Intelligent SSD: A turbo for big data mining

Conference Paper · October 2013

DOI: 10.1145/2505515.2507847

5 authors, including:

Duck-Ho Bae
Hanyang University
29 PUBLICATIONS **145** CITATIONS
SEE PROFILE

Jin Hyung Kim
Korea Aerospace Research Institute
13 PUBLICATIONS **49** CITATIONS
SEE PROFILE

Sang-Wook Kim
Hanyang University
302 PUBLICATIONS **1,833** CITATIONS
SEE PROFILE

Chanik Park
Samsung
40 PUBLICATIONS **1,259** CITATIONS
SEE PROFILE

Intelligent SSD: A Turbo for Big Data Mining

Duck-Ho Bae
Hanyang University
Seoul, Korea
dhbae@agape.hanyang.ac.kr

Jin-Hyung Kim
Hanyang University
Seoul, Korea
kjhfreedom@agape.hanyang.ac.kr

Sang-Wook Kim*
Hanyang University
Seoul, Korea
wook@hanyang.ac.kr

Hyunok Oh
Hanyang University
Seoul, Korea
hoh@hanyang.ac.kr

Chanik Park
Sansung Electronics Co., Ltd.
Kyunggi-do, Korea
ci.park@samsung.com

ABSTRACT

This paper introduces the notion of intelligent SSDs. First, we present the design considerations of intelligent SSDs, and then examine their potential benefits under various settings in data mining applications.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*

Keywords

Intelligent SSD, big data mining, in storage processing

1. INTRODUCTION

Recently, big data analysis becomes more important to extract interesting patterns from a huge amount of data. Owing to the high I/O bandwidth and low access latency, solid state drives (SSDs) have been rapidly replacing hard disk drives (HDDs) [1].

SSD manufacturers try to overcome the physical limits of flash memory and to provide a higher I/O bandwidth for fast data access. However, such efforts incur a new performance bottleneck in data-intensive applications for big data analysis [2].

In HDD environment, low I/O bandwidth of HDDs is a main performance bottleneck in data-intensive applications since HDDs read data by *physically rotating a magnetic disk*. However, SSDs employ no mechanical devices and provide high internal I/O bandwidth thanks to the multi-way and multi-channel interleaving. As a result, in SSD environment, *the I/O bandwidth of host interface* becomes a new bottleneck in data-intensive applications [2].

In this paper, in order to solve the performance bottleneck of the host interface, we propose an approach called *intelligent SSD* (iSSD) in which a large volume of data are directly processed by processing units inside SSD and only a small resulting data is transferred to the host. With this approach, the host interface is no longer the performance bottleneck in data-intensive applications.

The internal I/O bandwidth in SSDs can easily scale up by increasing the multi-way and multi-channel interleaving [3]. So, the additional increase of the internal I/O bandwidth can be fully utilized in data processing. This allows us to enjoy the significant performance improvement in data-intensive applications. Moreover, as the data is processed in the place much closer to the data source itself, we can save both of the data transfer cost and energy [4].

Data mining is a typical example of data-intensive applications and has the following characteristics: (1) It frequently accesses the stored data during its execution; (2) It conducts relatively simple operations repeatedly such as scanning and filtering on the accessed data [5]. In this paper, we present design considerations and the potential benefits of the iSSD in terms of data mining applications.

[*]Corresponding author

2. INTELLIGENT SSD

Figure 1 shows the overall architecture of the SSD [3]. Typically, the SSD contains 8∼32 *channels*. In each channel, there are (1) 8∼16 NAND flash memory cells for storing data, (2) a *flash memory controller (FMC)* for managing all the flash memory cells in the channel, and (3) DRAM for reading/writing data from/to flash memory cells. All the channels are managed by firmware called *flash translation layer (FTL)*. *Embedded CPUs (core CPUs)* execute the FTL and DRAM saves FTL metadata in the SSD.

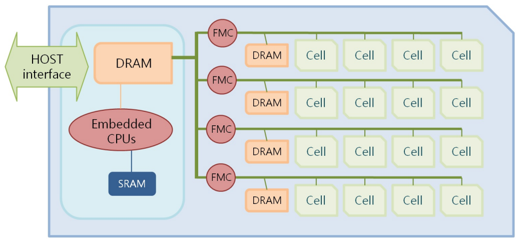


Figure 1: Architecture of an SSD.

