Hadoop MapReduce on Gfarm File System

Shunsuke Mikami
College of Information Sciences, University of Tsukuba
mikami@hpcs.cs.tsukuba.ac.jp

Kazuki Ohta
Department of Information Science, University of Tokyo
kzk@il.is.s.u-tokyo.ac.jp

Osamu Tatebe
Department of Computer Science, University of Tsukuba
tatebe@cs.tsukuba.ac.jp

Abstract

MapReduce is a promising parallel programming model for processing large data sets. This paper discusses the design and implementation of Hadoop-Gfarm plugin to enable access to the Gfarm file system by Hadoop MapReduce applications. The performance evaluation shows it has comparable performance to the Hadoop native HDFS. The Gfarm file system has advantage since it supports not only MapReduce applications but also POSIX and MPI-IO applications.

1. Introduction

MapReduce [1] is a parallel programming model for processing large data sets. Hadoop [2] is an up-and-coming open source implementation of MapReduce, which utilizes the Hadoop Distributed File System (HDFS) [3] to store input and output data. HDFS does not support the POSIX semantics since it is not required by MapReduce workloads. It does not support file modification after once closed and writes to a single file by multiple clients. MapReduce workloads always create new files and do not change contents of existing files. This means the storage space of HDFS can be used only by Hadoop MapReduce workloads but not by other workloads including legacy POSIX applications and MPI-IO applications.

Gfarm file system [4] is a global distributed file system that is conformable to the POSIX semantics. It has a similar architecture to the HDFS and the Google File System in terms of federating local file systems on compute nodes. This paper discusses the design and implementation of a Hadoop-Gfarm plugin that enables access to the Gfarm file system from Hadoop MapReduce applications. Hadoop's data location aware process scheduling also benefits the Gfarm file system to improve I/O performance of MapReduce applications. Performance evaluation using micro benchmarks and typical MapReduce applications shows that the Gfarm file system has comparable performance to the HDFS. The Gfarm file system has advantage since it supports not only MapReduce applications but also POSIX and MPI-IO applications.

2. Implementation of Hadoop-Gfarm plugin

Hadoop has a modular architecture to easily extend functionality of common features. The Hadoop Common is a set of common utilities that support the other Hadoop subprojects. It includes FileSystem interface to support various kinds of file systems. Hadoop-Gfarm plug-in [5] implements this interface to enable access to the Gfarm file system through the Java Native Interface. It contains not only common filesystem APIs such as open, read, write and mkdir but also getFileBlockLocations to expose the data location of file replicas. Using this interface, Hadoop MapReduce allocates tasks near input data.

3. Performance Evaluation

Table 1 shows the machine specification of the cluster node, each node is connected using dual trunks of Gigabit Ethernet. At first, the write performance is evaluated by the Teragen program that generates 10-byte random keys and 90-byte random values. Figure 1 shows the write performance when each map task generates 5-GB data. HDFS is a little bit faster than the Gfarm using 8 client nodes and more. Although we need to figure out the reason further, one of the reasons is the HDFS does not flush data before closing the file. Currently, the HDFS does not support the flush operation, which means it cannot guarantee the success of file writes.

Figure 2 shows the read performance when the map task reads data that the Teragen outputs. Gfarm w/ affinity is result with the data aware scheduling. Gfarm w/o affinity is result without the scheduling. The data location aware scheduling impacts the read performance, which improves 55% of the read performance in the case of 16 nodes. HDFS is slightly better than the Gfarm.

Figure 3 shows the performance of grep of the same input data. The result is similar to the read performance. It is understandable because grep is read intensive-application.

Figure 4 shows the performance to sort the same input data by key. HDFS is a little bit faster than the Gfarm, although both show scalable performance when the number of clients increases.

