IBIS : Interposed Big-data I/O Scheduler

Yiqi Xu Adrian Suarez Ming Zhao
Florida International University
{*yxu006,asuar054,ming*}*@cs.fiu.edu*

Abstract

Existing big-data systems (e.g., Hadoop/MapReduce) do not expose management of shared storage I/O resources. As such, application's performance may degrade in unpredictable ways under I/O contention, even with fair sharing of computing resources. This paper proposes *IBIS*, a new Interposed Big-data I/O Scheduler, to provide performance differentiation for competing applications' I/Os in a shared MapReduce-type big-data system. IBIS is implemented in Hadoop by interposing HDFS I/Os and use an SFQ-based proportional-sharing algorithm. Experiments show that the IBIS provides strong performance isolation for one application against another highly I/O-intensive application. IBIS also enforces good proportional sharing of the global bandwidth among competing parallel applications, by coordinating distributed IBIS schedulers to deal with the uneven distribution of local services in big-data systems.

1 Introduction

Big-data applications need to process and analyze massive amounts of data in parallel (e.g., MapReduce [2]) and often have complex I/O phases is highly distributed across many data nodes. Thus, storage systems that can provide high scalability and availability (e.g., Hadoop HDFS [3]) needs to be SLA aware in the shared infrastructure. However, existing big-data systems do not expose management of shared storage I/O resources. As a result, an application's performance may degrade in unpredictable ways when there is I/O contention.

This paper proposes *IBIS*, a new Interposed Big-data I/O Scheduler, to provide performance differentiation for competing applications' I/Os in a shared MapReduce-type big-data system. This scheduler solves the problem of differentiating the I/Os among competing applications *on individual data nodes* and schedule them according to the applications' bandwidth demands. The proposed IBIS scheduler is able to transparently intercept the I/Os from big-data applications and schedule them on every data node via an I/O interposition layer. IBIS also coordinates I/O scheduling *across distributed data nodes* to allocate the total storage service of the entire big-data system to the parallel tasks of competing applications.

The IBIS prototype is implemented in Hadoop by interposing HDFS I/Os and scheduling them using an SFQ-based proportional-sharing algorithm [4]. Experimental results show that with IBIS, an application's performance can be strongly isolated from the contention by a highly I/O-intensive application (TeraGen) ($< 5\%$ slowdown in total runtime), even with uneven available bandwidth on different nodes.

2 Approach

IBIS is designed to *effectively differentiate I/Os from competing applications and allocate the shared storage bandwidth on individual data nodes in a big-data system*. IBIS is based on *virtualization* principles (Figure 1), where an indirection layer exposes the interfaces already in use by the big-data system to access storage, allowing applications to time-share the storage system without modifications, while enforcing performance isolation and differentiation among them. Step 1-5 corresponds to map read, map output, reduce shuffle, reduce merge and reduce write. We chose to introduce virtualization at a DataNode layer of the storage hierarchy to gain more control of I/O executions and utilization while supporting more diverse applications. The DFSClient interface between the tasks and DataNode is modified to allow application-specific information to be carried as part of the request header of each block request issued by the map/reduce task, transparent to the applications.

IBIS also *efficiently coordinates the distributed I/O schedulers across data nodes in order to allocate the global storage bandwidth for the parallel tasks of applications in a big data system*. The total service that an application gets across the whole system is the sum of the services that it obtains from every data node where its tasks run. The amount of local service that it actually obtains from a data node varies across nodes and over time and each local scheduler needs a global view of aggregate I/O throughput to converge to the I/O sharing ratio collectively on all data nodes. To address the challenge of synchronization of global I/O view between data nodes, the IBIS schedulers exchange their local I/O service information and obtain global views of total I/O services by piggybacking upon the existing RPCs between TaskTrackers and JobTrackers. The scalability of this global coordination scheme is made possible by the scalability of the JobTrackers (in YARN [1] for large systems). Specifically, local scheduler adjusts the local I/O service ratios among the tasks on its data node in order to achieve global fairness of total I/O service among competing parallel applications, by delaying those that are above their global fair shares and promoting those below their global fair shares.

