IEICE TRANS. INF. & SYST., VOL.E100–D, NO.12 DECEMBER 2017

LETTER

DiSC: A Distributed In-Storage Computing Platform Using Cost-Effective Hardware Devices

Jaehwan LEE[†a)], ***Member***, **Joohwan KIM**[†], ***and*** **Ji Sun SHIN**[††b)], ***Nonmembers***

SUMMARY The ability to efficiently process exponentially increasing data remains a challenging issue for computer platforms. In legacy computing platforms, large amounts of data can cause performance bottlenecks at the I/O interfaces between CPUs and storage devices. To overcome this problem, the in-storage computing (ISC) technique is introduced, which offloads some of the computations from the CPUs to the storage devices. In this paper, we propose *DiSC*, a distributed in-storage computing platform using cost-effective hardware. First, we designed a general-purpose ISC device, a so-called DiSC endpoint, by combining an inexpensive single-board computer (SBC) and a hard disk. Second, a Mesos-based resource manager is adapted into the DiSC platform to schedule the DiSC endpoint tasks. To draw comparisons to a general CPU-based platform, a DiSC testbed is constructed and experiments are carried out using essential applications. The experimental results show that DiSC attains cost-efficient performance advantages over a desktop, particularly for searching and filtering workloads.

key words: *distributed data processing, in-storage computing, Mesos, single-board computer*

1. Introduction

Big data applications are increasingly accounting for a greater percentage of the workloads being placed on general computing systems. As such, a shift has occurred in computing platform cores from CPU-intensive arithmetic computations to I/O intensive processes that efficiently transfer large amounts of data from storage to memory. However, when an application requires the transfer of a large amount of data, the I/O interface between the CPU and storage device becomes a performance bottleneck, which degrades the performance of the total system. Therefore, there is a need for a new storage technique that can overcome these I/O bottlenecks. One of approach to mitigate the I/O bottleneck can be to increase I/O bandwidth using high-performance storage interface like Non-volatile memory express (NVMe). However, hardware performance improvement has limitation such as scalability to deal with large amount of data. Of note, although big data applications need to scan a large amount of data in storage systems, it is common that only a small amount of target data needs to be processed. If unnecessary data can be filtered out within the storage system, the amount of data transmitted over the I/O interface from storage to main memory can be significantly reduced. The in-storage computing (ISC) technique has been proposed by several studies [1]–[3] as a way to solve the performance bottleneck caused by differences in I/O interface speeds. ISC uses a separate computing processor added to the storage to offload the work that can be performed before sending the data.

This paper proposes a distributed in-storage computing (DiSC) platform using cost-effective hardware. First, we model a general-purpose ISC device, called a DiSC endpoint, using an inexpensive single-board computer (SBC) and a hard disk to increase data transmission efficiency in an I/O interface by adding intelligence inside storage device. Because previous ISC devices [4] are company-dependent and closed to the public, this study proposes an open ISC architecture using cost-efficient hardware and commodity operating systems. Second, DiSC adapts a Mesos-based resource manager to schedule tasks and assign them to the DiSC endpoints [5]. To cover extensive applications, Mesos, an open-source based distributed scheduler, is employed. We built a testbed cluster using DiSC endpoints and a Mesos resource manager for evaluation. Extensive experiments are conducted on the DiSC testbed, and the performance results are compared to the results from a legacy desktop machine.

The contributions in this paper are as follows. First, a new open-architecture, open-source platform is suggested for distributed ISC using cost-effective hardware. Using DiSC endpoints and a Mesos-based scheduler, users can execute ISC applications without modifying source codes. Thus far, it supports C, C++, and Python-based applications on DiSC endpoints. Second, extensive experiments are conducted on a real DiSC testbed, and the cost-efficiency of DiSC is verified. For example, the Egrep application shows better performance in DiSC than on a desktop machine by up to two times in terms of the price-performance ratio. According to the experimental results, DiSC shows excellent performance depending on the offloaded workload type; the search and filtering applications show the best performance on the DiSC platform, which corresponds to ISC's original purpose.

