A Distributed Cache for Hadoop Distributed File System in Real-time Cloud Services

Authors: Jing Zhang, Gongqing Wu, Xuegang Hu, Xindong Wu

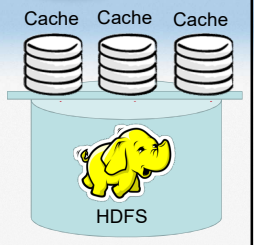
Presenters: Cindy Wang, Minh Vu, Travis Arriola

Talk Outline

- Hadoop Distributed File System (HDFS)
  - What is HDFS?
  - Why is HDFS important?
  - What are problems related to HDFS?
  - What are some proposed solutions?
  - What are the findings from the proposed solutions?

Paper overview

- Aim : To investigate the access performance when an additional cache system is added on top of the HDFS
- Problem : HDFS stores massive data but lack real-time file access



Introduction

Step 1) Analyze preconditions / environments in the cloud

Step 2) Look at some related work for suggesting how to improve the access performance in HDFS.

Step 3) Design a novel distributed cache system (HDCache) using shared memory as an infrastructure.

Step 4) Test the access performance for different file sizes and application threads.

Background / Related Work

Modify HDFS I/O Features

How to Deal with small files?

Related Work: Modify HDFS I/O Features

- Jiang et al. (2010) “The optimization of HDFS based on small files”
- Mackey et al., (2009) “Improving metadata management for small files in HDFS”

Background / Related Work

Modify HDFS I/O Features

Combine small files into a larger file

Related Work: Combine small files into a larger file

- Liu et al. (2009) “Implementing WebGIS on Hadoop: A case study of improving small file I/O performance on HDFS”

Background / Related Work

Modify HDFS I/O Features

Combine small files into a larger file

Key-Value distributed database

Related Work: Key-Value distributed database

- Chang et al., (2008) “Bigtable: A Distributed Storage System for Structured Data”

Background / Related Work

Modify HDFS I/O Features

Combine small files into a larger file

Key-Value distributed database

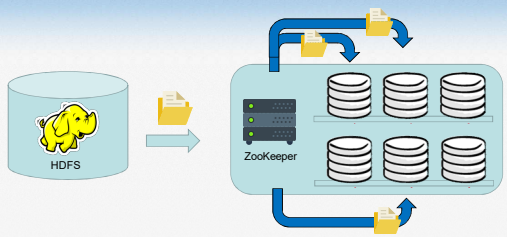
Summary of HDCache Design Implementations

On-the-top Method V.S., Build-in Method

Network I/O V.S., Disk I/O

Layered Data Accessing Model

Summary of HDCache Architecture

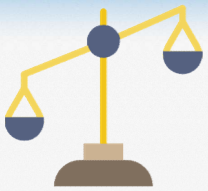


Key contributions

- On-the-top HDCache
- 3 replicas per file
- Shared memory as an infrastructure
- Solve small files problem

Author's Evaluation #1 YCSB (Yahoo Cloud Serving Benchmark)

- Result 1) small files processing perform equally as the benchmark
- Result 2) large files processing perform better than the benchmark
- Result 3) HDCache has a fast access performance in highly concurrent environment



Author's Evaluation #2 Hit Ratio

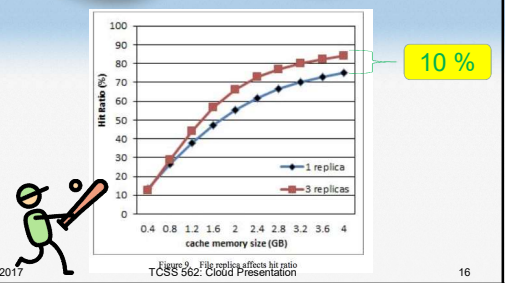


Figure 9. File replica affects hit ratio

Conclusions

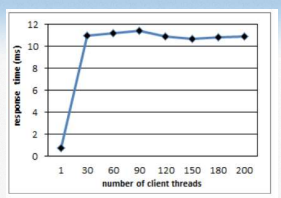


Figure 7. cache_fopen operation response time on different number of concurrent threads

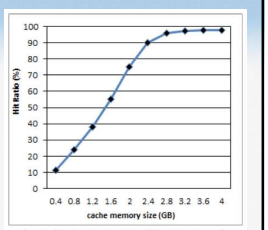


Figure 8. Hit ratio of a single node in different cache memory size

Critique: Strengths

- Fast file access performance
- Fault Tolerance
- Scalable

Critique: Weaknesses

- Throughput & Concurrency

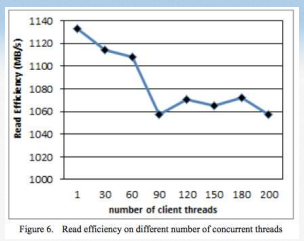


Figure 6. Read efficiency on different number of concurrent threads

Critique: Evaluation

- 1500+ paper downloads
- Cited by 35+ ACM/IEEE papers

Future Work

- The HDCache system is based on the classic write-once-read many data access model in the cloud.
- However, many real-time services have more complex and dynamic data access models.