

Panthera: A Study of Caching in Distributed Computing

Dhaivat Pandya
Appleton North High School

Abstract

The adoption of distributed computing systems has grown massively in the past few years. In particular, Apache Hadoop, which allows developers to create applications that run on a cluster of computers, is currently used throughout academia and industry. However, the Hadoop File System fails to effectively utilize memory and local storage in order to reduce waiting time (i.e. latency). In this project, a system, named Panthera, was developed, which creates a caching system for Hadoop metadata and data. Panthera caches both information about files and the file contents, thereby reducing waiting time in downloading and accessing the files and related information. It runs with an unmodified version of Hadoop, meaning that it can integrate easily into existing architecture. Panthera was tested using both single node and multi- node setups. The results showed an 90.8% decrease in waiting time with the Panthera metadata cache, and a 95.7% decrease with the Panthera data cache. Such drastic decreases in latency can greatly increase the efficiency of existing Hadoop applications and also allow the creation of algorithms that were previously not feasible. Panthera has numerous applications in fields ranging from market research to bioinformatics. All code associated with the project will be open sourced to further development in the area.

1 Introduction

Distributed computing systems are currently employed across academia and industry, with applications varying from market research for department stores to bioinformatics. In particular, the Hadoop distributed system [13] has seen tremendous growth. Hadoop is an open source version of the revolutionary MapReduce system developed at Google. With it, developers can easily take advantage of large, multi-node clusters to solve computational problems.

Current Hadoop deployments exist in the thousands of nodes at companies such as Cloudera, Facebook, Yahoo, etc. However, especially in larger networks, file access latency can be a significant issue. The Hadoop File System currently does not use local (i.e. on the client) random access memory (RAM) to improve latency.

Latency in distributed systems can have significant effects, and a reduction of the same comes with tremendous benefits. As the RAMCloud project has outlined [11], low latency can greatly extend the applications of distributed computing systems. For example, tree traversal algorithms are largely impractical in Hadoop, however, with lower latency, such algorithms may become useful and applicable.

In this paper, we discuss *Panthera*, a cache layer for Hadoop. In Section 2, we discuss the specific challenges associated with such a project. Section 3 details the architecture differences between a standard Hadoop installation and one with *Panthera*. Sections 4 and 5 detail the differences between metadata and data caching within *Panthera*. Results and conclusions are covered in Sections 6, 7, 8, 9, concluding with a roadmap for future development in Section 10.

2 Constraints

There are several systems built on Hadoop that are in widespread use [3, 17, 10]. The Hadoop project is also supported by large corporations with a consistent release cycle. Thus, for *Panthera* to be practical, it must operate independently of the existing Hadoop codebase, i.e. as a layer, rather than a modification. This implies that *Panthera* must be implemented without a single change in the existing Hadoop codebase, i.e. as a "drop-in" solution.

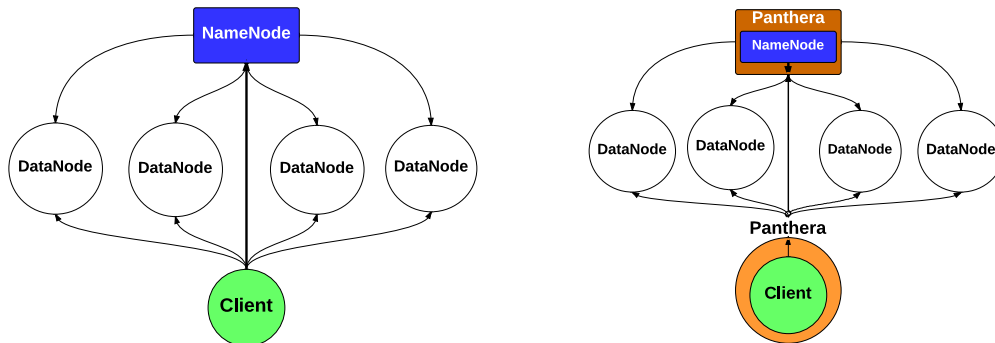
Additionally, for non-cache related requests, *Panthera* must add insignificant latency. Finally, data and metadata request latency should be significantly with *Panthera* in comparison to a standard Hadoop installation.

