Accelerating Reverse Lookup in HDFS using Replicated BlockReport

Junyao Zhang, Jun Wang

Departments of EECS, University of Central Florida, 4000 Univ. Blvd., Orlando FL 32816

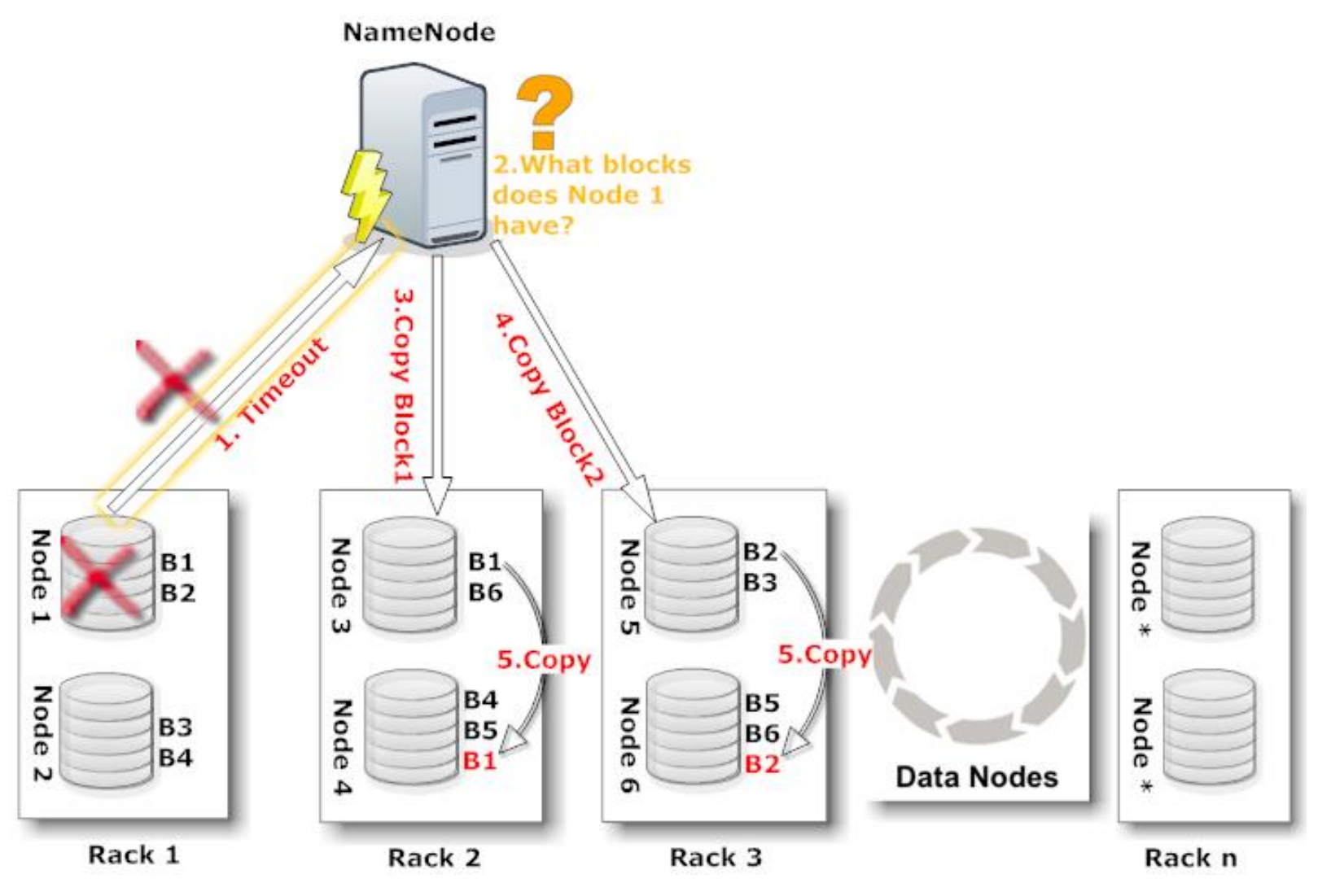
Introduction

In today's a peta-scale ware-house clusters, node failure has become norm instead of a rare situation. Despite the state-of-art solutions such as multi-way replication, the first question popped up is that after node failure, how does the system retrieve the information of the missing blocks quickly and efficiently. We define this problem as Reverse Lookup (RL). In this work, we study the RL problem in Hadoop Distributed File System(HDFS), which is one of the most popular distributed storage system for hosting big data applications. We observe that current RL solution for HDFS is slow and inefficient---the namenode (metadata server) has to browse the whole content of the metadata inode tree to retrieve the metadata for the missing blocks. Especially with the rapid expanding of storage size, this traversal method could become a potential hazard for recovery. In this work, we explore the possibility of utilizing