**History of File Systems**

A file system is the methods and data structures a computer operating system uses to retrieve, catalog, and store files. Without a file system present in an operating system, data stored on a machine will gather into one large body of data with no efficient way of tell where one file starts and the other begins. By giving each piece of data a name, the information can easily be identified and stored into different directories for easy access. There are a wide variety of different file systems, and each one has been developed to serve a specific purpose. Some of these different file systems have been designed to share files between computers, transfer files using network protocols, and quickly store large amounts of data known as big data.

The first file system was created in 1973 by Gary Kildall and was called CP/M. Gary was a programmer who wrote a program called “Control Program for Microcomputers” (Reimer, 2008) that stored files and ran programs from a floppy drive. The CP/M file system, stored files in a completely flat hierarchy with no directories, which was limited to eight characters plus an extra dot three character extension that helped determine the file type. What made Gary’s program unique is that it “separated all the computer specific bits from the rest of the operating system” (Reimer, 2008). The program became what is now known as the basic input out system (BIOS). Eventually, Microsoft bought out the program and renamed the package MS-DOS, which became the main operating system for many industrial and personal computers we see today.

Back when the first file system was created, memory was scarce and expensive. Not many people had a personal computer, which meant sharing and storing a large amount of data wasn’t an issue. Fast forward to today and now everyone has multiple personal computers that constantly share information and data through a computer network. In order to cope with this “Big Data,” a new file system had to be developed that would quickly sort, store, and retrieve the enormous amount of data floating around the network. The Hadoop program created in 2005 by Doug Cutting and Mike Cafarella has proven to be very useful in sorting and managing large amounts of data efficiently. Now managed by Apache Hadoop, the “Hadoop Distributed Filesystem (HDFS) allows storage of enormous data sets through distributed clusters of servers and then analysis applications within each data cluster” (Proffitt, 2013). The program is designed to be robust and fail proof, it allows for “Big Data” applications to continue running on an individual server, even when a hard drive cluster fails. The program is made up of two core parts: a data processing framework, and a distributed filesystem for data storage. The data processing framework deals with the data itself, and is responsible for getting the data processed. The distributed filesystem is responsible for holding the actual data. Overall, the creation of the Hadoop filesystem has helped with managing large amounts of data quickly, cheaply, and efficiently, making it an ideal filesystem to use for processing big data.

**The Future of Hadoop Distribution Filesystems**

The next big wave for Hadoop data storage is providing an interface for programming entire clusters with implicit data parallelism and fault-tolerance. The first wave was about establishing technologies and making sure a good foundation was built so that individuals could build open source applications on top of the core program. Apache Spark is said to be the future of Hadoop, and is built off the whole HDFS software. Spark is an open source clustering computing framework; it can “provide programmers with a programming interface focused on data structures called resilient distributed dataset (RDD)” (Apache Spark, 2016). The RDD is maintained in a fault tolerant way through a read-only multiset of data items scattered over an array of machines. It was developed because of limitations in the “MapReduce” cluster computing model, which forces dataflow structure on distributed programs. The Spark program is currently being improved by a company called Cloudera who believe Spark is “great for specific use cases to being the default engine for MapReduce workloads as well as partner products” (Woodie, 2015). Cloudera will seek to improve Spark in four areas: security, scale, management, and streaming. By improving these main areas of the program, Spark can evolve into a powerful filesystem that can manage and organize a constant stream of live data. Ultimately Spark will be the core engine managing streams of data, phasing MapReduce out since it will become obsolete and unable to keep up with the speed of Spark. In conclusion, Spark is in the works of becoming the next core filesystem engine to handle the large amount of data found on computer networks, and will be able to quickly manage and store the information faster than any current filesystem used in today’s industry.

**References**

Apache Spark. (2016, April 18). Retrieved May 04, 2016, from https://en.wikipedia.org/wiki/Apache\_Spark

The Wikipedia page for Apache Spark gave a great overview of what the program. An individual reading from the web page can quickly get a great understanding of the Spark program and for what purpose it was developed. By reading through a few paragraphs, it was clear what Apache Spark does and how it is able to execute its tasks. In short, this Wikipedia web page will allow an interested reader to swiftly grasp the concept of Apache Spark and how the program manages data based off the Hadoop Distribution Filesystem software.

Proffitt, B. (2013, May 23). Hadoop: What It Is And How It Works - ReadWrite. Retrieved May 03, 2016, from http://readwrite.com/2013/05/23/hadoop-what-it-is-and-how-it-works/

The purpose of this article was to give the reader a broad overview of the Hadoop Distribution Filesystem. The article gave a good explanation of why the Hadoop program was created and how it processes the large amounts of data found throughout the computer network. The article also discussed the main components of the program and briefly described what each component was responsible for. In short, the article gives the reader a general explanation to the purpose behind Hadoop Distribution Filesystem, which is to quickly and cheaply store large amounts of data.

Reimer, J. (2008). From BFS to ZFS: Past, present, and future of file systems. Retrieved May 03, 2016, from http://arstechnica.com/gadgets/2008/03/past-present-future-file-systems/2/

The main idea of this article from ars technica was to give the reader insight on how the first filesystem was created as well as discussing the purpose of a filesystem in a computer. The article points out the past, present, and future of filesystems. The purpose of the article is to give individuals an in depth history lesson on filesystems and how they have revolutionized the way we use computers today. Overall, the article does a great job going into detail about the very first filesystem that was created in 1973 as well as discussing the future of filesystems and how they will aid individuals with storing through the large amount of data currently found on computer networks today.

Woodie, A. (2015, September 09). Spark Is the Future of Hadoop, Cloudera Says. Retrieved May 04, 2016, from http://www.datanami.com/2015/09/09/spark-is-the-future-of-hadoop-cloudera-says/

The main idea of this article written by Alex Woodie from Datanami was to inform the reader of the next steps to improve Apache Spark. The author gave a good overview of where Spark came from and mentions what is instore for the program. The main idea of the article explains how Cloudera has taken over the open source Spark software and is currently making improvements, which will allow the software to become the next core filesystem engine for processing large amounts of data. Overall, the article is a good source for understanding what will become of Apache Spark as well as the steps that are required to making it a very powerful filesystem to be able to handle storage and management of streamed data.