An FPGA-based Tightly Coupled Accelerator for Data-intensive Applications

Masato Yoshimi, Ryu Kudo, Yasin Oge
Graduate School of Information Systems
University of Electro-Communications
Chofu, Tokyo, JAPAN 182-8585
Email: yoshimi@is.uec.ac.jp

Yuta Terada
AVAL DATA
Machida, Tokyo, JAPAN 194-0023
Email: teraday@avaldata.co.jp

Hidetsugu Irie, Tsutomu Yoshinaga
Graduate School of Information Systems
University of Electro-Communications
Chofu, Tokyo, JAPAN 182-8585
Email: irie,yosinaga@is.uec.ac.jp

***Abstract*—Computation beside a data source plays an important role in achieving a high performance with low energy consumption in Big Data processing. In contrast to that of a conventional workload, the processing of Big Data frequently requires that a massive amount of data in distributed storage be scanned. A key technique for reducing energy-consuming processor loads is to install a reconfigurable accelerator that is tightly coupled to a computational resource with interfaces. The accelerator is capable of configuring application-specific hardware modules to allow some logical and arithmetic operations for data stream transmission between interfaces, as well as the offloading of control protocols for communication with other computing nodes or storage. In this paper, an FPGA-based accelerator, which is directly attached to DRAM, the network, and storage, is proposed in order to realize an energy efficient computing system. A simple application that counts the words appearing in the data is implemented to evaluate a prototype system. As the accelerator outperforms by 80.66 to 429 times similar applications executed on an SSD-based Hadoop framework, we confirm that the accelerator's utilization for Big Data processing is beneficial.**

***Index Terms*—FPGA-based tightly-coupled accelerator, Simple Word Counting, Big Data processing**

I. Introduction

A massive amount of data has been generated and accumulated as a result of the popularization of the various Web services and the growth of sensor technologies. Big Data processing plays an important role in extracting profitable perceptions from these large quantities of data. The MapReduce framework [1] advocated by Google encouraged many organizations to utilize Big Data in various application fields, such as decision making, risk management, and advertisement. Apache Hadoop is a well-known and commonly used implementation of a software framework based on MapReduce [1] and GFS [2] to achieve scalability, to improve fault-tolerance, and to increase I/O throughput by using parallel access to multiple storage locations in spatially distributed computers. A technique for scanning a large amount of data in storage is one of the most important factors in Big Data processing. Since the I/O throughput and latency of a magnetic disk drive are very slow as compared to those of the other semiconducting components in a computer, a PC cluster, the computing nodes of which provide multiple paths to storage devices, is used to improve I/O throughput by performing parallel read and write operations. In a recent project for building data centers, it has been promoted that the adoption of high-speed flash storage for storing data for highly efficient computing systems[3]. Although adopting an SSD (Solid State Drive) contributes to improving the I/O throughput, the platform for Big Data processing still requires a large number of nodes to complete an application within a practical computation time. It raises the problem of the operating cost to maintain many computing nodes installed only to provide throughput for data access.

There are two primary approaches for reducing the operating cost of Big Data processing: (1) Improving computational speed by an accelerator and (2) reducing the number of nodes required by improving the I/O throughput for storage. In the case of MapReduce, several studies on utilizing multi-core processors [4], GPUs [5] and dedicated hardware [6] are in progress.

To resolve the problem, we propose an FPGA-based accelerator that is tightly coupled with flash storage and optical network interfaces. In this paper, we describe a mechanism for accelerating Big Data processing by using FPGA-based hardware tightly coupled with storage, a network, and a host PC. The performance is evaluated for a prototype system using a simple word counting application. The primary contributions of this paper are as follows.

- An FPGA-based accelerator tightly coupled with storage and a network is proposed,
- The implementation of an accelerator that performs data stream operations is described, and
- The actual performance of the system using a PC-cluster consisting of four nodes is evaluated.

II. Related Research

A. Big Data Processing Utilizing Accelerators

Big Data processing requires speed and efficiency, since a vast amount of data has to be computed. The installation of an accelerator in the computing system has been described as effective. It has been reported that accelerators, such as GPU [5], FPGA [6] [7] [8], Cell B./E. [9], and TILE64 [4], contribute to an improvement in the processing performance.

In [4], it was shown that the storage throughput and I/O response determine almost the entire computational time in current Big Data processing. A software-based approach for developing a novel computing sequence of the MapReduce framework to reduce the I/O wait time to write and read intermediate data was proposed [10][11].

The integration of data transfer and computation is expected to be an effective approach, since all data are frequently scanned in Big Data processing. By attaching a storage computing functionality, the CPU load can be mitigated drastically by offloading operations and access to reduce latency by trimming data[12][13].