Table 1. Machine specification

CPU	2.33GHz Quadcore Xeon E5410 (2 sockets)
Memory	32 GB
OS	Linux 2.6.18-6-amd64 SMP
Disk	Hitachi HUA72101 1TB

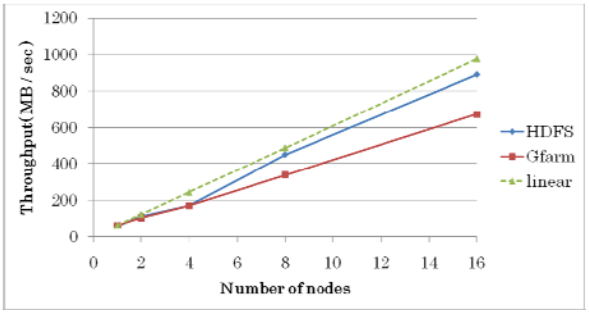

Figure 1. Write performance

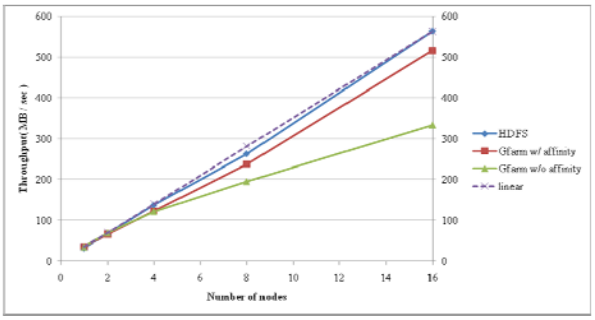

Figure 2. Read performance

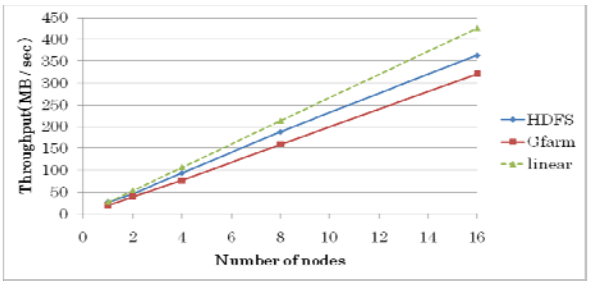

Figure 3. Grep performance

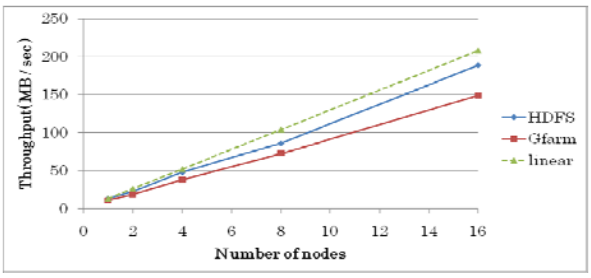

Figure 4. Sort performance

4. Summary

We have developed Hadoop-Gfarm plug-in to enable access to the Gfarm file system by Hadoop MapReduce applications. The performance result shows that Gfarm has comparable performance to the HDFS and both show scalable performance in terms of the number of clients. Gfarm has advantage such that it can be used by not only MapReduce jobs but also POSIX and MPI-IO applications.

Our future work includes the evaluation by variety of applications in wide area environment.

5. References

[1] Jeffrey Dean, Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. Proceedings of OSDI '04, 2004.

[2] Hadoop. Apache Hadoop. http://hadoop.apache.org

[3] Borthakur, D., "The Hadoop Distributed File System: Architecture and Design", http://hadoop.apache.org/common/docs/current/hdfs_design.html

[4] Osamu Tatebe, Noriyuki Soda, Youhei Morita, Satoshi Matsuoka, Satoshi Sekiguchi, "Gfarm v2: A Grid file system that supports high-performance distributed and parallel data computing," Proceedings of CHEP '04, 2004.

[5] Kazuki Ohta, Shunsuke Mikami. Hadoop-Gfarm. https://gfarm.svn.sourceforge.net/svnroot/gfarm/gfarm_hadoop/trunk/