How Much Computation Power do you need for Near-Data Processing in Cloud?

Namhyung Kim[†‡], Jeongseob Ahn[‡], Sungpack Hong[‡], Hassan Chafi[‡], and Kiyoung Choi[†]
[†]Seoul National University [‡]Oracle Labs
nhkim@dal.snu.ac.kr {jeongseob.ahn, sungpack.hong, hassan.chafi}@oracle.com kchoi@snu.ac.kr

***Abstract*—In modern cloud server systems, the big data application domain is one of the most important ones these days. However, as a large amount of data has to be transferred between storage unit and computing unit, the conventional server systems failed to fully utilize their resources. Near-data processing has been proposed to mitigate this inefficiency. Due to the reduced data transfer and its scalability, utilizing near-data processing can perform better than servers with high-end computing units. It can reduce data transfer by offloading computation to the embedded CPU in the storage device. In this work, based on the evaluation and analysis, we demonstrate advantages and disadvantages of existing server systems with near-data processing and show the potential of them with an alternative design. We also demonstrate that embedded CPUs are not enough to run applications such as big data applications. Instead, our evaluations show that computing units with better energy efficiency such as embedded GPU or ASIC can be a promising alternative that better exploits the advantage of near-data processing.**

I. Introduction

Big Data [1], data analysis, and Cloud are three outstanding trends in IT industry that reinforce one another. The proliferation of Big Data or massive data repositories has played a crucial role in recent improvements in data analysis methodologies, including deep learning and graph analysis. The success stories of data-driven applications motivate people to accumulate even larger data sets so that they can extract even more valuable information. On the other hand, a modern Cloud environment often offers sufficient amount of storage capacity and computation power for Big Data-driven applications within affordable price range. The growing number of those applications, in turn, solicits even more storage and computation, which Cloud business can thrive upon.

While a typical data-driven application requires both large amount of data and significant amount of computation, the portion of the two differs from one application to another. Cloud, on the other hand, works at scale and aims to address various characteristics of user applications in elastic manners. Therefore, Cloud providers tend to deploy different classes of servers for different purposes in their data center, e.g., storage servers and compute servers. Storage servers are built with a large number of disk bays, but with relatively wimpy CPUs, in order to maximize data density per rack. Compute servers, to the contrary, are built with beefy CPUs (or GPUs), larger DRAMs, and high-bandwidth SSDs to provide high computation throughput. Consequently, a Cloud provider can keep deploying different number of compute servers and storage servers, based on the aggregated requirements of all the customers.

However, this elastic design of Cloud infrastructure creates another challenge for data-driven applications – the large amount of data has to move from storage server into compute server before it can be possessed. Near-Data Processing (NDP) is one answer to this challenge. The idea is to move computation near the place where the data resides and to perform computation there, instead of moving data around. In fact, there are many proposals/systems that applies NDP principle in different layers. For instance, modern enterprise database systems (e.g., Oracle Exadata) pushes down certain operations (e.g., scan) to storage layer to reduce the amount of data movement in query processing. Intelligent storage devices [2]–[5] go one step further, as they push down computation into the storage device (i.e., SSD), exploiting embedded computing units in the device controller. The effectiveness of NDP proposals, however, depend not only on available computation and data access bandwidth of the proposed architectures but also on the characteristics of applications.

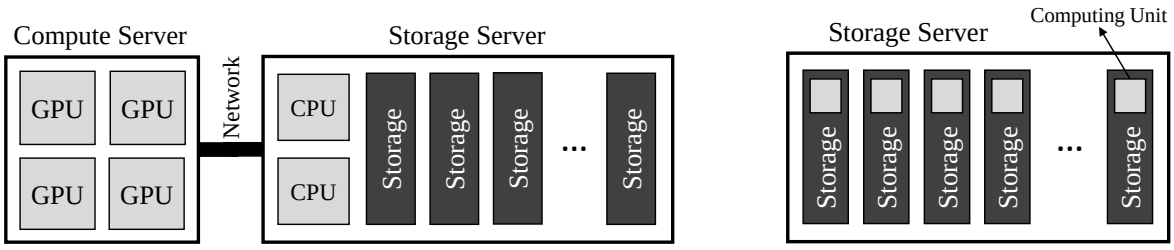
In this study, we explore the potential benefits of NDP for executing data applications in Cloud environments. Specifically, we make the following contributions in this paper.