Table 1 shows the I/O bandwidth of each part [3]. The internal I/O bandwidth of the SSD with 8-way and 16-channel is 6.4GB/s. This indicates that the internal bandwidth of the SSD is bound to that of host interface (PCIe x16: 6GB/s).

Table 1: I/O bandwidth of each part in SSDs

	I/O bandwidth
Host interface	SATA 3G: 250MB/s SAS 6G: 1GB/s PCIe x16: 6GB/s
NAND flash memory cell	SLC: 40MB/s MLC: 400MB/s
Main memory	DDR3: 6.4GB/s
Internal bandwidth of an SSD	8-way, 16-channel: 6.4GB/s

In order to fully exploit the *internal* I/O bandwidth for data processing, sufficient processing power is required *inside the iSSD*[1]; Otherwise, it could become a new performance bottleneck. We could add a cheap (while having relatively low performance) processor called *channel CPU* to each channel inside the iSSD.

The advantages with this approach are as follows: (1) Data can be processed over all channel CPUs *in parallel*. The size of the data processed in a channel CPU is in inverse proportion to the number of channels in the iSSD. Therefore, each cheap channel CPU sufficiently handles its data if we increase the number of channels inside the iSSD; (2) Adding a channel into the iSSD provides not only the expansion of the internal bandwidth but also the improvement of the inside processing power, thereby not burdening both the host interface and the host CPU; (3) A number of cheap channel CPUs are better than a single or a few high performance core CPUs inside an iSSD in the thermal aspect [7].

There have been several researches on processing data in storage devices [4], mainly focusing on HDDs. Since the HDD has a limitation to the expansion of the internal I/O bandwidth due to its physical characteristics (i.e., very high seek time and rotational latency), their approaches have not been practically adopted.

On the contrast, since the SSD employs no mechanical devices and provides *high scalability of internal I/O bandwidth*, the iSSD could be a good solution for dealing with large-scaled data mining applications. To support our opinion, recently several studies have explored the feasibility of in storage processing on SSDs [2].

3. DATA MINING IN ISSD

In this section, we analyze the execution costs of data mining applications. We formulate two cost models: Conventional data processing based on the host CPU (In-Host Processing, IHP) and data processing in the iSSD (In-Storage Processing, ISP). Figure 2 illustrates the execution steps of a data mining application in the IHP and the ISP, respectively.

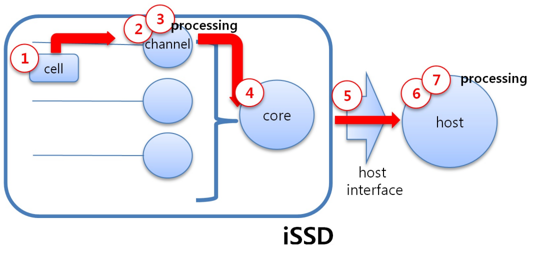
The execution costs of data mining applications are mainly composed of two parts: (1) data accessing and (2) data processing. Through analyzing these two costs, we identify the potential and new bottleneck of the ISP compared to the IHP.

3.1 Data access cost

We begin by defining some preliminaries to describe the cost to transfer d bytes in each path, similar to the derivation in [3].

• **Cell $\rightarrow$ channel:** Equation 1 represents the cost to read d bytes from flash memory cell to channel memory. Here, S_{page} is the size of a single NAND flash memory page, N_{way} is the number of ways in a channel, $t_{cell-read}$ is the time to read 1 page from flash memory cell, and t_{bus} is the elapsed time to load a page from a

[1]It is noteworthy that a special-purpose host interface such as *object storage device* [6] is needed for processing data inside the iSSD. However, such interface is orthogonal to our contributions, and thus is not mentioned further.



