Dipayan Dev, Ripon Patgiri, 2016. Dr. Hadoop : an infinite scalable metadata management for Hadoop — How the baby elephant becomes immortal. *Frontiers of Information Technology & Electronic Engineering*, **17**(1):15-31. http://dx.doi.org/10.1631/FITEE.1500015

Dr. Hadoop : an infinite scalable metadata management for Hadoop— How the baby elephant becomes immortal

Key words: Hadoop, NameNode, Metadata, Locality-preserving hashing, Consistent hashing

Contact: Dipayan Dev
E-mail: dev.dipayan16@gmail.com

ORCID: http://orcid.org/0000-0002-0285-9551

Introduction

The goal of this work is to model an infinite scalable NameNode cluster for Hadoop as well as to maintain metadata locality and a balanced workload among Metadata Servers. The main problem of designing MDS cluster is how to partition metadata efficiently among them to provide high-performance metadata services.

Motivation

- Hadoop suffers from **Single Point of Failure (SPOF).**
- 90% of the data that exists today were created over the last two years.
- Compared to the overall data space, the size of metadata is relatively small, and it is typically 0.1% to 1% of data Space. For PB scale file systems , **it goes up to 1-10 TB of metadata.**
- Although the amount of metadata is small, **metadata operations** account for **over 60 percent** of the operations in typical workloads.
- These requirements indicate that centralized metadata processing that relies on a **single metadata server is impractical** and it is necessary to use **decentralized metadata processing** that utilizes a group of **metadata servers** (MDS).

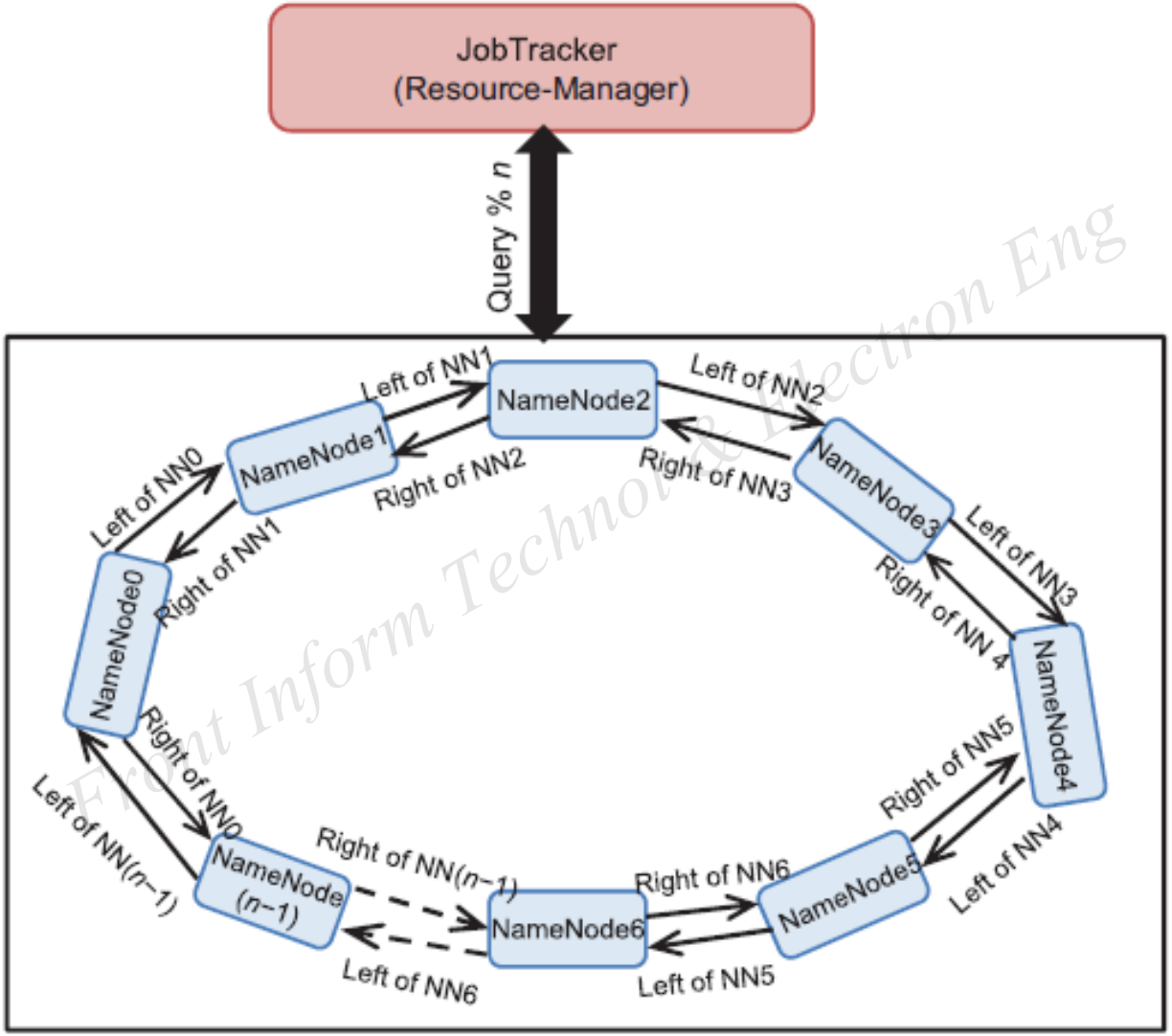
Paper contribution

- We design a scalable, dynamic and circular metadata management model, named ***Dynamic Metadata Management Splitting*** (***DCMS***), to store many metadata and support fast metadata lookups in an ultra-large-scale file system with multiple Metadata Server (MDS).

