Panthera: Caching and Cache-based Schedulingin Distributed Computing Systems

Introduction

Distributed computing has seen tremendous growth in the past few years. Essentially, it allows computational algorithms to execute over a network (i.e. ***cluster***) of computers. This project is focused on the ***Hadoop*** distributed computing system, which has found extensive use throughout academia and industry. For example, major software companies such as Yahoo, Microsoft, Facebook, Cloudera, etc. all utilize Hadoop heavily. Hadoop provides a mechanism (called a ***distributed filesystem***) that allows each computer (i.e. ***node***) in a cluster to access the same set of files. However, it fails to utilize local Random Access Memory (RAM) resources to reduce the waiting time (i.e. ***latency***) associated with obtaining these files. **This project develops and tests scheduling and caching mechanisms to reduce latency in Hadoop, thereby greatly increasing Hadoop's efficiency and applicability in fields ranging from bioinformatics to artificial intelligence.**

Past Research

- Several sources (e.g. T. White) have identified latency as a core problem in Hadoop's architecture. However, cache-based latency reduction has been unexplored.
- Schedulers have been developed for Hadoop (by M. Zaharia, Y. He, etc.), though none have utilized file storage information.
- A wide range of algorithms for operating system-level scheduling exist, but Hadoop's architecture requires a fairly ad-hoc approach.

Caching

There is significant waiting time/latency associated with retrieving files from distributed filesystems. **Caches** are used to store some files in local RAM so that they can be accessed much more quickly. Panthera implements a cache and a cache-based scheduler for Hadoop.

Problem

- **Develop a caching layer** for the Hadoop Distributed File System that can cache **both data and metadata**.
- The solution should be a **"drop-in" addition**, allowing for easy adoption and integration into existing systems.
- The layer **should reduce file access latency** significantly in comparison to **the control group: a standard Hadoop installation**.
- **Develop a scheduler** for the Hadoop which **schedules jobs based on what is available in the cache**. In addition, it should be be able to **prefetch files that could be needed in the future**.

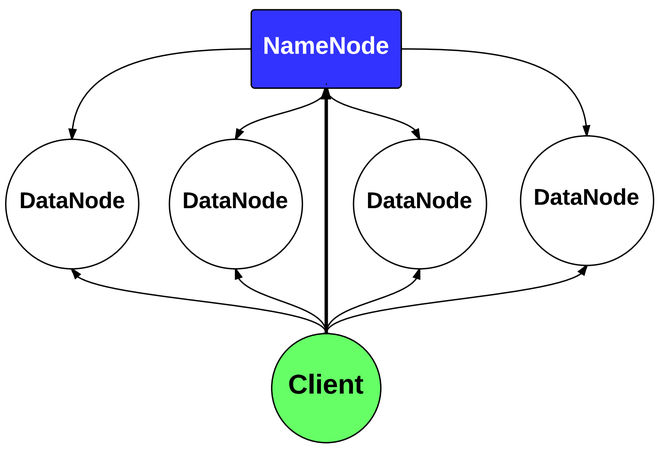
Hypothesis

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. It will also be feasible to create a cache-based scheduler for Hadoop.

Data vs Metadata

- **Data** within the Hadoop file system **consists of the actual contents of a file.**
- **Metadata is 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

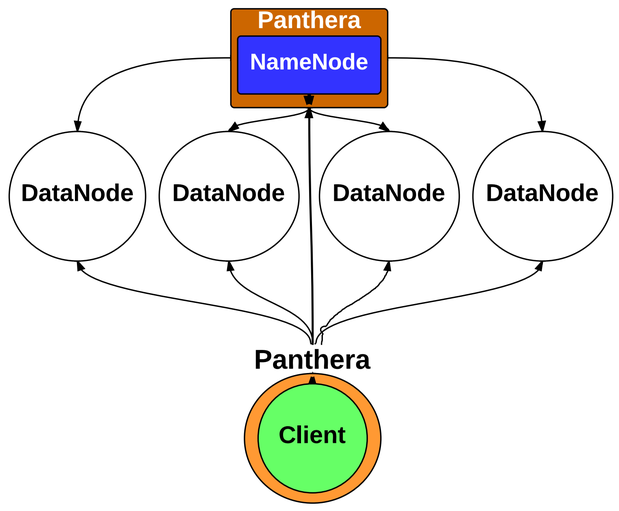


Latency Types - Theory

In the Hadoop File System Architecture, there are several types of latencies involved (listed from highest to lowest magnitude):