Several studies on accelerating applications by using storage tightly coupled with computation are also advancing. For specific applications, such as chromosomal microarray analytics in biology, in [14] it was shown that a computing system called the Netezza Performance Server outperforms parallel computation on a PC cluster and Hadoop. Netezza [15], which is a shared-nothing architecture, consists of an Active Disk, equipped with blade servers, that is an FPGA tightly coupled to storage as a data warehouse appliance. In the field of medicine, hardware tightly coupled to storage is also applied for storing large amounts of MRI images obtained using fast recording [16]. IBEX is a back-end database system that uses an FPGA board, which is equipped with a direct SATA-II link to an SSD [17]. In [17], a small setup of the system was demonstrated and evaluated. It was shown that the system outperforms software-based implementations dozens of times in terms of energy consumption and computational speed.

B. FPGA-based Accelerators for Big Data

High-speed Big Data processing using a reconfigurable processor has become an active area of research [18] [19]. For a search engine, in [18] FPGA was adopted to connect computing nodes via SAS ports in order to create a macro pipeline that integrates a number of computing nodes. In [19], an FPGA-based memcached appliance was developed that consists of a distributed key value store on DRAMs to replace energy-consuming server machines. Jun *et al.* proposed an FPGA-based system with an on-board flash array as storage, an inter-FPGA network interface, and accelerator modules [20]. In [20], the computing system provides a file system implemented by FUSE to realize the control of fine-grained units of data. Since the system adopted certain old flash chips in the FPGA board, which provide less throughput than do more advanced flash storages, and a DRAM-cache was not implemented, the architecture should be evaluated using more advanced devices.

Our study has the following advantages over the previous research studies mentioned in this section.

- A direct connection between data in external storage, including a large amount of data that cannot be contained in DRAMs, and accelerator modules to enable high-speed data transmission by utilizing Hard IPs in a recent FPGA,
- A flexible accelerator that takes advantage of built-in memory, DRAM, and IPs, and
- Evaluation using a large amount of data over 100 GB.

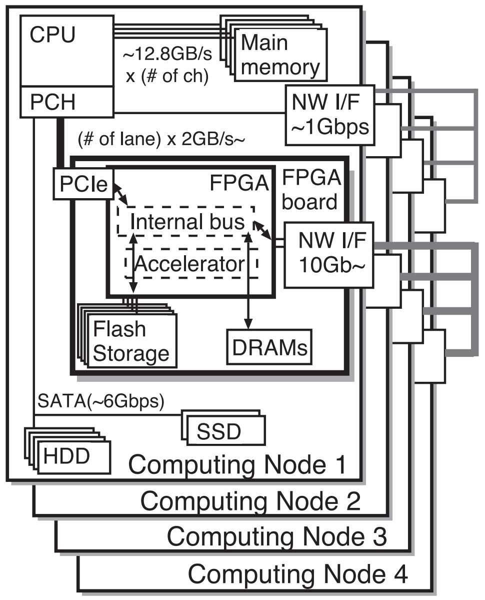


Fig. 1. The proposed computing system

III. FPGA-BASED TIGHTLY COUPLED ACCELERATOR

A. Distributed Systems for Big Data

In this paper, we propose an FPGA-based tightly-coupled accelerator for data-intensive applications. Tightly-coupled means combining processors or accelerators with a data source, such as storage and a network. For example, hardware-based RAID is one of the most well known functionalities of tightly-coupled accelerators for reducing the CPU load. The FPGA-based tightly-coupled accelerator integrates computing resources and interfaces configured by FPGA into an offloading engine attached to a host PC.

Fig. 1 shows the entire structure of the computing system. The computing system consists of a PC cluster with shared-nothing architecture that uses multiple commonly-used PCs as computing nodes. Each computing node is equipped with an FPGA board via a PCI-Express bus as an accelerator. The FPGA board provides four interfaces, which communicate with DRAM, storage, the network and a host PC, respectively. Interfaces and accelerator modules are integrated with the internal bus and configured on an FPGA. Users of the computing system can issue commands to transfer data between them.

The storage interface is connected to a flash storage, which is faster than conventional magnetic storage. The data to be processed are distributed to the computing nodes and stored in the flash storage. The network interface is also connected to the other computing nodes and data are transferred between DRAMs on FPGA boards. The network is isolated from the inter-node network, since the network provides a dedicated protocol to exploit the transference performance.

The computing system has the following two advantages. (1) The FPGA preloads user-defined custom logic to accelerate computation. The accelerator applies operations frequently

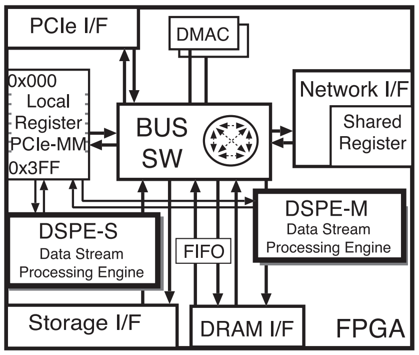


Fig. 2. The accelerator implemented on the FPGA

used in large-scale data processing, such as filters and aggregations, to data stream transference between interfaces. The active-disk mechanism introduces an improvement in computational efficiency by avoiding the occupation of the main memory bus by such enormous, but uncomplicated, operations. (2) The host PC can issue a command to transfer data stored in the flash storage to other computing nodes without relying on the main memories. The CPU load is also mitigated, since the FPGA covers the process of controlling the communication protocol to read and write to the storage.