2. DiSC Cluster Architecture

2.1 DiSC Endpoint Hardware Architecture

The DiSC testbed cluster, which implements the distributed

Manuscript received May 11, 2017.
Manuscript revised July 31, 2017.
Manuscript publicized August 23, 2017.

[†]The authors are with the School of Electronics and Information Engineering, Korea Aerospace University, Goyang-city, Korea.
[††]The author is with the Department of Computer and Information Security, Sejong University, Seoul, Korea.
a) E-mail: jlee@kau.ac.kr
b) E-mail: jsshin@sejong.ac.kr (Corresponding author)

DOI: 10.1587/transinf.2017EDL8104

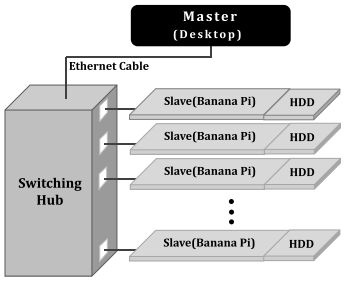

Fig. 1 DiSC cluster diagram.

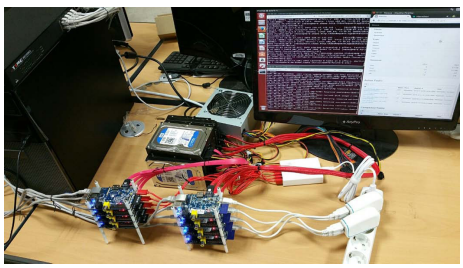

Fig. 2 DiSC cluster testbed.

in-storage processing system, consists of a single main processing machine (desktop) and four DiSC endpoints, as shown in Fig. 1. Each DiSC endpoint is composed of Banana Pi, an SBC equipped with a SATA interface and 1GB Ethernet, and an HDD. Banana PI can serve as a processing unit in a storage device to handle offloaded computations. A single DiSC endpoint does not offer improved internal I/O bandwidth, but we can move computation to DiSC endpoint to achieve optimized total system performance.

Lubuntu 3.1.1, a limited-resource computer OS, is installed on the Banana Pi to build a cluster environment. The Ubuntu 14.04 LTS version is installed on the desktop. Instead of directly attaching HDDs via the SATA interface to the main machine, DiSC endpoints are attached to the main machine over the 1GB Ethernet network. Since the bandwidth of the 1GB Ethernet is sufficiently large compared to the throughput of an HDD, DiSC can be modeled over Ethernet as an active ISC device with storage. Banana PI with Lubuntu is a small-size computing device, so all application types can be offloaded without modifying source codes in the storage device. Figure 2 shows the testbed DiSC cluster.

2.2 Distributed Processing Frameworks

2.2.1 Mesos

Mesos [5], [6] is a framework for distributed systems used in data centers and cloud environments. The Mesos kernel runs in each machine and supports resource management, fault tolerance, and scheduling functions in conjunction with applications such as Hadoop, Spark, Kafka, Elastic Search, Zookeeper, and Marathon. The architecture of Mesos is shown in Fig. 3. Mesos consists of a master daemon that controls each cluster node, and Mesos frameworks that execute actual tasks on these nodes. A Mesos framework is composed of two parts: a resource scheduler to manage resources and a process executor to run the tasks.

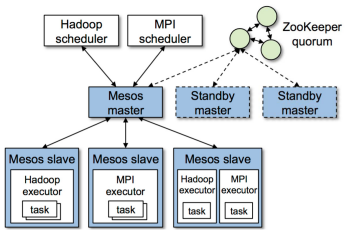

Fig. 3 Mesos architecture diagram, showing two running frameworks [6].

2.2.2 Marathon

Marathon [7] is a container-integrated Mesos framework. It creates an application and provides a resource scheduling function to the slave. Marathon also provides Web UI. After creating several configurations, the application is created and each slave node starts executing the task after job scheduling.