the Blockreport for fast and efficient reverse lookup. The key insight is that, the block list is prestored in the blockreport, which is current used to build upt the block distribution map for namenode. Thus by keeping the availability of blockreport, the list of missing blocks can be easily retrieved.

Background

In HDFS, when the namenode detects this condition of heartbeat absence, it marks that datanode as dead and stops sending IO requests to that datanode. Meanwhile, the namenode browses its whole metadata inode tree to find the missing blocks resulted by the this datanode death and conduct "minus one" operation on their replication numbers. When the replication number for a chunk falls below than the threshold, the namenode will initiate a re-replication process for this block.

- RL Problem: retrieving the block list for the failed node
- Existing Solution: Browsing the whole metadata tree; time complexity is O(N), where N is the total number of metadata inode entries. Linear increasing.



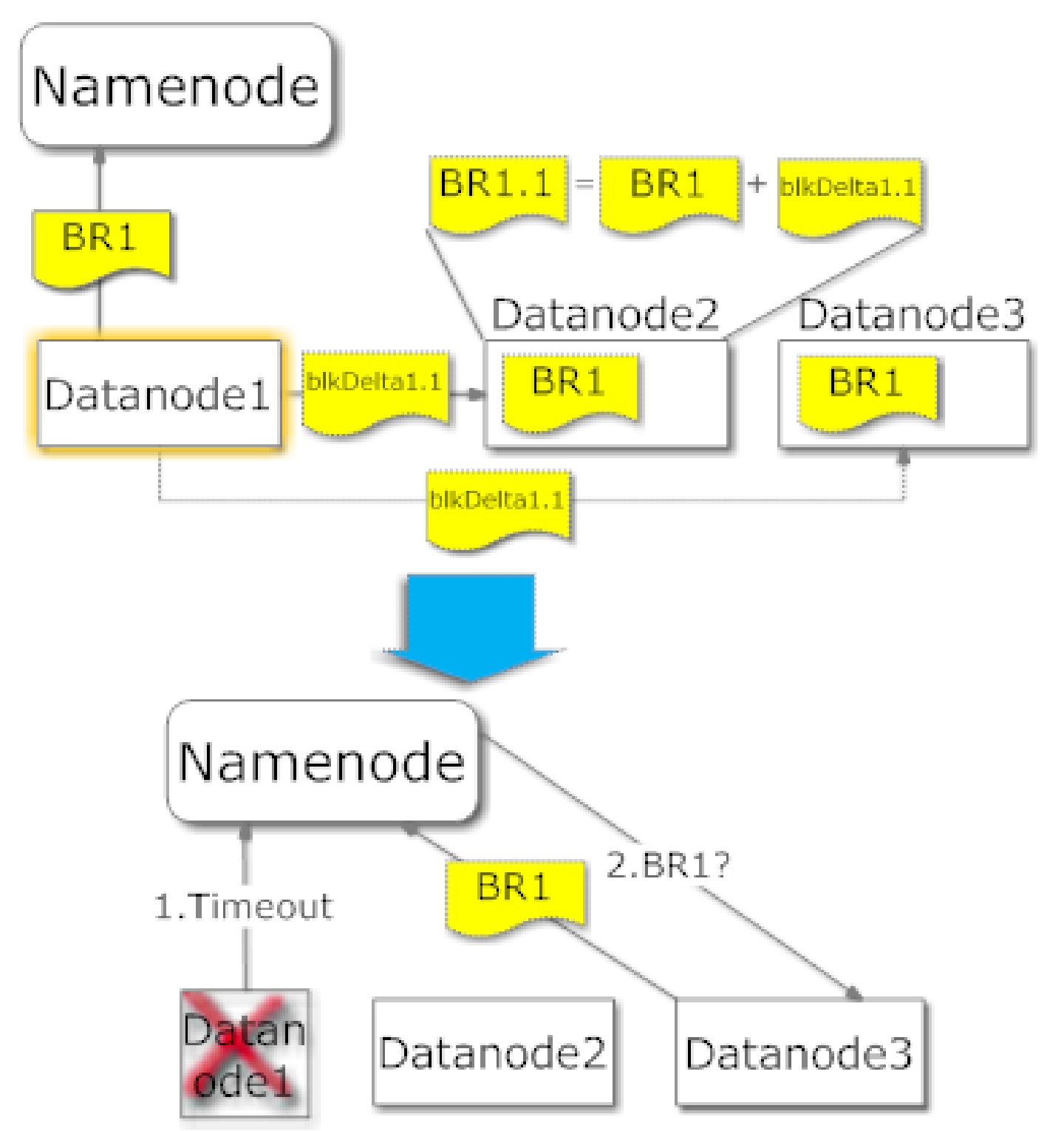
The steps of reverse lookup in HDFS; Reverse lookup is the first thing to do after the node failure.