B. The In-chip Hardware Architecture

Fig. 2 shows the in-chip structure of the accelerator configured in an FPGA. Based on the products provided by AVALDATA, we expanded the architecture to integrate the functionality of accelerators. The accelerator consists of the following components.

- Interfaces for the network, storage, DRAM and PCI-Express bus,
- A bus switch that connects the interfaces to each other,
- A local register array, which is memory-mapped to the PCI-Express address,
- DMA controllers that transfer data between interfaces, and
- Two types of accelerator modules, called DSPE-S and DSPE-D. DSPE stands for Data Stream Processing Engine, and the postfixes S and D stand for Storage and DRAM, respectively.

To smooth the transmission of a large amount of data between interfaces, an FIFO module, which connects two of four channels to access DRAM, is used as a buffer. The host PC kicks the DMACs via PCI-Express to transfer data between interfaces.

DSPEs are primary hardware modules in the accelerator for applying some logical and arithmetic operations for data stream transmission through the bus switch. The two DSPE modules can be implemented according to the applications. The DSPE-S in the output signals of the storage is a hardware module for streaming algorithms such as encryption and for scanning filter and aggregation algorithms, since a unit of data read from the storage is too large to be stored in the embedded memories in the FPGA. On the other hand, DSPE-D in the input signals of DRAM is suited to more flexible operations, including random access to DRAM, since a unit of data communicated between the FPGA and DRAM is sufficiently small. For example, DSPE-D can be implemented in scatter/gather operations.

TABLE I
SPECIFICATIONS OF COMPUTING SYSTEM

No. of Node	4	
CPU	Intel Core i7-4770S	3.10GHz (4C8T)
Memory	DDR3 1866MHz$\times$4	32.0 GB
Motherboard	GIGABYTE G1.Sniper M5	
Network	on board Ethernet	1Gbps
SSD	Crucial CT480M500SSD1	480GB
HDD	Seagate ST2000DM001	2.0TB

TABLE II
SPECIFICATION OF AVALDATA APX-880A

FPGA Device	Stratix IV GX EP4SGX230KF40C2	125 [MHz]
DRAM	DDR3(533MHz), 512MB	8.53 GB/s
Storage	18 SD Memory Cards$\times$1 or 2 (up to 64[GB]$\times 32 = 2$[TB]) unit to transfer	1.5 GB/s$\times$2ch two SDs for parity 128[MB]
Network	Proprietary GiGA CHANNEL Optical token ring network unit to transfer	8.5 Gbps$\times$2ch 16[KB]
PCIe I/F	Gen2$\times$4 Lane	2.0 GB/s (simplex)
Internal Bus	Proprietary AVAL-bus	128bits-width

IV. SETTING UP THE PROTOTYPING SYSTEM

A. Experimental Environment

A prototype system was built as an experimental environment for evaluating the proposed system described in Section III. The prototype system is a PC cluster consisting of four computing nodes, as shown in Tab. I, with an AVALDATA APX-880A [21] as an FPGA-based accelerator, the specifications of which are shown in Tab. II. Since the storage interface accesses two sets of 18 SD memory cards in parallel in order to improve throughput, the size of the data block, which is read or write at a time, is 128 MB. Parallel data reading and writing to SD memory cards allows the throughput with DRAM to be higher than 1 GB/s. This performance is superior to that of recent PCI-Express-based SSD products.

The operating system and other software for each computing node are installed as shown in Tab. III. We connected all the

TABLE III
OPERATING SYSTEM AND SOFTWARE INSTALLATION

OS	CentOS 6.5 kernel-2.6.32-431.11.2.el6.x86_64
SW Compiler	gcc-4.4.7 -O3
HW CAD	Altera Quartus 10.1sp1
Hadoop	Cloudera CDH 4.6.0-1.cdh4.6.0.p0.26 Data replication: 1, HDFS is configured on SSD