- IHP: $1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$
- ISP: $1 \rightarrow 2 \rightarrow 3$

Figure 2: Execution steps of data mining applications.

flash memory bus after the busy phase for a page read.

$$t_{cell \rightarrow ch} = \lceil \frac{d}{S_{page} \cdot N_{way}} \rceil t_{cell-read} + \lceil \frac{d}{S_{page}} \rceil t_{bus} \quad (1)$$

• **Channel $\rightarrow$ core:** For efficiency, data transfer between channel memory and core memory employs *the direct memory access (DMA) operation* and *the pipeline strategy* [3]. Equation 2 represents the cost to read d bytes from channel memory to core memory. Here, $t_{core-write}$ is the time to write d bytes to core memory.

$$t_{ch \rightarrow core} = max(t_{cell \rightarrow ch}, t_{core-write}) \quad (2)$$

• **Core $\rightarrow$ host:** Similar to $t_{ch \rightarrow core}$, data transfer between core memory and host memory also employs the DMA operation and the pipeline strategy. Equation 3 represents the cost to read d bytes from core memory to host memory. Here, $t_{host-write}$ is the time to write d bytes to host memory through the host interface and $t_{core-read}$ is the time to read d bytes from core memory.

$$t_{core \rightarrow host} = max(t_{host-write}, t_{core-read}) \quad (3)$$

The data access costs, derived from above preliminaries, in the IHP and in the ISP are as follows. Here, N_{ch} is the number of channels in the iSSD.

- *data_access_cost(IHP)* = $t_{ch \rightarrow core}/N_{ch} + t_{core \rightarrow host}$
- *data_access_cost(ISP)* = $t_{cell \rightarrow ch}/N_{ch}$

As shown in Figure 2, the data transfer path of the ISP is shorter than that of the IHP. Moreover, in the ISP, data does not move through the host interface which is a new bottleneck in iSSDs. So, the data access cost in the ISP is much lower than that of the IHP.

3.2 Data processing cost

We formulate the data processing cost of a target data mining application by *a unit of a function*. This is because, even with the functions belonging to one data mining application, their execution times show huge difference depending on their characteristics.

The data processing costs of the target data mining application in the IHP and in the ISP are formulated as follows. Note that, in the ISP, data mining applications are performed over all channel CPUs *in parallel*. Here, $|F|$ is the number of functions in the target data mining application and $t_{IHP(f_k)}$ ($t_{ISP(f_k)}$) is the time for performing function f_k in the host CPU (in the channel CPU).

- *data_processing_cost(IHP)* = $\sum_{k=1}^{|F|} t_{IHP(f_k)}$
- *data_processing_cost(ISP)* = $\sum_{k=1}^{|F|} t_{ISP(f_k)} \times 1/N_{ch}$

The data processing cost directly depends on *a CPU rate* of the place where the function is performed. So, the data processing cost in the ISP is much high compared with that of the IHP, even if the iSSD performs data mining applications with multiple channel CPUs in parallel.

In the ISP, some data mining applications require *global merge* that aggregates all local results produced by channel CPUs. The global merge adversely affects the performance in the ISP because it cannot be performed over channel CPUs in parallel. We assume that the global merge is conducted in the core CPU since it efficiently communicates with every channel CPU and has performance higher than channel CPUs.

The global merge cost in the ISP is as follows. Here, $t_{m(f_k)}$ is the time for performing global merge for function f_k in the core CPU and $R(f_k)$ is the local result size of function f_k.

- *merge_cost(ISP)* = $\sum_{k=1}^{|F|}\{t_{m(f_k)}+(t_{ch\rightarrow core}\cdot R(f_k)\cdot N_{ch})/d\}$

The global merge cost is influenced by (1) the size of a local result from each channel CPU, (2) the number of channels in the iSSD, and (3) the complexity of the global merge operation: I.e., the global merge cost increases when the size of a local result, the number of channels, and the complexity of the global merge operation increases.

3.3 Total execution cost

Equations 4 and 5 represent the total execution costs of the target data mining application in the IHP and in the ISP, respectively. With these equations, we can quantify the degree of the performance improvement in the ISP over the IHP. Here, $N(f_k)$ is input data size for function f_k.

- *total_cost(IHP) = data_access_cost(IHP)* $\cdot \sum_{k=1}^{|F|}(N(f_k)/d)$

  + *data_processing_cost(IHP)* (4)

- *total_cost(ISP) = data_access_cost(ISP)* $\cdot \sum_{k=1}^{|F|}(N(f_k)/d)$

  + *data_processing_cost(ISP) + merge_cost(ISP)* (5)

4. PERFORMANCE EVALUATION

4.1 Cost model validation

We first validate the cost models through three well-known data mining applications of k-means [5], PageRank [8], and Apriori [5]. Note that, Apriori requires the *global merge* to aggregate all the local results, each of which is the local frequency of a candidate itemset obtained by each channel CPU, in order to get its global frequency [5].

