

Author: Scott Brisbane
Supervisor: Xiwei Xu

Background & Motivation

Traditional data analytics platforms, such as Hadoop on the Hadoop Distributed File System, are designed to operate on a known and controlled cluster of servers within an organisation, and on a dataset that is completely contained within the analytics cluster. With the increasing volume of data that society is generating, it is not hard to imagine scenarios where peer-to-peer based analytics could be of use, including cross-organisation, cross-region and cross-device scenarios.

Peer-to-peer based analytics could even be used to analyse personal data, where the data itself never leaves the individuals device - computation occurs on their device and only the final result is ever collected. Further to this, a peer-to-peer based analytics platform could eliminate the need for ingesting data from another file system into HDFS and improve decentralisation of data set hosting.

This project aims to implement an interface between Hadoop and an existing decentralised data storage system in order to enable such peer-to-peer analytics. Such an interface will allow MapReduce tasks to be executed directly on data stored in this alternate storage system.

Chosen Solution

The chosen solution was to implement Hadoop's Java based FileSystem interface for the decentralised, peer-to-peer file system IPFS (InterPlanetary File System). This interface would allow Hadoop MapReduce jobs to be performed directly on data hosted on IPFS. This same interface is used by Apache Spark, and would also allow Spark jobs to run on top of IPFS.



In deciding on which file system to use for the project, a number of alternatives were considered. IPFS was chosen primarily because:

- Files are replicated based on interest (peers that are interested in the data choose to keep a replica), improving decentralisation of data set hosting.
- Files are content addressed (using a hash of the file contents) which provides versioning guarantees.

MapReduce & MapReduce on IPFS

The MapReduce model (as used by Hadoop) is a popular approach to performing analytics on big data across a computing cluster. The basic idea is that programmers create MapReduce programs consisting of a Map and a Reduce phase. Input data is split into many different chunks and Map tasks executed on the data across all nodes in the cluster. The results from the Map phase are then passed to the Reduce phase which performs more computation before outputting the final result.

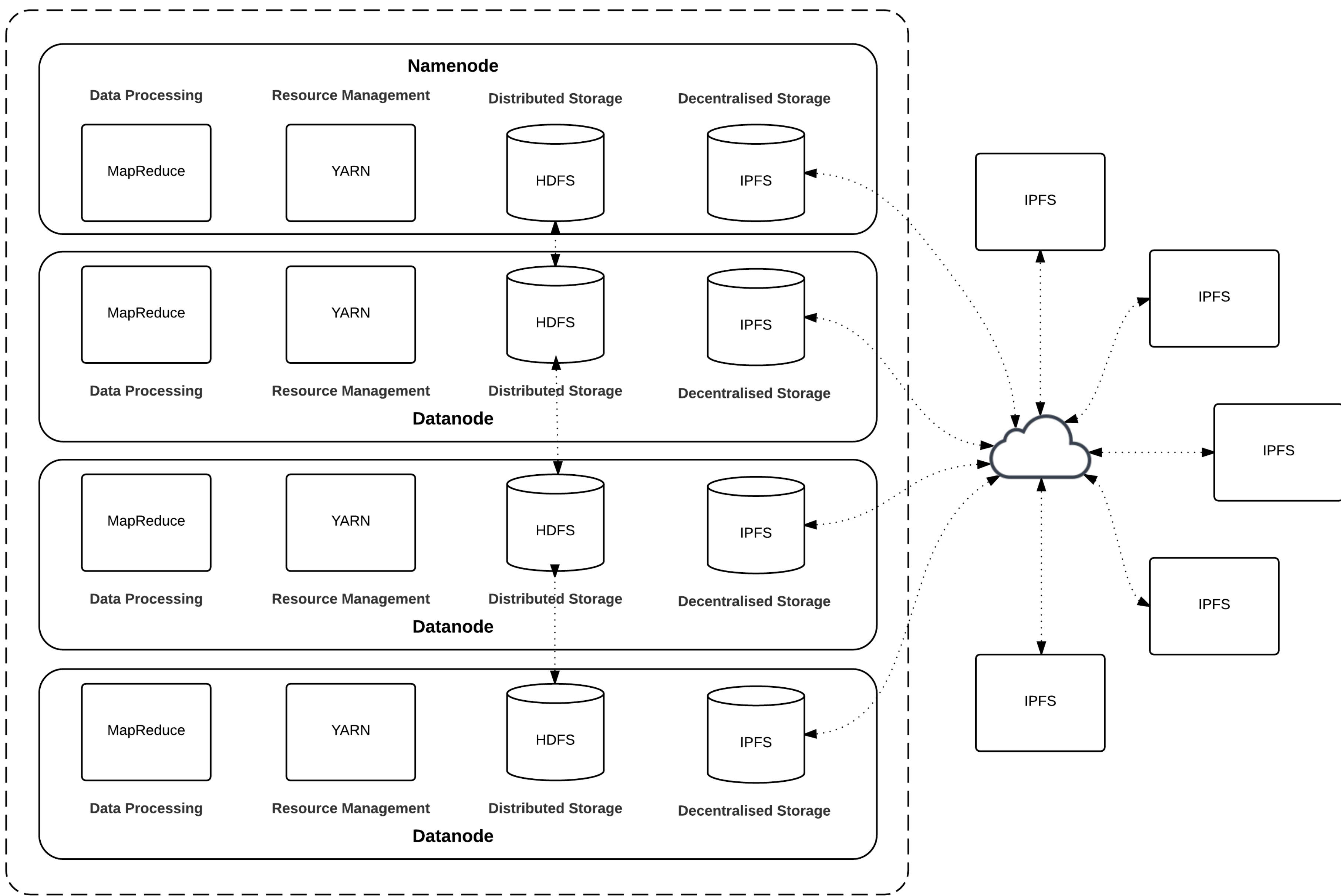
The input and output of the overall job, and of each phase, are typically stored on Hadoop's HDFS file system. This work enables the initial input and final output to be stored on the IPFS file system, but still requires HDFS to remain in place for any intermediate files (such as the output of the Map phase).