3 Architecture

Figure 1 outlines the architecture differences between a standard Hadoop File System architecture and an installation with *Panthera*.

Within the standard conditions, the *NameNode* serves as a metadata server, handling all requests from clients relating to metadata (e.g. directory listing, file sizes, block locations, etc.). *DataNode(s)* hold data and serve it at request from the client. A more complete description of the Hadoop File System may

Figure 1: Standard Hadoop Architecture (left), Panthera-Hadoop Architecture (right)



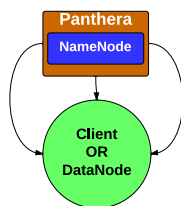
be found in [13].

Panthera operates by intercepting requests and responses at the NameNode and the client. Both nodes have a running instance of *Panthera*. Every request issued from the client is first reviewed by Panthera, and if possible, answered immediately from the cache. The *Panthera* instance on the NameNode constantly monitors the filesystem for changes that need to be reflected in the cache (e.g. if a file is deleted, the cache on the client should have a copy of it).

Panthera, since it caches both data and metadata, can be considered two separate subsystems, which are now discussed.

4 Metadata caching

Figure 2: Panthera Metadata System



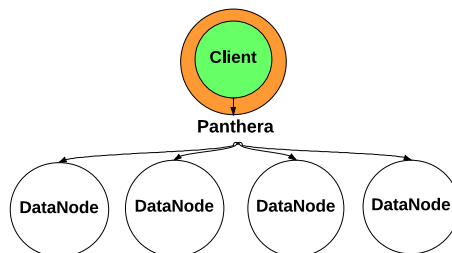
The Panthera metadata system runs on the NameNode and communicates with the Panthera instance on the client. The challenge on the metadata portion consist of reducing computational latency to memory latency. For example, for some intensive methods (e.g. a recursive directory listing), it may be worthwhile to cache a response in order to prevent computing the same list again.

However, the problem lies in identifying such methods. Obviously, there are some methods for which caching is not worthwhile since their execution time is lower or comparable to memory latency. Within Panthera, six different Hadoop query methods were identified as medium to high latency. Panthera identifies packets that contain these methods and subsequently caches them.

Currently, a standard LRU cache is used for each method. Testing of different caching methods and predictive prefetching algorithms is under way.

5 Data caching

Figure 3: Panthera Data System



Panthera’s data caching system runs on the client. The architecture is similar to that of the metadata caching system. Every request that the client makes to the DataNodes is first processed by Panthera and responded to from the cache if possible.

The challenges with data caching differ significantly from those in meta-data caching. Here, the goal is to convert network and hard drive read latency into memory latency. As [11, 13] note, memory latency can be extremely low in modern systems, especially in comparison to network and hard drive latencies. Thus, in some respects, the goal is ”easier” to attain in the data cache. However, for smaller files, it may be difficult to obtain latency improvement.