Methods

The HDFS maintains its namespace in memory and the blockreport is the way for HDFS to rapidly generate a overview of the block distribution over the file system. Blockreport is created and sent to namenode whenever a node starts up and then periodically checks in with namenode for the block distribution map. Namenode compares the blockreport information with its own namespace and conduct necessary operations such as remove, add etc.

Our solution:

- Upon initialization, beside sending the blockreport to namenode; replicate two copies of blockreport on neighbor nodes, which is the smallest two comparing the hash value. Randomly choose one if collision occurs. This allows fast lookup (Time complexity O(1)).
- When getting blockreport with new time-stamp, compare with the existing one and generate a blockreport diff--blkDelta
- Whenever sending a blockreport is sent to namenode, send blkDelta to the other two copies.
- Namenode acquires blockreport following the hash table lookup when heartbeat timeout.



The Overall Design of replicating blockreport and using it for fast reverse lookup

Table 1 Blockreport storage overhead per node with different block size and data size per node

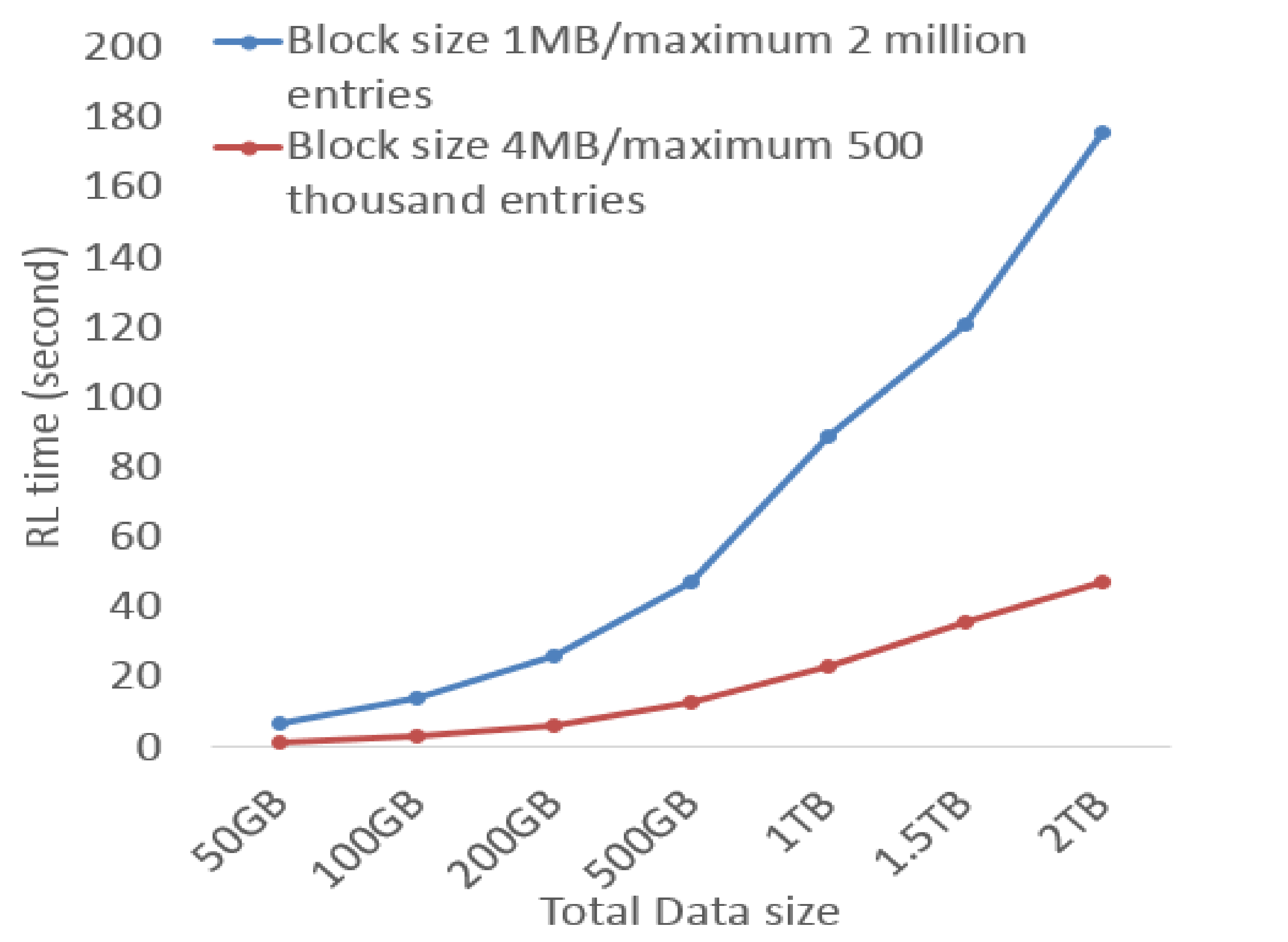
	500GB	1TB	2TB
4MB	4MB per node	8MB per node	16MB per node
16MB	1MB per node	2MB per node	4MB per node
64MB	250KB per node	512KB per node	1MB per node

Storage and Network Overhead

The size of a blockreport is only related to the data size located on a particular node. It is related to

Storage overhead: each block is kept in 2 long types (64*2=128 bits) in the blockreport array list. In a HDFS with n nodes and B_i blocks in the i-th node, storage overhead is $256\sum_{i=1}^{n} B_i$ bits.

Network transfer overhead: HDFS default update blockreport one hour a time; the first time incurs $256\sum_{i=1}^{n} B_i$ bits transfer; after that only blkDelta*2