Table 2 represents the descriptions of primitive functions in each data mining application. Here, N is a total data size. To extract the number of cycles per instruction (CPI) and the number of instructions in each data mining application, we executed the application with V-Tune [2].

We can estimate the total execution cost of each data mining application in the IHP (ISP) to substitute the descriptions shown in Table 2 into Equation 4 (5). For the data access cost, we used the Samsung SSD datasheet[3]. To obtain the time for performing each function in the data processing cost, we used Equation 6.

[2]http://software.intel.com/en-us/intel-vtune-amplifier-xe/

[3]http://www.samsung.com/global/business/semiconductor/product/flash-ssd/catalogue

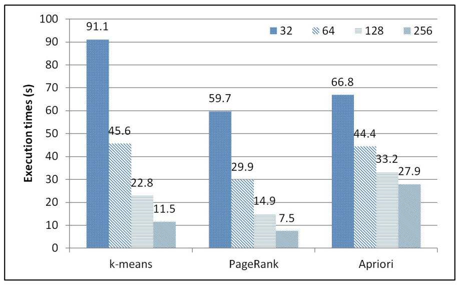
*Func. execution time=(CPI * #instructions) / CPU rate* (6)

For validating the correctness of our cost models, we compared the execution time of k-means, PageRank, and Apriori estimated by our IHP model with that obtained by executing them in a real machine. The results indicate a fairly high accuracy of 95∼99% with our models.

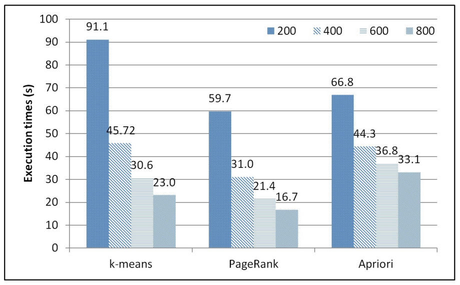
4.2 Results and analyses

Experiment 1. ISP with different iSSD settings: We changed the number of channels in the iSSD as 32, 64, 128, and 256, while setting the rates of a core CPU and a channel CPU as 400MHz and 200MHz, respectively. As shown in Figure 3(a), the performance of the ISP in all cases dramatically increases as the number of channels increases. In particular, it increases almost linearly with the number of channels for k-means and PageRank which do not have the stage of global merge.

We changed the channel CPU rates as 200MHz to 800MHz in step of 200MHz. We fixed the rate of a core CPU and the number of channels as 400MHz and 32, respectively. In Figure 3(b), we observe linear performance improvement with channel CPU rates in k-means and PageRank. However, in case of Apriori, the performance improves much less due to its global merge.



(a) With different numbers of channels.



(b) With different channel CPU rates.

Figure 3: Performance of the ISP with different iSSD settings.

Experiment 2. Comparison of ISP with IHP: For showing the potential benefits of the iSSD quantitatively, we compared the performance of IHP and ISP in k-means, PageRank, and Apriori. For the ISP, we considered two iSSD architectures: *iSSD_C* (typical setting in the current SSD) with a core CPU of 400MHz, 32 channels, and their CPUs of 200MHz and *iSSD_F* (setting in the future iSSD) with a core CPU of 800MHz, 256 channels, and their CPUs of 800MHz. For the IHP, we employed a host CPU of 2.5GHz, and also deployed both HDD and SSD for examining performance change with different storage media.

In Figure 4, the ISP in both iSSD architectures is shown to significantly improve the performance in all cases. Even with the current settings (iSSD_C), the ISP outperforms the IHP about 2∼3 times.

Table 2: Descriptions of data mining applications

	CPI	# of INS	Data access	Global merge required
k-means				
create_initial_center	0.60	15939.00	-	X
calculate_distance	0.75	3553.81	Read N bytes	X
clustering	2.98	3.61	-	X
insert_entry	0.99	212.63	Write N bytes	X
update_MSE	0.62	86.22	-	X
calculate_new_center	0.75	1268.35	-	X
PageRank				
initlalize_ranking_vector	0.56	199.41	Read N bytes	X
matrix_multiplication	0.60	18.12	-	X
add_restart_vector	2.22	3.80	Write N bytes	X
Apriori				
count_frequency	0.50	10.02	Read N bytes	X
check_MinSupport	0.56	7.32	-	O
create_candidate_itemsets	0.51	42.89	-	O
create_frequent_itemsets	1.57	49.58	Write N bytes	O

With the future settings (iSSD_F), the ISP dramatically improves the performance of the IHP up to 83 times.