6 Testing Methodology

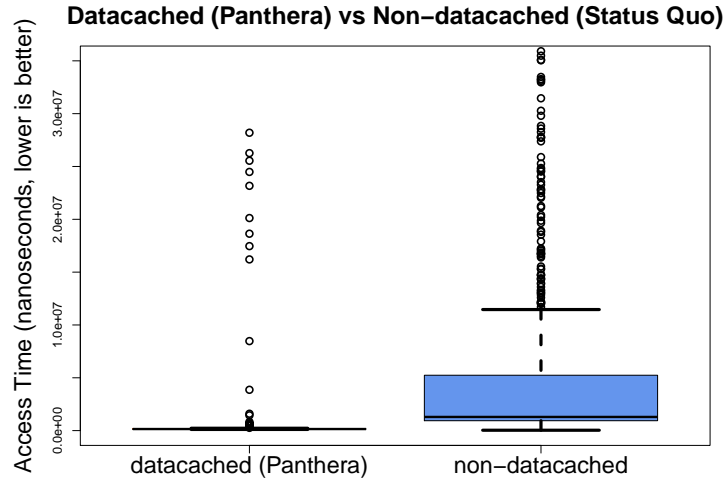
In testing Panthera, very simple strategies were adopted. In order to test the metadata cache, a directory listing query was repeatedly run on a directory with 100 files in it, and the latency times for responding are measured, with Panthera and without. The reasoning for picking such a test lies in the fact that a directory listing in Hadoop involves all six of the cached methods as well as a few non-cached methods. Thus, the test provides a good estimate of the efficacy of the cache.

Latency times for the data cache were obtained by repeatedly querying for a one megabyte large file. The test encapsulates the most common use case for Panthera and the Hadoop File System.

Tests for the metadata cache were conducted on both single node and multinode setups, whereas the datanode cache testing was run on multinode setups.

7 Results: Data caching

Figure 4: Panthera data cache latencies

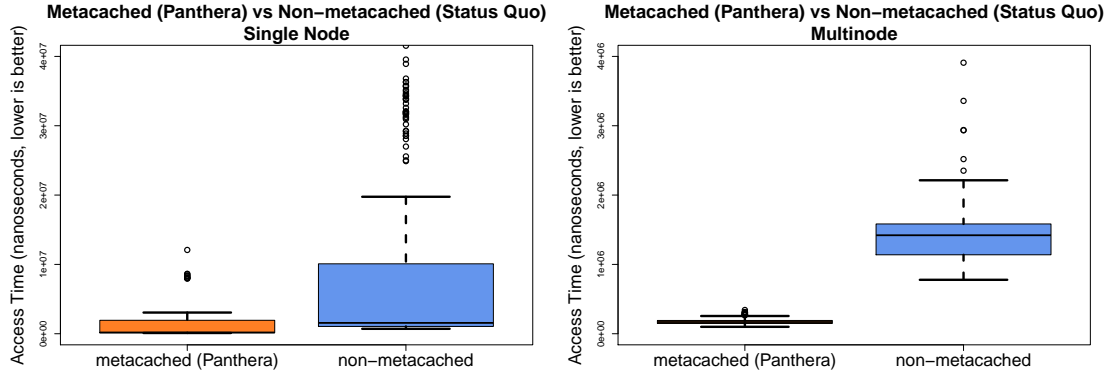


The Panthera data cache showed a 95.7% decrease in the latency in comparison to a standard deviation non-datacached installation. In addition, the standard of deviation was lower by a factor of 27. For any deployed project, maximum levels of latency must be considered when designing a system. Thus,

the *status quo* of the non-cached version severely limits possibilities within Hadoop.

8 Results: Metadata caching

Figure 5: Panthera metadata cache latencies



Metadata caching has similarly incredible results. *Panthera* showed a 9 fold or 90.8% improvement in the total latency as measured on the client side in a multinode setup. Additionally, the standard deviation is 24 fold lower in *Panthera*.

9 Conclusion

Panthera fulfils the constraints mentioned at the outset. It is a layer on Hadoop rather than a main repository code modification. Considering the latency measurements, it is clear that it has also met the requirements in terms of benefit. *Panthera* was able to reduce data access latency by 95.7% and reduce basic metadata access latency by 90.8%. Additionally, *Panthera* greatly reduced the variation in the access latency, greatly aiding in the pre-development plans and extending the capabilities of distributed computing systems such as Hadoop.

10 Future work

Panthera can be extremely useful in a wide variety of applications, specifically in those which require large datasets in separate files (e.g. unstructured

data). In particular, I will be concentrating on bioinformatics and integrating Panthera’s caching with algorithms used in the area.