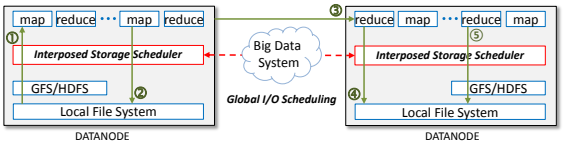

Figure 1: Architecture of IBIS

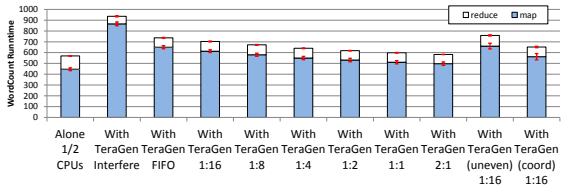

Figure 2: Runtime of WordCount with/without IBIS with varying I/O sharing ratios (WordCount:TeraGen)

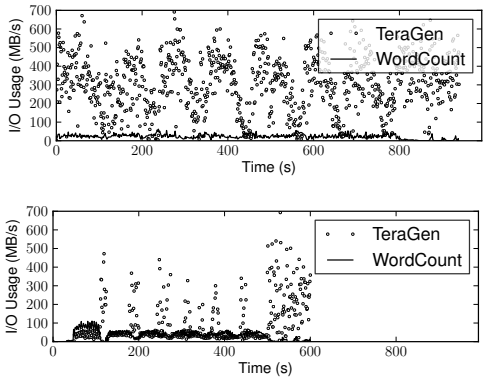

Figure 3: WordCount aggregate I/O throughput under TeraGen contention without and with IBIS (2:1)

3 Evaluation

Hadoop-based IBIS prototype was implemented and evaluated on a testbed consisting of eight nodes each with two six-core 2.4GHz AMD Opteron CPUs, 32GB of RAM, and two 500GB 7.2K RPM SAS disks, interconnected by a Gigabit Ethernet switch. All the nodes run the Debian 4.3.5-4 Linux with the 3.2.20-amd64 kernel and use EXT3 as the local file system. One of the eight cluster nodes runs *JobTracker*, one as *NameNode*, and the other six nodes as *TaskTrackers* as well as *DataNodes*. HDFS is configured to use one of the two disks on available on each data node, while the other is used to the data spilled directly to the local file system (map outputs and reduce inputs) to reduce self-interference. Each node is assigned 10 map slots and 2 reduce slots, with Hadoop fair scheduler turned on for equal share of slots between two applications so the contention is purely from I/O side.

Figure 2 show a comparison of different WordCount runtime when run alone(half CPUs) or against TeraGen(two evenly using all CPUs). TeraGen's I/O contention caused more than 65% runtime increase to WordCount from the 1^{st}(alone) bar to the 2^{nd} bar although native Hadoop fair scheduler assigns the same number of CPUs to both jobs. When applied a virtualization layer with a depth of 4 in the 3^{rd}(FIFO) bar, the depth control of writes already shielded partial interference to WordCount, reducing the runtime by 21%. From the 4^{th} bar to the right we apply SFQ(D) gradually increasing the share of TeraGen, and achieved within 105% of original alone performance at the ratio of 2:1. The last two bars shows with uneven available bandwidth on one of the data nodes (introduced by another I/O intensive application), uncoordinated (uneven) 1:16 target ratio cannot be reached as when bandwidth is even (4^{th} bar). By adjusting unaffected nodes' bandwidth share, coordinated case on the rightmost bar(coord) can gain performance back.

Figure 3 collects per-second aggregate HDFS system bandwidth allocated to WordCount and TeraGen, without and with IBIS(2:1). TeraGen writes suppressing the WordCount I/O without IBIS is on the top figure, while bottom figure shows the effectiveness of IBIS by: *1)* allowing approximately 1/3 of the available bandwidth to TeraGen and 2/3 to WordCount; *2)* allowing TeraGen to consume available bandwidth when WordCount issues less I/O. As a result, WordCount's I/Os are prioritized on all the datanodes and completes faster by 40%.

4 Conclusion and Future Work

This paper proposes IBIS, an Interposed Big-data I/O Scheduler, to provide global I/O performance differentiation to big-data applications. Experimental evaluation shows with IBIS, an application's (WordCount) performance can be strongly isolated from the contention generated by a highly I/O-intensive application (TeraGen) ($< 5\%$ slowdown in total runtime). The results also show that IBIS can effectively achieve specified sharing ratio of the global bandwidth between two competing parallel applications by coordination. In the future work, IBIS will support the scheduling of other I/Os used by big-data applications in addition to HDFS I/Os. The I/O scheduling provided by IBIS will then be integrated with the existing CPU scheduling in big-data systems. Both types of resources are essential to the different stages of big-data applications and need to be managed holistically to achieve the application-desired quality of service.

References

[1] Yet another resource negotiator. hadoop.apache.org/docs/current/hadoop-yarn/.

[2] DEAN, J., AND GHEMAWAT, S. MapReduce: simplified data processing on large clusters. In *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6* (Berkeley, CA, USA, 2004), OSDI'04, USENIX Association, p. 10.

[3] SHVACHKO, K., KUANG, H., RADIA, S., AND CHANSLER, R. The Hadoop Distributed File System. In *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)* (May 2010), IEEE, pp. 1–10.

[4] WANG, Y., AND MERCHANT, A. Proportional-share scheduling for distributed storage systems. In *FAST* (2007), USENIX, pp. 47–60.