- We explain pros and cons of NDP proposals with a simple roofline model, as well as characterize and validate a few representative data-driven applications for the model.
- Our model shows that the intelligent storage approach has great potentials for streaming applications, as it can exploit all the bandwidth of the disks as the data size scales up.
- We also observe that current intelligent storage proposals, however, do not reach their full potential due to the inferior compute-per-power of embedded CPU. Rather, we propose that using more power-efficient units such as embedded GPU or ASIC would make this approach viable even for fairly compute intensive applications.

II. Background and Motivation

A. Near-Data Processing

The concept of the near-data processing was first proposed a few decades ago [6], [7]. In recent few years, it is gaining attention again through emerging memory architectures (e.g., 3D stacked memory, Hybrid Memory Cube, and SSD). Since NDP moves computation to inside the memory or storage layer



(a) Conventional server system with GPUs

(b) Near-data processing

Fig. 1. System configurations.

where the data is stored, it can significantly save the cost of data movement. Instead of transferring the whole data, it only transfers subset or pre-processed data. In this study, we focus on NDP systems enabled in the storage layer. For instance, in SSDs, general purpose embedded CPUs are adopted as a controller to run firmware (e.g., flash translation layer) in many cases. Therefore, we can exploit the computing units for NDP.

With NDP, since most of the data stays inside the storage and does not move outside of it, NDP can utilize the internal bandwidth of the storage which is higher than the external bandwidth. For example, the internal bandwidth of recent high speed SSD is 30% higher than the external bandwidth (e.g., PCIe 3.0 x4) [4]. Furthermore, the internal bandwidth of SSD will be improved rapidly over time with its internal parallelism [3], [5].

There are a bunch of previous work of NDP with SSDs. Tiwari et al. [2] and Cho et al. [3] use analytic model to estimate benefits of NDP running some data-intensive applications and estimate its performance and energy consumption. Do et al. [5] and Gu et al. [4] evaluate NDP with real NDP-enabled SSD and show significant performance improvement and energy saving through the use of NDP.

B. State-of-the-art Server System in Cloud

Fig. 1a shows the state-of-the-art GPU-enabled system in cloud servers [8]. It is configured with host servers, compute servers, and storage servers. A compute server consists of a bunch of high-end GPU devices [9] and utilizes them as an accelerator by exploiting their high parallelism. A storage server has tens of storage devices and provides a large capacity of storage with high density to clouds [10]. It also has few CPUs for computing capability. These components in the system are connected to each other through the network such as Infiniband or Ethernet.

To accelerate an application with GPUs, the data accessed by applications needs to be copied to a GPU server ahead of the kernel execution. Therefore, the data-intensive workloads like data mining applications may suffer from frequent and large data transfer. Particularly, some streaming applications without data reuse need to load every data they access from the storage. To reduce the amount of data movement between storage server and compute server, the CPUs in the storage server can be utilized.

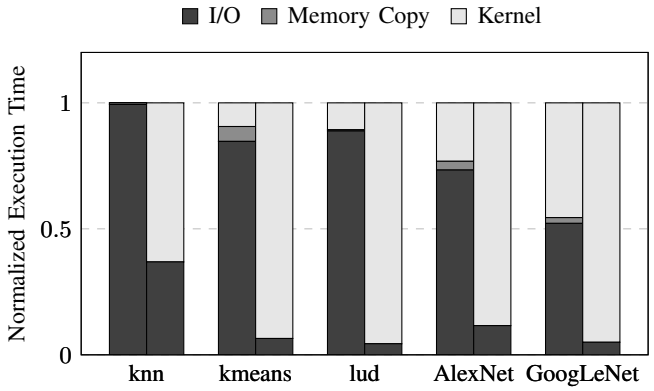


Fig. 2. Relative execution time breakdown. High-end GPU (left) and embedded CPU (right).

As an alternative way, we can build NDP-enabled servers shown in Fig. 1b. Each storage device in the server is configured with a computing unit and memory (e.g., flash memory) and the NDP-enabled server can be connected to other compute or host servers like Fig. 1a. Such NDP servers can be beneficial to reduce the amount of data movement by processing the data stored in the same storage device. In addition, since the computation resource resides in every storage device, the total computing power and the internal bandwidth of NDP can be scaled as the number of attached storage devices increases. If applications offloaded to the storage are distributed and processed only with local data, the performance can be improved linearly to the number of storage devices.

C. Limitation of Existing Systems

There are two main concerns in designing scalable server architecture. First, the amount of data processed by emerging applications is growing faster. It turns out that the amount of data needed to be loaded from the storage is also increasing. Second, the computing capability of the server system is also growing faster with accelerators like GPUs, but still there is a lack of the network bandwidth between the storage and the compute server. Although we can take advantage of the prior NDP systems [4] to reduce the data movement by moving computation inside the storage devices, the effectiveness depends on available computation bandwidth.