I am also working on a cache-based scheduler for Hadoop. Essentially, it will use information about what files are available in various clients and arrange Hadoop tasks such that the use of the cache is maximized.

Finally, all code associated with this paper and *Panthera* will be open sourced in order to further development of Hadoop and distributed computing.

References

- [1] Dhruba Borthakur, Jonathan Gray, Joydeep Sen Sarma, Kannan Muthukkaruppan, Nicolas Spiegelberg, Hairong Kuang, Karthik Ranganathan, Dmytro Molkov, Aravind Menon, Samuel Rash, et al. Apache hadoop goes realtime at facebook. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 1071–1080. ACM, 2011.
- [2] Sudipto Das, Yannis Sismanis, Kevin S Beyer, Rainer Gemulla, Peter J Haas, and John McPherson. Ricardo: integrating r and hadoop. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 987–998. ACM, 2010.
- [3] Lars George. *HBase: the definitive guide*. O’Reilly Media, Inc., 2011.
- [4] Karthik Kambatla, Abhinav Pathak, and Himabindu Pucha. Towards optimizing hadoop provisioning in the cloud. In *Proc. of the First Workshop on Hot Topics in Cloud Computing*, page 118, 2009.
- [5] David Karger, Eric Lehman, Tom Leighton, Rina Panigrahy, Matthew Levine, and Daniel Lewin. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 654–663. ACM, 1997.
- [6] Kamal Kc and Kemafor Anyanwu. Scheduling hadoop jobs to meet deadlines. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 388–392. IEEE, 2010.
- [7] Jacob Leverich and Christos Kozyrakis. On the energy (in) efficiency of hadoop clusters. *ACM SIGOPS Operating Systems Review*, 44(1):61–65, 2010.

- [8] Xuhui Liu, Jizhong Han, Yunqin Zhong, Chengde Han, and Xubin He. Implementing webgis on hadoop: A case study of improving small file i/o performance on hdfs. In *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*, pages 1–8. IEEE, 2009.
- [9] Michael N Nelson, Brent B Welch, and John K Ousterhout. Caching in the sprite network file system. *ACM Transactions on Computer Systems (TOCS)*, 6(1):134–154, 1988.
- [10] Christopher Olston, Benjamin Reed, Utkarsh Srivastava, Ravi Kumar, and Andrew Tomkins. Pig latin: a not-so-foreign language for data processing. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1099–1110. ACM, 2008.
- [11] John Ousterhout, Parag Agrawal, David Erickson, Christos Kozyrakis, Jacob Leverich, David Mazières, Subhasish Mitra, Aravind Narayanan, Guru Parulkar, Mendel Rosenblum, et al. The case for ramclouds: scalable high-performance storage entirely in dram. *ACM SIGOPS Operating Systems Review*, 43(4):92–105, 2010.
- [12] Antony Rowstron and Peter Druschel. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. In *ACM SIGOPS Operating Systems Review*, volume 35, pages 188–201. ACM, 2001.
- [13] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pages 1–10. IEEE, 2010.
- [14] An-I Wang, Peter L Reiher, Gerald J Popek, and Geoffrey H Kuenning. Conquest: Better performance through a disk/persistent-ram hybrid file system. In *USENIX Annual Technical Conference, General Track*, pages 15–28, 2002.
- [15] Jia Wang. A survey of web caching schemes for the internet. *ACM SIGCOMM Computer Communication Review*, 29(5):36–46, 1999.
- [16] Jianwu Wang, Daniel Crawl, and Ilkay Altintas. Kepler+ hadoop: a general architecture facilitating data-intensive applications in scientific workflow systems. In *Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science*, page 12. ACM, 2009.

- [17] Chen Zhang and Hans De Sterck. Cloudbatch: A batch job queuing system on clouds with hadoop and hbase. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 368–375. IEEE, 2010.