Deployment

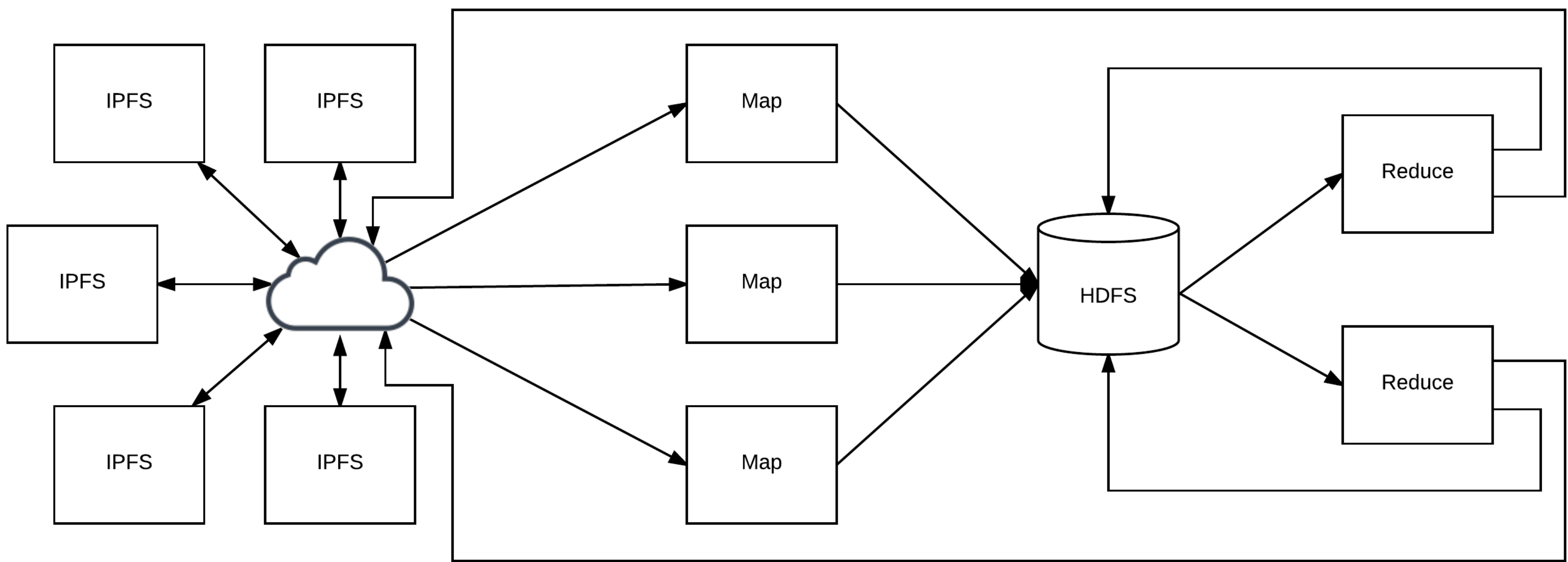
The system works by installing IPFS on each node in the Hadoop cluster and allowing Hadoop to access and create files on IPFS directly.

This is done by installing the completed FileSystem interface (as a jar file) on each node and modifying the Hadoop configuration to use the installed interface for file path's using the *ipfs://* scheme.

With IPFS installed on each node, the traffic between the DataNodes (slaves) and the NameNode (master) is reduced. This is important for performance in a peer-to-peer cluster.



System structure diagram showing the Hadoop nodes and their connection to IPFS



Typical data flow in the Hadoop on IPFS system

Evaluation

The two main evaluation criteria for the work were functionality and performance.

Functionality was tested by deploying the system to a cluster on AWS and determining whether the functional objectives of the work were met.