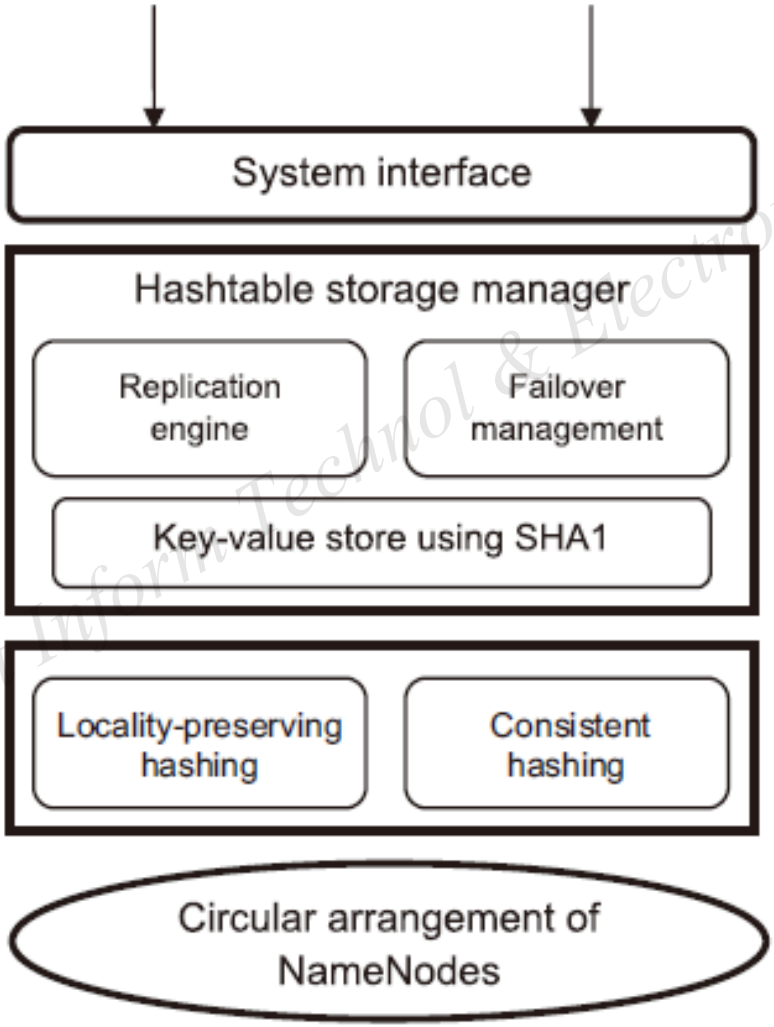
- We preserve metadata locality and workload among MDS using **Consistent Hashing** as well as **Locality-preserving Hashing** (LpH). It also keeps replicated metadata for excellent reliability.

- We examine the proposed DCMS structure in **Dr. Hadoop** through extensive trace-driven simulations and experiments on a prototype implementation in Linux.