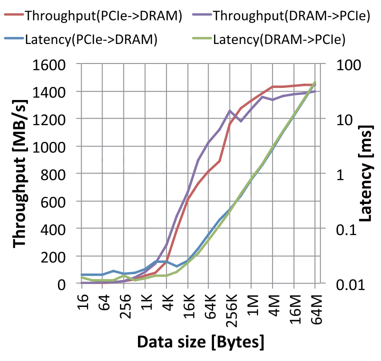


Fig. 3. Performance of Data Transfer to DRAM via PCI-Express bus

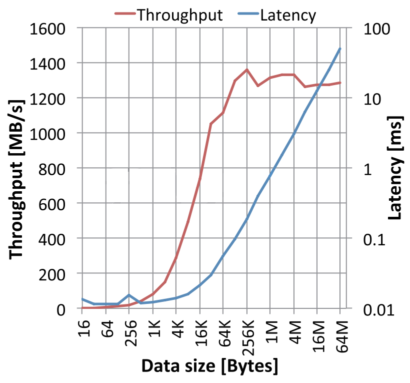


Fig. 4. Performance of data transfer via network

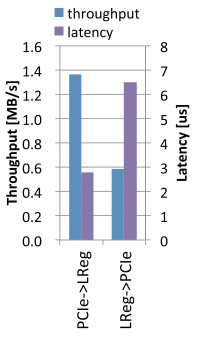


Fig. 5. Performance of read/write to local register array (4 Bytes/command)

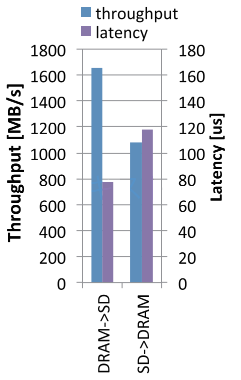


Fig. 6. Performance of data transfer between DRAM and storage (128MB/Block)

APX880A boards in each computing node by optical cables. Eighteen SD memory cards of 32 GB were installed in each APX-880A to construct 512 GB of flash storage per node.

B. Implementation of DSPE

We implemented the hardware and customized the device driver to realize a tightly-coupled accelerator as follows.

- The accelerator modules, DSPE-S and DSPE-D, are embedded, as shown in Fig. 2. The accelerator modules have also signals to access the local register array to communicate commands and results to the host PC.
- The device driver is customized to allow flexible data transfer between interfaces for DRAM, storage, network, and PCI-Express bus, since the product of APX-880A provides only limited paths between storage, network, and PCI-Express for storing high-speed images from multiple cameras.

A hardware with empty DSPE-S and DSPE-D, which only pass signals, is called “Base” to refer to the evaluation for resource utilization.

C. Performance of Data Transfer Primitives

Before implementing the accelerator modules, the data transfer capabilities between the interfaces in the Base hardware are measured, since the throughput determines the performance of the accelerator.

Fig. 3 shows the latency and throughput of data transfer between the main memory in the host PC and DRAM on APX-880A, when a variable amount of data is transferred through the PCI-Express bus. Since the current device driver does not support pipelining to issue read and write commands, the latency is much lower than that of the recent PCI-Express-based SSD when the data are smaller. However, the throughput reached approximately 1.4 GB/s, which is considered reasonable, since the theoretical simplex throughput of PCI-Express Gen2 with four Lanes is 2.0 GB/s.

The network performance among the DRAMs of the four APX-880A boards, shown in Fig. 4, was evaluated by measuring the turn around time from a command being issued to the complete transference of the data. Although the throughput is a little lower than the theoretical value, it does not cause performance degradation, because data are copied to all DRAMs by using the token ring network, which uses the “round-robin” message transmission strategy.

The performance of read/write to the local register array from the host PC is shown in Fig. 5. Since a unit of data is very fine, consisting of 4 Bytes, repeatedly updating the local register array has an influence on the whole computation time of the application. Fig. 6 shows the throughput and latency of data transfer between DRAM and the storage on APX-880A and its effect on the performance of the DSPE-S.

Although the unit size of data for accessing storage limits the performance of the application, the capability of the FPGA board is shown to be sufficient to allow data-intensive applications to scan and apply operations to all the data in the storage.

V. Evaluation for an Application

A. Example Application

To evaluate the advantageous effect of the accelerator, we implemented an accelerator module that executes Simple Word Counting, abbreviated to SWC hereafter. SWC is a good benchmark for evaluating the performance of a hardware-assisted platform for Big Data processing and was also adopted in [20]. It is a simplified application of Wordcount that counts the number of user-registered words appearing in text data. Although Wordcount, which obtains the number of all the words that occur in the text, is frequently used to show the superiority of software-based distributed systems and algorithms, the accelerator cannot easily handle all the counts of a word, because Wordcount requires a large hash table. When Wordcount is computed on the accelerator, the user should be computed by circulation of SWC to count the all types of word.

We adopted an example program provided by Apache Hadoop, which counts the number of user-specified expressions in text files in HDFS as a software-based target for comparison purposes, since the program is very similar to SWC.

B. SWC Accelerator

Prior to implementation, we designed an SWC accelerator module to be configured in the DSPE. Although the text processing performance is an important application in a distributed framework, such as MapReduce, text data composed of words of variable-length strings are not suitable for maintaining a sufficient performance in hardware operations. By utilizing the characteristic of SWC that it counts the simple equivalences for two words, text data are stored in the flash storage after converting the words to fixed-length hash values. While the conversion is of advantage to the accelerator, irreversible transformation for storing intermediate data according to the platform is not unreasonable, since a database system requires indexing entries or some preprocessing [22].