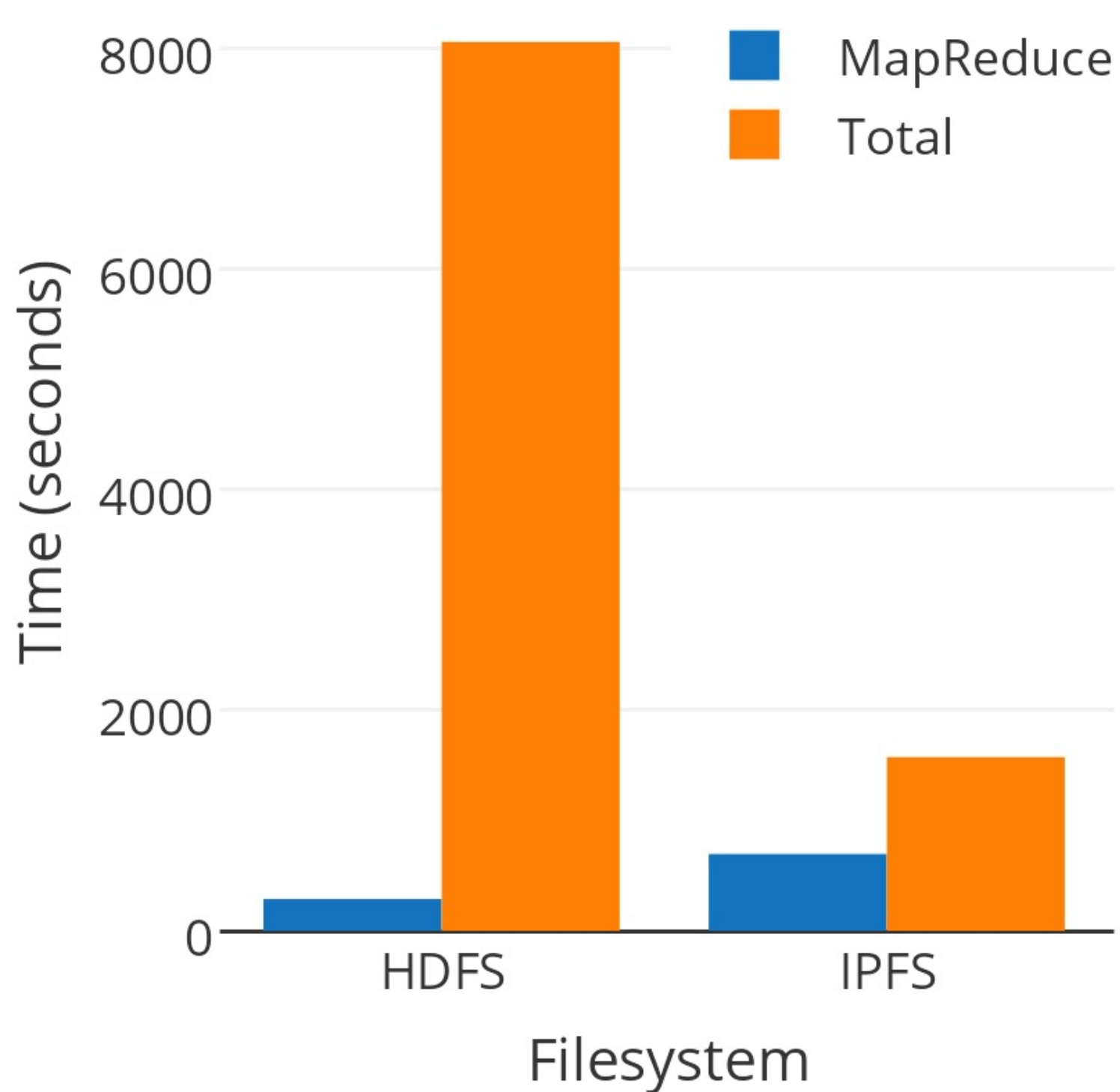
Performance was tested by deploying the system to AWS and comparing the time taken to complete a word count MapReduce program on our system running IPFS and a control running the standard file system HDFS. Two comparisons were made: one with all nodes in the cluster in the same AWS region, and the other with each node in a separate region. This cross-region comparison aimed to test the system in a scenario as close to its intended purpose as possible.

Results

The finished solution achieves its functional aims as it successfully enables cross-region, peer-to-peer based data analytics. It also improves decentralisation of data set hosting through IPFS' replication strategy and eliminates the need for ingesting data sets before performing any analysis.

As for performance, on a local cluster (all nodes in one region) the control running HDFS out-performs the IPFS solution when looking at the MapReduce execution time as well as the time to ingest a data set and run the MapReduce job. On a cross-region cluster, the HDFS system outperforms our IPFS based solution on the MapReduce task execution. When factoring in the time to ingest the data set however, the IPFS solution is considerably faster.

HDFS vs IPFS on a Global Cluster



Conclusion

The Hadoop on IPFS system successfully enables peer-to-peer based data analytics by providing an interface between a peer-to-peer file storage system and Hadoop.

The system improves decentralisation of data set hosting through IPFS' replication strategy that is interest based. This strategy means that the longevity of the data set is guaranteed as long as someone maintains interest in keeping a copy of it.

It also eliminates the need for ingestion of data (provided the data is hosted on IPFS) which in my testing has proven to be a significant performance boost for cross-region systems.