DCMS: dynamic circular metadata splitting



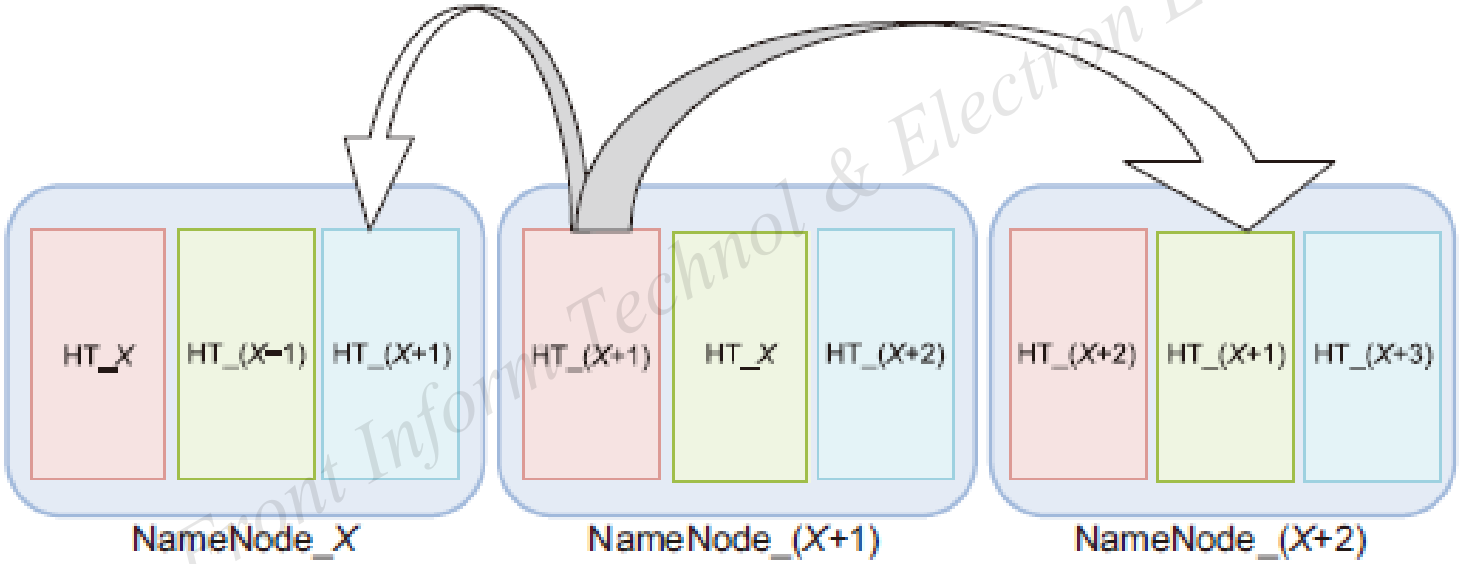
Skeleton of MetaData Server in Dr. Hadoop



Replication management

- Our back-of-the-envelope calculation suggests that a **replication factor of 3** is the optimum value for the lowest failure rate.
- Dr. Hadoop uses **Distributed Hash Table (DHT) - One-copy consistency**.
- Each NameNode has its two **hot-standby MDS**.
- NameNode_*X* will contain the Hashtable of NameNode_*X*, NameNode_(*X*-1) and NameNode_(*X*+1).
- The Hashtable of NameNode_*X* is denoted by HT_*X*.
- So, NameNode_*X* will contain three Hashtable, viz. HT_*X*, HT_(*X*-1) and HT_(*X*+1).

Contd..



Locality-preserving Hashing (LpH)

- LpH is required to have the lowest number of RPC for every R/W.
- To attain near-optimal locality, the entire directory tree nested under a point has to reside on the same NameNode.
- Dr. Hadoop makes the hash of the parent path so that, the metadata for all the files in a given directory is present at the same NameNode.

Consistent Hashing

- The following results are proven in the papers that introduced consistent hashing (Karger *et al.*, 1997; Lewin, 1998) that are foundations of our architecture.

THEOREM 1. For any set of N nodes and K keys, with high probability:
1. Each node is responsible for at most$(1+\varepsilon)K/N$ keys.
2. When an $(N+1)$th node joins or leaves the network, responsibility for $O(K/N)$ keys changes hands (and only to or from the joining or leaving node)

- SHA1() is applied to each blocks to generate unique 20 Bytes identifier which is stored in the hashtable as the key.

"/home/dip/folder/fold/fold1/file.txt" are respectively,
SHA1(home/dip/folder/fold/ fold1/file.txt, 0) and
SHA1 (home/ dip/folder/fold/fold1/file.txt, 2)

Design analysis

- Total number of machines: *n*
- In traditional Hadoop, there is exactly 2 servers and the remaining *n*−2 machines are DataNodes that store the actual data.
- In Dr. Hadoop, we have *m* NameNodes that distribute the metadata *r* times. Hadoop-DR can thus survive the crash of *r*−1 NameNodes.

Metric	Traditional Hadoop	Dr. Hadoop
Meta-server failures that can be tolerated	0	$r-1$
RPCs required for a read	1	1
RPCs required for a write	1	r
Metadata storage per NameNode	X	$(X/r)m$

Experiment on real data traces

Traces	No. of Files	Data Size	Metadata Extracted
Yahoo!	8,139,723	256.8 GB	596.4 MB
Microsoft	7,725,928	223.7 GB	416.2 MB

**** Metadata crawler is applied to the datasets that recursively extract file/directory metadata using stat() function.**

Experimental setup

- 20 DataNodes, one ResourceManager and 10 NameNodes.
- Each node is running at 3.10 GHz clock speed and with 4 GB RAM and a gigabit Ethernet NIC.
- All nodes are configured with 500 GB Hard Disk. Ubuntu[21] 14.04 is used as our operating system.
- Hadoop 2.5.0 version is configured for comparison between Hadoop and Dr. Hadoop, keeping the HDFS block-size as 512MB.

Conclusions

- We built a an infinite scalable metadata management, DCMS, and implemented it on Dr. Hadoop which is a modification of traditional Hadoop.
- DCMS shows infinite scalability with the minimum number of NameNodes.
- The system has removed the SPOF to a great extent with an availability of 99.99%.
- Various experimental works are carried out on read time traces to verify our model.