3. Experiments

3.1 Experiment Environments

The configuration of the cluster used in the experiment consists of four DiSC endpoints, which serve as slave nodes, and a desktop machine, which is the master node. The desktop has an Intel (R) Core (TM) i5-3470 CPU @ 3.20GHz CPU and a 4GB memory. The Banana Pi in each DiSC endpoint has an A20 ARM Cortex-A7 Dual-Core CPU and a 1GB memory. Desktop node and Banana Pi node have a single HDD via SATA interface. Performance is compared for three cases: (1) the DiSC cluster, (2) the desktop machine (denoted as CPU processing), and (3) Distributed Storage. For the CPU processing, the available memory in the desktop is set to 2800MB, and the applications are only executed on the CPU in the desktop. For the DiSC case, the available memory is set to 700MB in Banana Pi, and the applications are run on Banana Pi. 700MB is selected after considering the available memory in each case. In this experiment, Marathon is used as a scheduler. When DiSC creates an application in Marathon, the memory is set to 700MB, and the instance is set to 4. Distributed Storage is the case that the HDDs in DiSC Endpoint is mounted to the Desktop machine via Networked File System (NFS), so DiSC Endpoint works as a dummy storage device case. In this case, all processing is done in CPU in the desktop.

3.2 Wordcount Result

Wordcount is an I/O intensive workload, but it also needs a lot of memory and a significant number of computations. In this experiment, four different input data file sizes are used

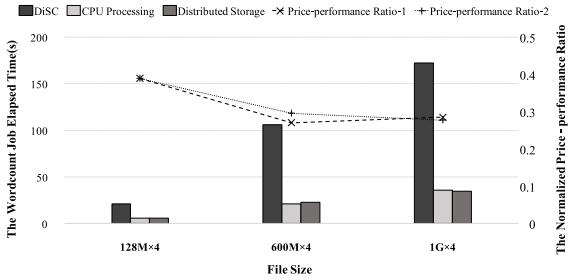

Fig. 4 Execution times of Wordcount application.

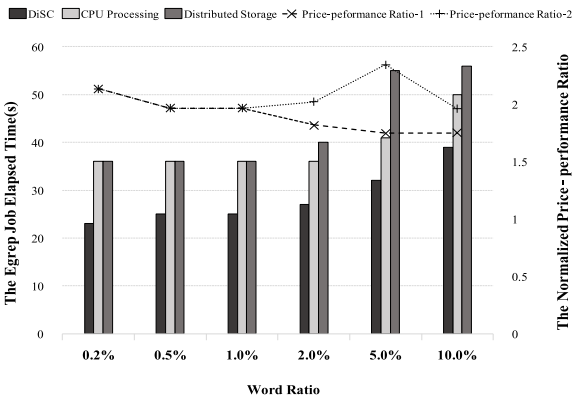

Fig. 5 Execution times of Egrep application.

- 128MB×4, 600MB×4, and 1GB×4. The experimental results are shown in Fig. 4. It shows that the overall performance of CPU processing and Distributed storage exceeds that of DiSC. There are two reasons for this. First, Wordcount applications require a large amount of memory to store the intermediate results, and it also requires a number of computations to match words. ARM CPU in Banana PI does not have sufficient computing power compared to the Intel CPU in the desktop, and the available memory in Banana PI is limited to the desktop environment. Second, even though Wordcount summarizes raw input data, its output size is not significantly reduced. Typically, an ISC model benefits from reduced transfer times at the I/O interface despite the additional CPU cycles in ISC devices. However, Wordcount is not an application with a small output-input ratio, so the reduced data amount is not large compared to the processing time in the ISC device.

3.3 Egrep Result

In the Egrep experiment, measurements are taken for the execution time needed to scan keywords in a 1GB XML file. The experiment varies the keywords, which have different populations in the input file. As shown in Fig. 5, the experiment is divided into six cases - 0.2%, 0.5%, 1%, 2%, 5%, and 10% of the keyword populations. The results show similar execution times when the word population ratio is less than 2%. However, when the ratio increases (5% and 10%), the execution times increase. This indicates that the DiSC cluster consistently outperforms the CPU processing and the Distributed storage. In contrast to Wordcount, Egrep is a search algorithm. It is not influenced as much as Wordcount by memory size, cache memory, or CPU performance. The results demonstrate that DiSC can overcome the difference in CPU performance and turn out a well-scaled distributed processing system with excellent performance levels. Thus, DiSC can perform well in applications that do not overload the CPU and RAM usage and reduce output significantly.

