# Panthera: A Study of Caching in Distributed Computing

## Introduction

Distributed computing has seen tremendous growth in the past few years. In particular, Hadoop, an open source project that allows developers to write distributed applications, has found extensive use in academia and industry. For example, Yahoo, Facebook, Stanford Computer Science, etc. are all heavily involved with Hadoop development. But, Hadoop does not effectively utilize RAM memory and local storage to better performance. **This project develops and tests caching mechanisms, called Panthera, to reduce waiting time or latency in Hadoop,** thereby greatly increasing its efficiency and applicability in fields ranging from medical diagnosis to artificial intelligence.

## Problem

- Develop a caching layer for the Hadoop File System that can cache both data and metadata.
- The solution must not depend on a modified Hadoop version; it should be a "drop-in" addition.
- The layer should reduce file access latency/waiting time significantly in comparison to a vanilla Hadoop installation.

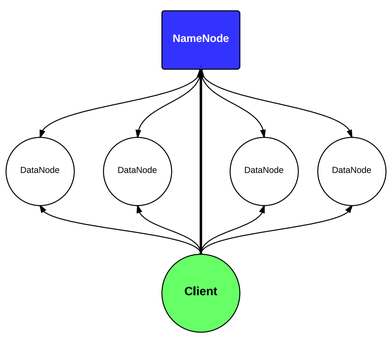
## Hypothesis

The cache layer designed (called **Panthera**) will have lower file access latency when compared to a standard Hadoop installation. Additionally, it will be possible to design **Panthera** such that the Hadoop source code remains unaffected.

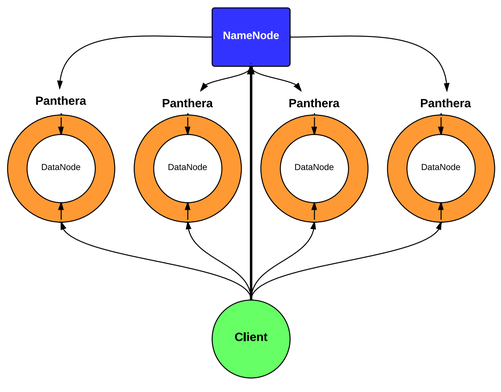
## Data vs Metadata

Data within the Hadoop file system consists of the actual contents of a file. Metadata is the information about files and directories, e.g. time a file was created, the list of files in a directory, etc. **Panthera performs both metadata and data caching; both have significantly different technical implications and challenges.**

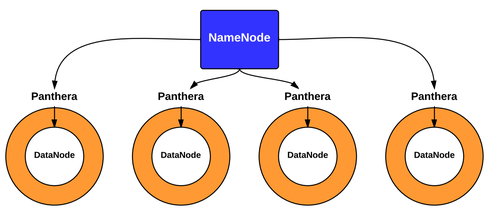
## Current Hadoop File System Architecture



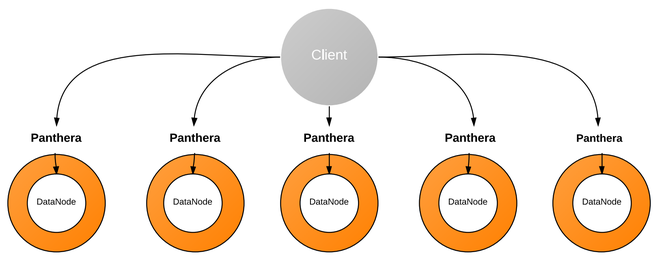

## Panthera Architecture




## Panthera Metadata Architecture




## Panthera Data Architecture




## Panthera - Implementation

- **Panthera is implemented on Google's Go programming language**. The language has excellent concurrency primitives and memory management, which are perfect for for software ,such as **Panthera**, that must run reliably with hundreds of clients.
- **Panthera** implements a large portion of the Hadoop File System protocol, enabling it to "speak" with the DataNode and NameNode.

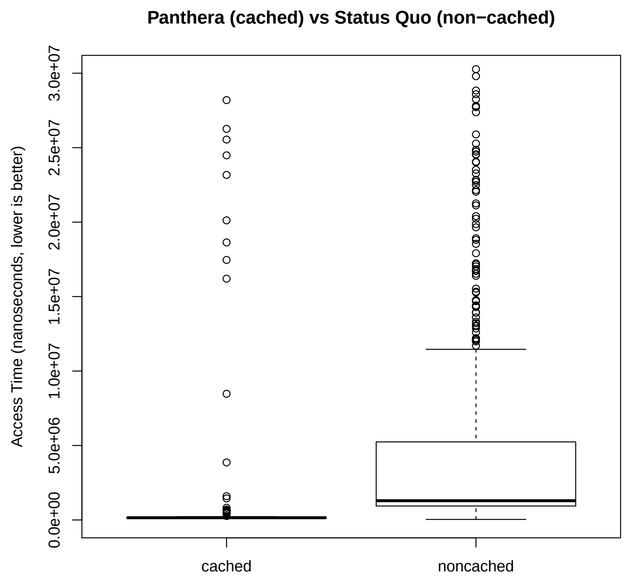
## Testing Methodology - Single Node vs Multinode