- **Network latency**
- **Hard drive latency**
- **Memory latency:** Reading from RAM memory has an extremely small waiting time associated with it ($< 1/100$th that of the hard drive). **Panthera converts hard drive and network latency to memory latency, and thus greatly reduces file access time.**

Panthera Architecture



Panthera - Implementation

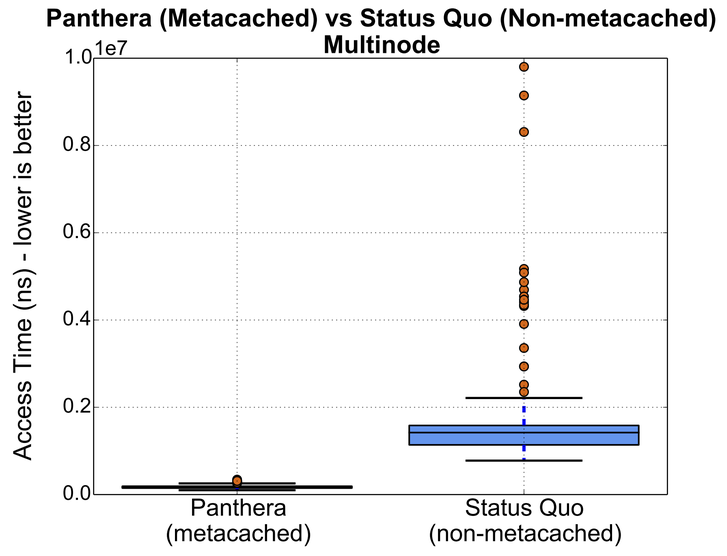
- **Panthera is implemented in Google's Go programming language** which has excellent concurrency primitives and memory management, making it perfect for software that must run reliably with hundreds of clients.

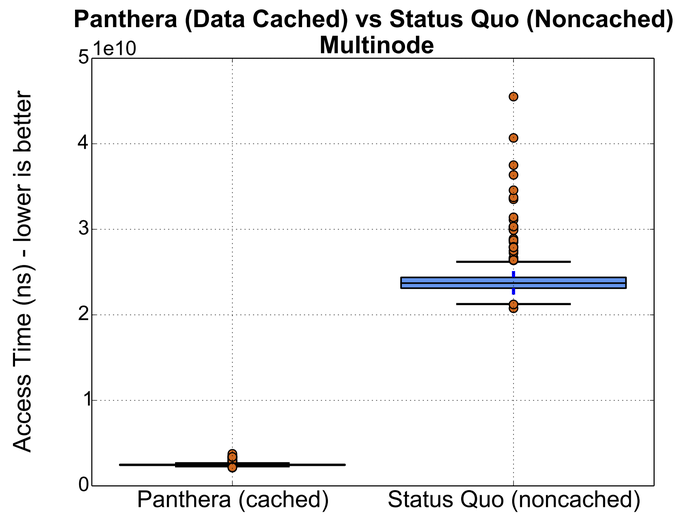
Latency Testing Methodology

Client and server are on separate machines for latency testing.

- **Metadata**: Repeatedly request directory listing for a directory with 100 files in it.
- **Data**: Repeatedly request file contents for a 64MB file (64MB = 1 block in Hadoop).