Reverse Lookup Time at scale

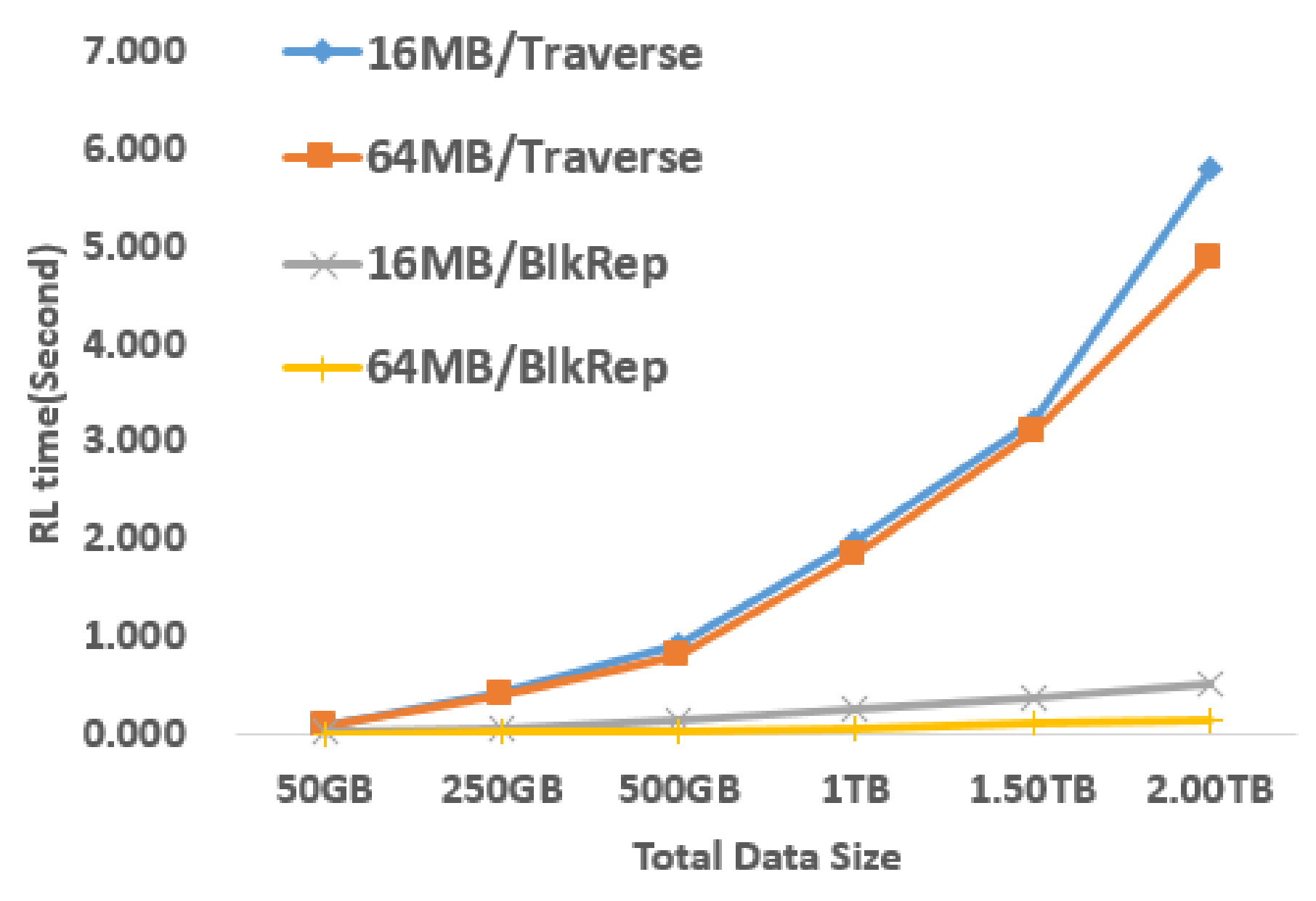
According to [1], a capacity of 10PB HDFS will have 100 million files and 200 million blocks. As shown in this Figure, browsing two million entries requires more than three minutes, and it will increase linearly with the growth of the data size. To examine the reverse lookup time at scale, we manually twist the HDFS default block size to 1MB. In this way, a large number of blocks will be generated within a relatively small data size. The data is retrieved on Marmot 64 nodes HDFS[2]. While 25 seconds are needed for browsing 50 thousand inode entries, more than three minutes are consumed to browse 200 million metadata entries,