SWC is executed by counting the times hash values calculated from the word to be counted are matched with values in a data stream read from the storage. To exploit parallelization further, we designed a hardware module that counts multiple words simultaneously. We adopted a 32-bit hash length to use for SWC without considering hash collisions, in an exploratory evaluation. The computation time does not include the time taken to calculate and transfer the hash values of the text data stored in the flash storage, since it is considered that during the time taken to store data the hash function can be applied while the data are accumulating.

SWC is achieved by only reading the data from the flash storage to DRAM. An acceleration module for DSPE-S, shown in Fig. 7, is implemented to compute SWC, the procedure of which is as follows.

- In preprocessing, hash values calculated from words produced by dividing the text data are stored in the flash storage on APX-880A.
- The host PC registers the objective hash values that are converted from the words to be counted in the local register array, and issues a command to transfer data from the storage to DRAM.
- The numbers of equivalences of the hash values in the data stream passing through the DSPE-S and the registered hash values are counted and accumulated at another location in the local register array.

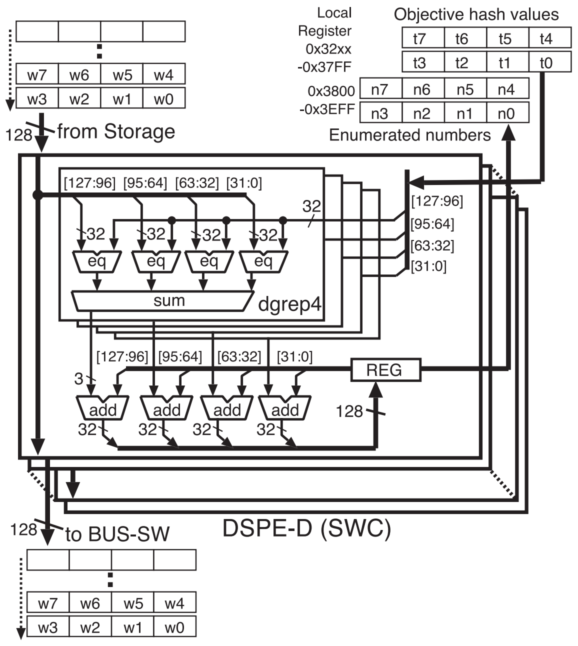


Fig. 7. The accelerator of Simple Word Counting

- The host PC reads the accumulated number from the local register array after the completion of the transfer of all data in the storage.

A hardware module for SWC consists of multiple dgrep4w modules, which compare and count four hash values. The input signals of all the dgrep4w modules are connected to the output signal from the storage interface and divided into four 32-bit hash values. The results of the comparison are summed and added to the data in the local register array, which stores the number of the occurrences of the words.

The maximum number of words that can be counted is restricted by the number of entries in the local register array or the logic resources of the FPGA. The architecture can count $W = N \times 4$ words in parallel when N dgrep4w modules are implemented in DSPE-S. For parallel execution in multiple computing nodes, we implemented software written in C language to start execution, copy the result of SWC from the local register array to DRAM, and send a computing node via the optical network to gather results. We executed the program code by using the SSH command for all nodes; the results are aggregated by a node.

C. Resource Utilization

The SWC module described in Section V-B is implemented in VHDL language. Tab. IV shows the utilization of the logic resources for implementations with several numbers of dgrep4w. It was confirmed that all the implementations run SWC with multiple nodes correctly through configuration in the APX-880A.

According to the restriction of the address width for the local register array, the maximum number of W is 256.

TABLE IV
LOGIC RESOURCE UTILIZATION

N: No. of dgrep4w	Base	1	4	16	64	
W: No. of words	0	4	16	64	256	
Logic Utilization [%]	64	64	65	69	84	100
Combinational ALUTs	70360	71139	72468	78846	104350	182400
Dedicated logic registers	85464	85626	86398	89335	100418	182400
Total block memory bits[Kbits]	10393	10328	10328	10328	10336	14283
DSP block 18-bit elements	4	4	4	4	4	1288

The delays related to the SWC module, which consists of simple and parallel comparators and adders, are not caused by the critical path. Through compilations for hardware, it was confirmed that the source of the maximum delay is the interface of the PCI-Express bus, and that the hardware, including the SWC module, behaves well at an operating frequency of 125 MHz. As shown in Tab. IV, the resource utilization of ALUTs and registers increases almost linearly according to the number of dgrep4w modules.

D. Setup of Software-based Simple Word Counting

For the experiments, 100 files, each consisting of 1 GB of text data, were generated by using the RandomTextWriter function of Apache Hadoop. The Apache Hadoop system consists of four computing nodes, the specification and software environment of which are shown in Tab. I and Tab. III, respectively. Two parameters are configured for Hadoop: the number of data replications is set to 1, the default value is 3, and the directory for the HDFS is set in SSDs. Therefore, 100 GB text data are uniformly distributed on each SSD.

