system to accommodate the increased traffic.

Advantages:

- Greater efficiency: Traffic is sent to various locations rather than one centralized system.
- Scalability: When the system increases in traffic/popularity bottlenecks may be encountered and with distributed systems it is easy to add additional resources to accommodate the increased traffic.
- 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 systems.
- Reliability: A distributed system has the ability to continue to function even if some nodes fail. The overall efficiency may decrease when these failures occur, but the system will still function.
- Open: The system can be accessed remotely, but also locally.

Disadvantages:

- Complexity: Maintaining and troubleshooting a distributed system is difficult, as there are many points of failure.
- 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.
- Security: Data is stored in different locations in a distributed system, and this data may be sensitive and this information needs to be secure no matter the location it is in, as well as making sure the correct access is granted upon requests for access (which can be challenging with distributed systems).
- Increased Software Development Costs: Hard to implement distributed systems and due to that, it is more expensive.

What applications distributed systems are most suitable for?

The applications most suited for distributed systems are when resource sharing, reliability, and scalability are important. Some examples are: streaming services like Netflix, banking services, file sharing services, the internet, and telecommunications services.

Why are distributed systems 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. 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.