A Distributed and Cooperative NameNode Cluster for a Highly-Available Hadoop Distributed File System

Yonghwan Kim† Tadashi Araragi‡ Junya Nakamura† Toshimitsu Masuzawa†

†Graduate School of Information and Technology, Osaka University, Osaka, Japan
‡Communication Science Laboratories, Nippon Telegraph and Telephone Corp., Kyoto, Japan

1 Introduction and Related Works

Recently, Hadoop attracts much attention of engineers and researchers as an emerging and effective framework for *Big Data*. *HDFS(Hadoop Distributed File System)* can manage huge amount of data with guaranteeing high performance and reliability with only commodity hardware.

However, HDFS requires a single master node, called *NameNode*, to manage the entire namespace (or all the i-nodes) of a file system. This causes *SPOF* (Single Point Of Failure) problem [1] because the file system becomes inaccessible when the *NameNode* fails. This also causes a *bottleneck of efficiency* since all the access requests to the file system have to contact the *NameNode*. Hadoop 2.0 resolves the SPOF problem by introducing manual failover based on two *NameNodes*, *Active* and *Standby*. However, it still has the efficiency bottleneck problem since all the access requests have to contact the *Active* in ordinary executions. It may also lose an advantage of using commodity hardware since the two *NameNodes* have to share a highly-reliable sophisticated storage.

In this paper, we propose a new HDFS architecture to resolve all the problems mentioned above. The proposed architecture has the following features and advantages.

1. Multiple *NameNodes* (not restricted to two) can be utilized to improve *availability*. The entire namespace of a file system is partitioned into several *fragments*, and *replicas* of each fragment are dispersed among the *NameNodes*. When each fragment has k replicas, the file system can tolerate up to ($\lfloor \frac{k}{2} - 1 \rfloor$) faulty *NameNodes*.
2. Multiple *NameNodes* can be utilized to improve *performance*. The performance bottleneck caused by a single *NameNode* can be circumvented by assigning different *NameNodes* to different fragments as the *primary* ones (or the entry points).
3. *The highly-reliable storage* shared by the *NameNodes* is removed by introducing message-based consistency mechanism among the *NameNodes*. The architecture requires only *commodity hardware*.

2 Distributed NameNode Cluster

2.1 Namespace Partitioning

Figure 1 represents overview of HDFSs in Hadoop 2.0 and our proposed architecture. The whole namespace is completely replicated in Hadoop 2.0, however, the namespace is partitioned into several fragments, NS_i, in our architecture. Replicas of each fragment are dispersed among the *NameNodes*, one of which works as a primary one and the others as backups. Namespace's partitioning rules and states can be changed dynamically for load-balancing, and recorded to ZooKeeper, as a specific hash table, named *NSTable*.

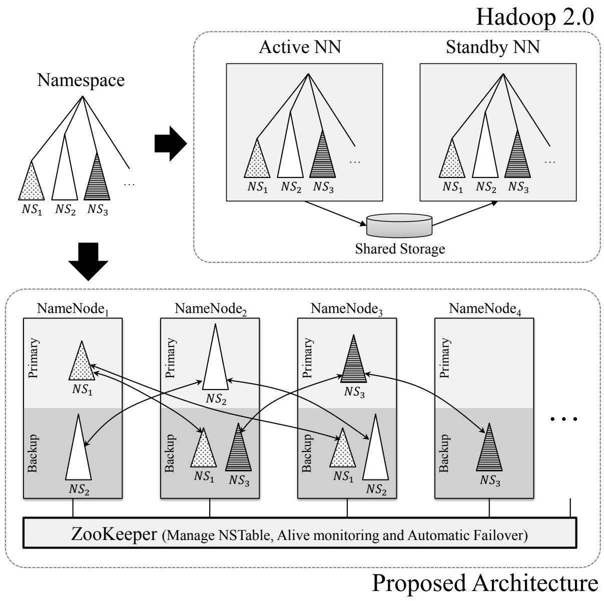

Figure 1: Namespace Partitioning

2.2 Cooperation among NameNodes

Consistency Mechanism: Replicas of each fragment dispersed among the NameNodes should keep consistency and this is achieved by the *majority-based* mechanism. Only the primary *NameNode* can receive the requests from clients. On receipt of the request, it must broadcast *sync* messages to the backup *NameNodes*. When it receives $\lceil \frac{k}{2} \rceil$ *ack* messages from backup *NameNodes*, it confirms the process and broadcasts *update* messages to backup *NameNodes*.

Automatic Failover: If ZooKeeper suspects a current primary *NameNode* fails, ZooKeeper elects a new primary *NameNode* and notifies it of the backup *NameNodes*. Each backup *NameNode* reports its fragment's version and records it to ZooKeeper. When a majority of *NameNodes* reports its fragment's version, the latest NS_i of them is chosen as the new common NS_i among the *NameNodes*. The consistency mechanism guarantees the consistency of this new NS_i's version through the automatic failover. Even if a newly elected primary *NameNode* fails during the automatic failover, the system can still guarantee consistency.

References

[1] K. Shvachko, "Warm HA NameNode going Hot," Apache Hadoop Issues, HDFS-2064, 2011.