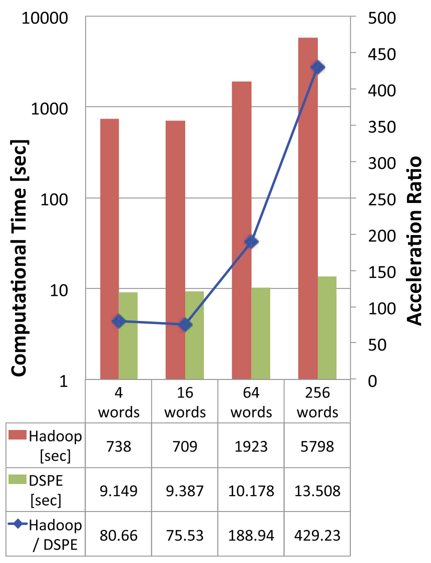
We can execute the 'grep' programs provided by Hadoop as an example of the following commands.

```
# Hadoop simple word count for four words
$ time Hadoop jar hadoop-mapreduce-examples.jar \
  grep text100G swc100G ”ell|kerykeion|knob|michigan”
```

E. Performance Evaluation

Fig. 8 shows the computational time taken for counting and the acceleration ratio of SWC as compared to the Hadoop-based execution. The SWC module with $N = 64$ is adopted to count a maximum of 256 words in parallel. The time lines of the CPU load, and the I/O for SSD and the I/O for the network of a representative computing node for each computing environment during the execution of simple word counting are shown in Fig. 9 and Fig. 10, respectively.

Counting by the SWC module is completed by reading the data in the storage to DRAM, according to the computing sequence described in Section V-B. The data stored in the storage become smaller than the raw text data, since all the words in the text data are transformed into 32-bit length hashes. Twenty-five files constituting 1 GB of text data are compressed into hash values of about 9.6 GB. Therefore, the computation time of DSPE-S in Fig. 8 is obtained from adding the time taken to read 9.6 GB data from the storage to DRAM, the I/O latencies to read and write data of the local register arrays, and the time taken to gather results into a computing node.



	4 words	16 words	64 words	256 words
🟥 Hadoop [sec]	738	709	1923	5798
🟩 DSPE [sec]	9.149	9.387	10.178	13.508
◆ Hadoop / DSPE	80.66	75.53	188.94	429.23

Fig. 8. Computational time and acceleration ratio for SWC

Fig. 8 shows the advantage of the tightly-coupled accelerator. It should be noted that the computational rate is observed to be almost 429 times faster than that of Hadoop. The source of the acceleration is not only the reduction in the text data size, but also the offloading of computation. It should be noted that, when the number of words to be counted is increased, the computation time taken by APX-880A does not increase drastically. Since the counting operations are spatially parallelized by multiple dgrep4w modules, only the number of accesses to the local register array before and after the data transfer operation influences the computation time. Fig. 9 also shows that the counting operation did not occupy CPU resources in the host PC, since the counting process was offloaded as data transfer from the storage to DRAM in APX-880A.

On the other hand, the computational time for Hadoop increased remarkably according to the number of words counted at a time. In particular, counting 256 words requires much more computational time than that of a smaller number of words. There are two reasons for the wide difference in the computing time of APX-880A and Hadoop. First, the Hadoop