3.4 Cost Efficiency

Because DiSC endpoints have limited resources based on inexpensive hardware, the DiSC and CPU processing price-performance ratios are compared. The price-performance ratio is defined as an inverse of the product of execution time and hardware costs. The ratios are calculated by summing up the hardware costs in each case[†]. The dotted lines in Figs. 4 and 5 show two kinds of the normalized price-performance ratio. The first is equal to the DiSC price-performance ratio divided by the CPU processing price-performance ratio, and the second is is equal to the DiSC price-performance ratio divided by the Distributed storage price-performance ratio. In other words, if the normalized price-performance ratio is larger than 1.0, DiSC is a more cost-efficient system than other type of processing. For Wordcount, CPU processing has a better ratio than DiSC. However, for Egrep, DiSC has a better ratio than CPU processing. Egrep remains a well-suited application for an ISC platform.

3.5 Discussion

In case of CPU processing and Distributed storage, data transfer time from storage to main memory is dominant to the total execution time. Since we use 1Gbits Ethernet to DiSC Endpoints, bandwidth of networked storage device is as large as that of the local storage, so the Distributed storage cannot get any benefits from the distributed architecture. It means that reducing data transfer in storage device through ISC works very effectively with distributed architecture.

However, if the dependency between data across different storage devices exists, DiSC endpoints might need to get the data from other endpoint through network. We are working on this feature like *shuffle* in Hadoop now, but DiSC architecture mainly focuses on distributed ISC for independent data because shuffle type of operation can incur bottleneck in the network.

RAID is one of popular technique to increase throughput of storage system. In our DiSC, we do not support RAID technique for throughput improvement, but block-level parallel execution in DiSC can increase the total system performance in distributed manner.

[†]Hardware price costs are estimated from Amazon prices.

4. Conclusions

This paper proposes DiSC, a new open-source, open-architecture ISC platform using cost-effective hardware. Various experiments are conducted, and the performance results are compared. These results indicate that search type applications perform better in DiSC. In order to solve the performance bottleneck problem at the I/O interface when processing a large amount of data, ISC applications in distributed systems are very important. The DiSC experimental results can be also extended to data centers or cloud that process large amounts of data for search and filtering workload.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (No. 2015R1C1A1A02036524), and by the GRRC program of Gyeonggi province [Video-Audio Space Convergence Technology Research Center].

References

[1] D. Park, J. Lee, and S. Hong, "SSD software platform simulator for in-storage processing," Journal of KIISE: Computing Practices and Letters, vol.18, no.8, pp.602–606, 2012.
[2] J. Do, Y.-S. Kee, J.M. Patel, C. Park, K. Park, and D.J. DeWitt, "Query processing on smart SSDs: Opportunities and challenges," Proc. 2013 ACM SIGMOD, pp.1221–1230, 2013.
[3] Y. Kang, Y.-S. Kang, E.L. Miller, and C. Park, "Enabling cost-effective data processing with smart SSD," 2013 IEEE 29th Symposium on Mass Storage Systems and Technologies (MSST), May, 2011.
[4] B. Gu, A.S. Yoon, D.-H. Bae, I. Jo, J. Lee, J. Yoon, J.-U. Kang, M. Kwon, C. Yoon, S. Cho, J. Jeong, and D. Chang, "Biscuit: A framework for near-data processing of big data workloads," 2016 ACM/IEEE ISCA, pp.153–165, Seoul, 2016.
[5] Apache Mesos, http://mesos.apache.org/
[6] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A.D. Joseph, R. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center," UC Berkeley, 2010.
[7] Marathon, https://github.com/mesosphere/marathon