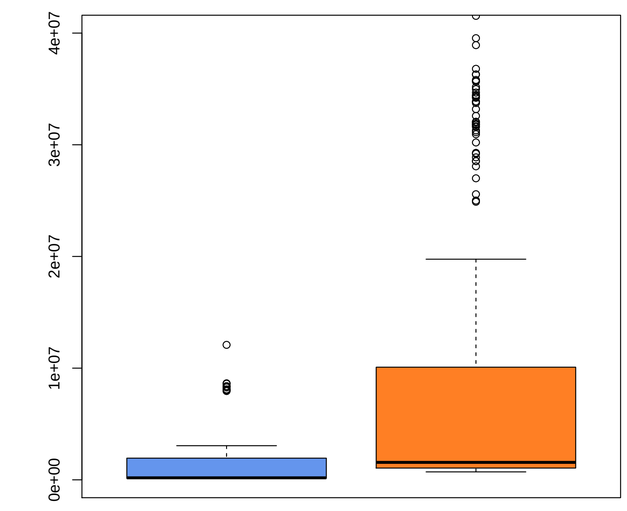
Testing procedures were conducted on both **single node** and **multi-node** setups.**Single node** testing shows Panthera's performance without regard to network latency, whereas **multinode** testing incorporates network factors.

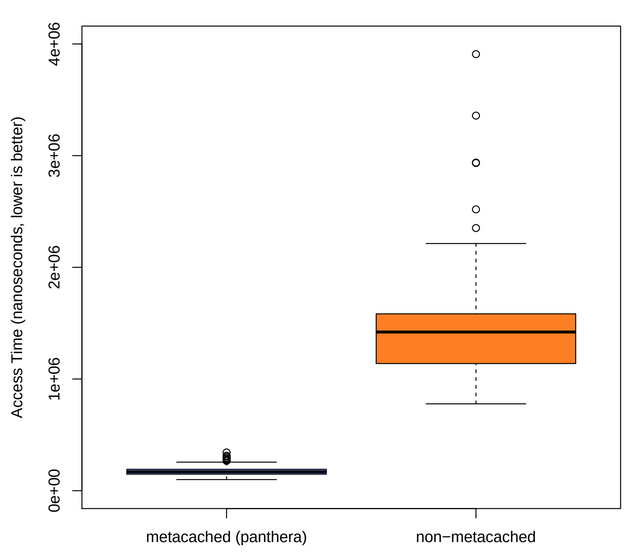
## Testing Methodology - Procedure

- **Metadata**: Repeatedly request directory listing for a directory with 100 files in it.
- **Data**: Repeatedly request file contents for a 1MB large file.

## Results




## Latency Types

In the Hadoop File System Architecture, there are several types of latency involved:

- **Network latency:** Information shared between client and server (e.g. between a DataNode and a client) has an attached waiting time. With **Panthera**, this latency is eliminated since the data is cached on the client.
- **Hard drive latency:** Reading from the hard drive has very significant latency. Hadoop generally reads from the hard drive when accessing files.
- **Memory latency:** Reading from RAM memory has an extremely small waiting time associated with it. **Memory waiting time is less than 1/1000th hard drive waiting time. Panthera converts hard drive latency to memory latency, thereby greatly reducing file access time.**

## Statistics

- **Data (single node, noncached)**
  - Mean: 12548998 nanoseconds
- **Data (single node, cached/Panthera)**
  - Mean: 511185.3 nanoseconds
- **Noncached/Panthera**: **95.9% improvement in latency with Panthera**, waiting time of noncached is 24.5 times higher than that of Panthera.
- **Metadata (single node, noncached)**
  - Mean: 8445265 nanoseconds
- **Metadata (single node, cached)**
  - Mean: 1055012 nanoseconds
- **Metadata (multinode, noncached)**
  - Mean: 1638184 nanoseconds
- **Metadata (multinode, Panthera cached)**
  - Mean: 176868.7 nanoseconds
- **Metadata - noncached/Panthera: 90.3% improvement with the Panthera metadata cache**. Panthera (developed during this project) decreases waiting time by a factor of 8.

## Conclusion

The results validate the hypothesis presented. Panthera was successfully developed as a "drop-in" solution and it was able to reduce file access latency in comparison to a standard Hadoop installation. In particular, **the Panthera data cache led to a 95.9% or 9 fold decrease in waiting time** and the **the Panthera metadata cache also gave a 90.3% or 8 fold decrease in latency**. Additionally, the **the cache developed during this project had a far lower standard deviation for access times and fewer outliers**. For large scale systems such as those in use at major companies and universities, such a drastic decrease in latency can have incredible effects on the efficiency of distributed computing systems.

## Applications

Hadoop has diverse applications and since Panthera was created as a "drop-in" solution, the **applications span an extremely wide range of disciplines**.

- Bioinformatics (e.g. CrossBow from Johns Hopkins University)
- Numerical/scientific computing
- Image processing
- Artificial Intelligence/Machine Learning (e.g. Apache Mahout)
- Market Research (e.g. Target)

## Future Work

- **All code related to this project will be open sourced** to further development in distributed computing.
- Currently working on testing a **bioinformatics toolkit with Panthera**; initial results are very promising and show significant reduction in total running time.
- Smart scheduling using Panthera; arranges tasks so that the cache is used most efficiently.