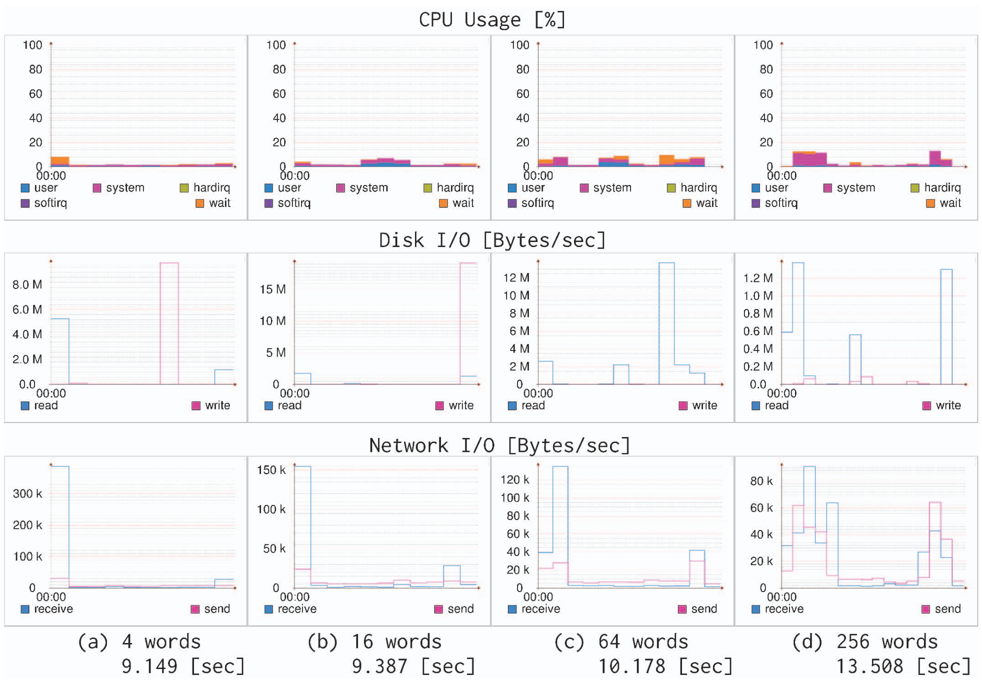


Fig. 9. Timeline for SWC in APX880A

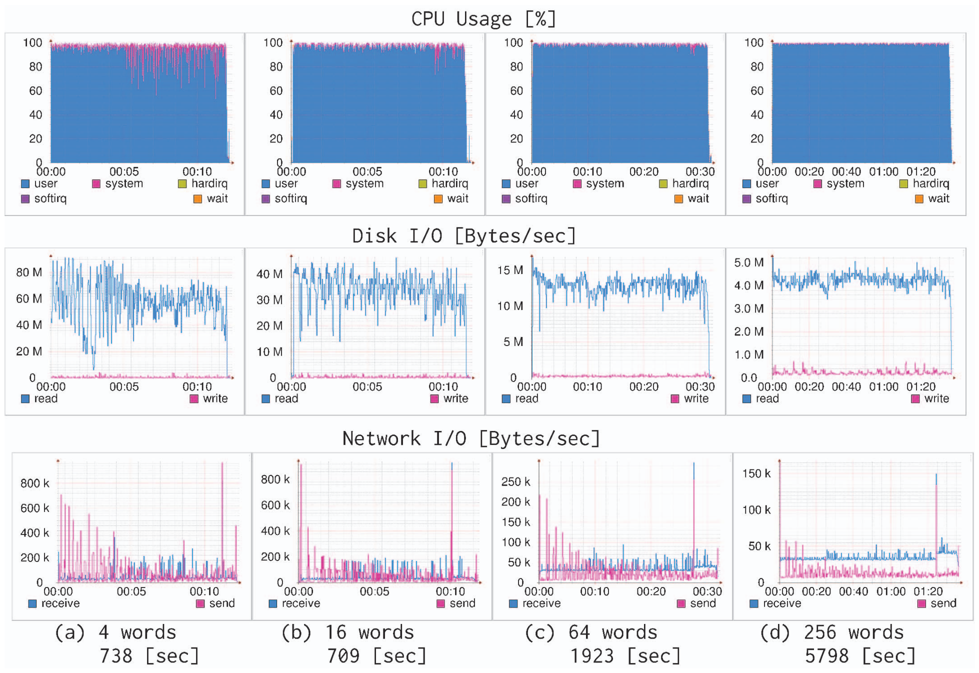


Fig. 10. Timeline for Apache Hadoop

framework forces a distortion of the algorithm such that it is divided for the mapper and reducer. According to the example program used in the evaluation, the mapper tasks divide the text data into words and the reducer tasks enumerate the number of matched words. The inadequate change of algorithm leads to a long computation time. The second reason is a problem related to treating text data: both parsing text data into lines and separating the lines into items are CPU-intensive tasks [4]. Fig. 10 shows that the process places the load only of counting words on the CPU. In addition, as HDFS is designed for large-scale distributed systems, the overhead causes a reduction in the throughput to access data to at most 90 MB/s, even when only four types of word are counted.

The summarized result of the evaluation is that the introduction of a tightly-coupled accelerator for a properly tuned algorithm such as SWC contributes to the acceleration by reducing the load on the host CPU, which leads to efficient energy consumption.

VI. Conclusion and Future Work

In this paper, an FPGA-based tightly coupled accelerator for Big Data processing was proposed. By implementing the accelerator with interfaces for storage and network, a data stream passed through the FPGA can be applied to the operation. Through the evaluation of an example application that uses simple word counting, it was confirmed that the load of the host PC is drastically reduced and the computation time is decreased 80.66 to 429 times as compared to that of software-based approaches. We intend to investigate other practical implementations of the accelerator utilizing both DSPE-S and DSPE-D and a network in data-intensive applications and to measure the energy efficiency of the computing system.

Acknowledgment

This work was partially supported by JSPS KAKENHI Grant Number 26330061, and Semiconductor Technology Academic Research Center (STARC).

References

[1] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation*, 2004, pp. 137–149.

[2] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, 2003, pp. 29–43.