In case of Apriori, the ISP(iSSD_F) shows insignificant performance improvement compared with the ISP(iSSD_C) due to its global merge. Such findings are vividly illustrated in Figure 5.

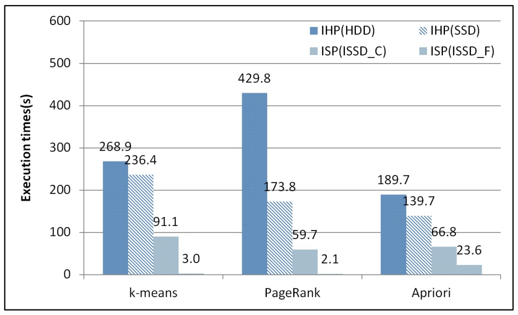


Figure 4: Performance comparison of ISP with IHP.

Figure 5 shows the breakdown result of the ISP cost. The global merge costs of Apriori in both ISP(iSSD_C) and ISP(iSSD_F) are almost same. This is because as the number of channels increases, the number of local results to be merged in the core CPU also increases. So, in Apriori, the data processing in the core CPU becomes new performance bottleneck. As a possible solution to solve the performance bottleneck in the core CPU, we can consider the method to conduct the global merge in the host CPU which has much higher performance than the core CPU.

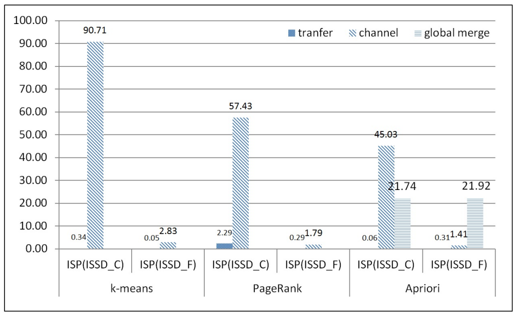


Figure 5: Breakdown of the ISP cost.

5. CONCLUSIONS

In this paper, we introduced the iSSD as a turbo for big data mining. We first presented design considerations for the iSSD. The iSSD should have (1) enlarging internal I/O bandwidth and (2) sufficient inside processing power. We then formulated the execution costs of data mining applications for predicting their performance in the iSSD. In the ISP, the execution cost is composed of (1) data access cost, (2) data processing cost for accessed data, and (3) global merge cost for aggregating all local results in core CPUs. Finally, we validated the cost models and showed the potential of the ISP through a series of experiments. The results show that our iSSD-based processing achieves speed up by up to 83 times compared with current host-CPU based processing.

6. ACKNOWLEDGEMENT

This work was financially supported by (1) Semiconductor Industry Collaborative Project between Hanyang University and Samsung Electronics Co. Ltd, (2) the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. NRF-2012047724), and (3) Seoul Creative Human Development Program (HM120006).

7. REFERENCES

[1] D. Bae, J. Chang, and S. Kim, "An Efficient Method for Record Management in Flash Memory Environment," *Journal of Systems Architecture*, pp. 221-232, 2012.

[2] J. Do et al., "Query Processing on Smart SSDs: Opportunities and Challenges," *ACM SIGMOD*, 2013.

[3] S. Kim et al., "Fast, Energy Efficient Scan Inside Flash Memory SSD," *ADMS*, 2011.

[4] E. Riedel, G. Gibson, and C. Faloutsos, "Active Storage For Large-Scale Data Mining and Multimedia," *VLDB*, pp. 62-73, 1998.

[5] J. Han and M. Kamber, *Data Mining: Concepts and Techniques(2nd Edition)*, Morgan Kaufmann, 2006.

[6] D. Du et al., "Experiences Building an Object-Based System based on the OSD T-10 Standard," *IEEE MSST*, 2006.

[7] V. Reddi et al., "Web Search using Small Cores: Quantifying and Mitigating the Price of Efficiency," *ISCA*, pp. 314-325, 2010.

[8] L. Page et al., The PageRank Citation Ranking: Bringing Order to the Web, Technical Report, Stanford University, 1999.