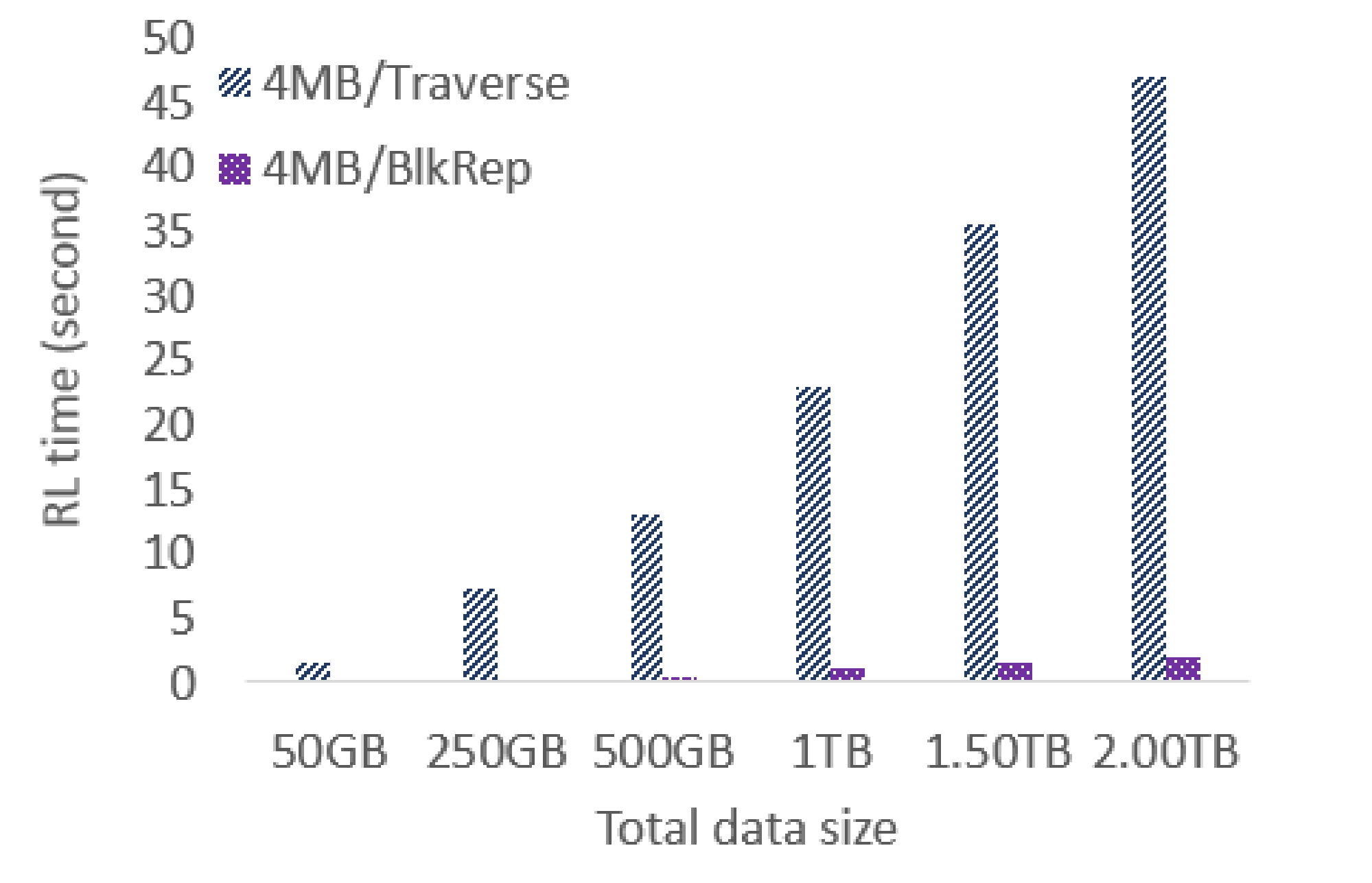
Reverse Lookup Time Results

HDFS built on 32 nodes of PRObE Marmot cluster; 2TB HDD; 1Gbps shared network.

Traversal time is large and it increase with the growth of total data size, while RL time using replicated blockreport size finishes within 0.5 seconds.

Average speedup of RL using blockreplication is 27 time for 4MB, 8 times for 16MB and 30 times for 64MB.





Future work

1. Study the performance impact of replicated blockreport and the recovery time.
2. Seeking more optimization opportunities to speedup reverse lookup time such as using SDN to prioritize the network packages.

Reference

1. HDFS Scalability: The Limits to Growth, The Magazine of USENIX, Vol. 35, No. 2. (April 2010), pp. 6-16 by Konstantin V. Shvachko
2. An Integrated Experimental Environment for Distributed Systems and Networks, Proceedings of the {USENIX} Symposium on Operating System Design and Implementation (OSDI), Dec 2002, B. White and J. Lepreau, L. Stoller, R. Ricci and S. Guruprasad, M. Newbold, M. Hibler, Chad Barb, Abhijeet Joglekar

The work uses some of PRObE clusters which are supported in part by the National Science Foundation under awards CNS-1042537 and CNS-1042543 (PRObE).