[3] "Open Compute Project," http://www.opencompute.org/.

[4] T. Honjo and K. Oikawa, "Hardware acceleration of hadoop mapreduce," in *Proceedings of 2013 IEEE International Conference on Big Data*, 2013, pp. 118–124.

[5] B. He, W. Fang, Q. Luo, N. K. Govindaraju, and T. Wang, "Mars: A mapreduce framework on graphics processors," in *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, 2008, pp. 260–269.

[6] Y. Shan, B. Wang, J. Yan, Y. Wang, N. Xu, and H. Yang, "Fpmr: Mapreduce framework on fpga," in *Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, 2010, pp. 93–102.

[7] J. H. C. Yeung, C. C. Tsang, K. H. Tsoi, B. S. H. Kwan, C. C. C. Cheung, A. P. C. Chan, and P. H. W. Leong, "Map-reduce as a programming model for custom computing machines," in *Proceedings of the 2008 16th International Symposium on Field-Programmable Custom Computing Machines*, 2008, pp. 149–159.

[8] A. De, M. Gokhale, R. Gupta, and S. Swanson, "Minerva: Accelerating data analysis in next-generation ssds," in *Proceedings of the 2013 IEEE 21st Annual International Symposium on Field-Programmable Custom Computing Machines*, 2013, pp. 9–16.

[9] M. M. Rafique, B. Rose, A. R. Butt, and D. S. Nikolopoulos, "Cellmr: A framework for supporting mapreduce on asymmetric cell-based clusters," in *Proceedings of IEEE International Symposium on Parallel Distributed Processing*, May 2009, pp. 1–12.

[10] T. Araki, K. Narita, and H. Tamano, "Feliss: Flexible distributed computing framework with light-weight checkpointing," in *Proceedings of 2013 IEEE International Conference on Big Data*, 2013, pp. 143–149.

[11] E. Mazur, B. Li, Y. Diao, and P. Shenoy, "Towards scalable one-pass analytics using mapreduce," in *Proceedings of 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW)*, May 2011, pp. 1102–1111.

[12] S. Cho, C. Park, H. Oh, S. Kim, Y. Yi, and G. R. Ganger, "Active disk meets flash: A case for intelligent ssds," in *Proceedings of the 27th International ACM Conference on International Conference on Supercomputing*, 2013, pp. 91–102.

[13] S. Kim, H. Oh, C. Park, S. Cho, and S.-W. Lee, "Fast, energy efficient scan inside flash memory ssds," in *Second International Workshop on Accelerating Data Management Systems (ADMS)*, 2011, pp. 1–8.

[14] J. A. Delmerico, N. A. Byrnes, A. E. Bruno, M. D. Jones, S. M. Gallo, and V. Chaudhary, "Comparing the performance of clusters, hadoop, and active disks on microarray correlation computations," in *Proceedings of 2009 International Conference on High Performance Computing (HiPC)*, 2009, pp. 378–387.

[15] G. S. Davidson, K. W. Boyack, R. A. Zacharski, S. C. Helmreich, and J. R. Cowie, "Data-centric computing with the netezza architecture," *SANDIA REPORT*, pp. 1–24, Apr. 2006.

[16] J. E. Breeding, W. F. Jones, J. H. Reed, and T. Sangpaithoon, "Petlink ™; stream buffer: Using an fpga-based raid controller with solid-state drives to achieve lossless, high count-rate 64-bit coincidence event acquisition for 3-d pet," in *Proceedings of 2011 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*, 2011, pp. 3894–3900.

[17] L. Woods, J. Teubner, and G. Alonso, "Less watts, more performance: An intelligent storage engine for data appliances," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, 2013, pp. 1073–1076.

[18] A. Putnam, A. Caulfield, E. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. P. Gopal, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hormati, J.-Y. Kim, S. Lanka, J. Larus, E. Peterson, S. Pope, A. Smith, J. Thong, P. Y. Xiao, and D. Burger, "A recongurable fabric for accelerating large-scale datacenter services," in *41st Annual International Symposium on Computer Architecture (ISCA)*, 2014.

[19] S. R. Chalamalasetti, K. Lim, M. Wright, A. AuYoung, P. Ranganathan, and M. Margala, "An fpga memcached appliance," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA '13, 2013, pp. 245–254.

[20] S.-W. Jun, M. Liu, K. E. Fleming, and Arvind, "Scalable multi-access flash store for big data analytics," in *Proceedings of the 2014 ACM/SIGDA International Symposium on Field-programmable Gate Arrays*, 2014, pp. 55–64.

[21] AVAL DATA, "APX880," https://www.avaldata.co.jp/english_08/products/giga/tera_storage/apx880.html.

[22] S. Morishima and H. Matsutani, "Performance evaluations of graph database using cuda and openmp-compatible libraries," in *Proceedings of the 5th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies (HEART'14)*, 2014, pp. 77–82.