Latency Results

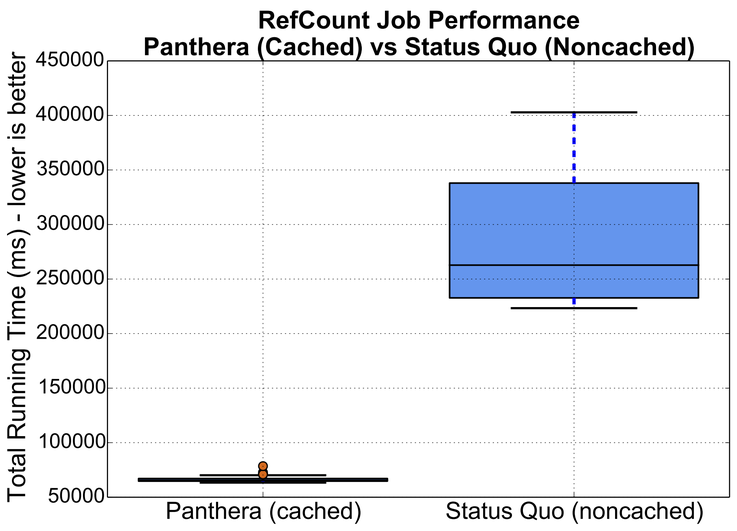




Procedure Testing Methodology

Panthera was also tested with running times of algorithms executing on Hadoop (as opposed to only downloading data or metadata). The input sample to both algorithms tested was a repeated copy of "The Adventures of Sherlock Holmes" (192 MB). Algorithms tested:

- **WordCount**: A classic distributed algorithm that counts occurrences of all words in a given text.
- **ReferenceCount**: Counts occurrences of words in a given text, but only considers the words that are in a given reference file.



Panthera Scheduling Algorithm Theory

- **Problem**: Given $\mathbf{J} = J_1, J_2, ..., J_k$ jobs to be run on a Hadoop cluster with caches $\mathbf{C} = C_1, C_2, ..., C_n$, what ordering will best take advantage of the caches?
- **Panthera job score**: Each J_i accesses a set of blocks F_i and each C_i holds a set of blocks B_i. Every J_i is assigned a job score x_i:

$$x_i = \frac{|F_i \cap B_t|}{|B_t|} \text{ where } B_t = \bigcup_{j=1}^{n} B_j \quad (1)$$

- **Panthera scheduling algorithm**: Sort $\mathbf{J}$ in descending order with respect to the x_i associated with each J_i. While J_i is running, Panthera may prefetch data for J_{i+1} to further reduce latency.

Statistics

Type	Mean time improvement
Metadata	8.13x
Data	8.63x
Word count	1.25x
Ref. count	3.31x

Table: Improvement with Panthera in the total amount of time required to execute algorithms/requirements (compared with status quo). **An improvement of "3x" implies that the mean time was 3 times lower with Panthera compared to the status quo.**

Type	Standard deviation contraction
Metadata	27.38x
Data	11.14x
Word count	1.02x
Ref. count	22.91x

Table: Reduction obtained with Panthera in the standard deviation of running times, i.e. a **contraction of 9x implies that the standard deviation was reduced by a factor of 9 with Panthera compared to the status quo.**

Conclusion

The results validate the hypothesis presented.

- Panthera was successfully developed as a "drop-in" solution
- It was able to reduce latency in comparison to a standard Hadoop installation.
- A scheduler was also implemented which uses cache information to schedule Hadoop jobs.
- Metadata latency was **decreased by a factor of 8.13**
- Data latency was **decreased by a factor of 8.63**
- The running of time of WordCount (an algorithm with very few file accesses) was **improved by 25.1%** with the code left unmodified.
- The running time of RefCount was **improved by a factor of 3.31** with its code also unmodified.

Thus, Panthera can significantly improve existing distributed systems without any code modifications and also enable developers to implement algorithms that were previously impractical (such as RefCount).

Applications

Panthera can be applied in a vast range of disciplines since the Hadoop distributed system is in widespread use. A few examples follow:

- **Machine Learning**
  - Panthera was tested with Apache Mahout, which is a library of machine learning algorithms for Hadoop
- **Bioinformatics**
  - BioDoop: a bioinformatics framework for Hadoop.
  - Distributed sequence alignment
- **Predictive Analytics**
  - Hadoop is heavily used for predictive analytics. Panthera can be used to reduce the running time of queries and schedule queries for maximal efficiency, reducing total turnaround time.
- **Image Processing**
- **Numerical/Scientific Computing**
  - Apache Hama is a matrix computational package for Hadoop. Applications using it can be made more efficient without any modifications to their codebases due to Panthera's "drop-in" nature,.

Due to the drastic decreases in latency and running times that Panthera provides (in addition to allowing the implementation of previously impractical algorithms), **Panthera can truly transform computational work across many fields of study.**

Future Work

- **All code developed during this project will be open sourced to further development in distributed computing.**
- Expose an API to allow external caching requests (almost complete)
- Currently working on testing a **bioinformatics toolkit with Panthera**; initial results are very promising and show significant reduction in total running time.
- Implement **predictive prefetching** algorithms on Panthera
- Consider the effects of different cache replacement algorithms