We run experiments to figure out how much computation is required on a couple of applications. Fig. 2 shows execution

time breakdown of selected applications with two different computing units, the high-end GPU and embedded CPU. To precisely compare the computation bandwidth between them, we assume that each computing unit is attached to the same type of storage and network. In the high-end GPU setting, we can see that a large portion of time (79.7%) is spent on the data transfer since the data transfer rate does not meet the the data processing rate. It turns out that the high-end GPU becomes idle frequently. On the other hand, with an embedded CPU, 87.1% of the time is spent on computation because of the relatively low performance of embedded CPU. In such a case, the computation capability is a dominant factor deciding the performance.

As can be seen here, previous two state-of-the-art systems are not balanced between computing power and bandwidth. The high-performance GPU server system is bounded by I/O bandwidth and the system with NDP is bounded by its low computing power. Therefore, both of server systems are very inefficient and this imbalance must be addressed.

III. Evaluation: Near-Data Processing for Cloud Environment

In this section, we take the approach of limiting factor analysis in discussing the effectiveness of Near-Data Processing for a Cloud environment. We assume that a data-driven application is applied to *infinitely large* amount of data that is stored in numerous storage devices across the Cloud. Then we analyze the maximally achievable application throughput for different system configurations under the same electric power budget for computation.

A. Roofline Model

For the sake of discussion, we adopt the roofline model [11], as it illustrates the impact of the limiting factor between computation and communication very well. Fig. 3 is an example of a typical roofline plot. The x-axis represents computation intensity of the target application as computation per byte (i.e. FLOPs/Byte). That is, an application is characterized as a single value (computation per byte), assuming linear scaling of computation with data. On the other hand, the y-axis is the achieved performance (or throughput) of the application, as FLOPs per second. Therefore, for example, if two different implementations of the same application are put in the plot, a *faster* implementation would be placed on top of the other, since they share the same innate computation intensity of the application, as shown in the figure.

More importantly, the roofline model clearly indicates the maximally achievable performance of any application, from two key characteristics of the execution environment: communication bandwidth and computation throughput. In the plot, this performance limit is rendered into two segments of line – slanted region and flat region. The slanted region is for the application that has low computation intensity, since their maximum performance is bounded by the communication bandwidth of the execution environment. In the plot, the slope of the slanted region indicates the communication bandwidth.

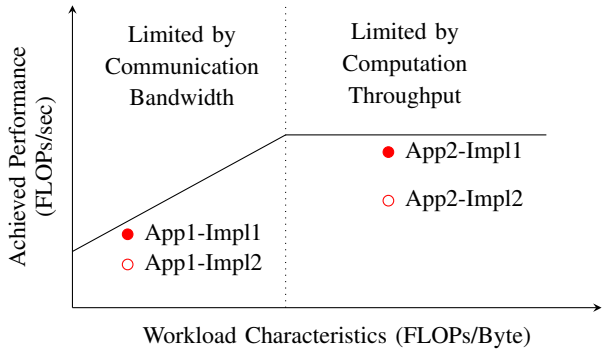


Fig. 3. Typical Roofline model.

In contrast, the flat region is for the computation-intensive applications where their maximum performance is limited by computation throughput of the execution environment. This limit is indicated as flat line in the region.

B. Workload Characteristics

For our discussion, we choose seven data-driven applications and get their computation and communication characteristics. We use two from traditional data mining applications (knn, kmeans), one from computational analysis (i.e., linear algebra) applications (lud), and two from convolutional neural network (CNN) applications (Alexnet [12] and GoogLeNet [13]). Knn, kmeans, and lud are from Rodinia benchmark suite [14] and Alexnet and GoogLeNet are from Caffe [15]. For each CNN application, we took its training and inference as two separate applications since they have different characteristics.

The descriptions of selected workloads are summarized in Table I. Note that the first five applications have streaming or embarrassingly parallel communication pattern. On the other hand, the CNN training applications require both streaming of image data and synchronization of updating weight values. Here we assume simple synchronized approach for parallel/distributed learning, where each learner communicates with a central weight server at every batch.

In addition, we measured the computation per byte using the GPU implementations of these applications. That is, we profiled GPU execution to obtain the number of floating point operations of main kernels in these applications and divided the number by the size of necessary data movement. In the table, CNN inference has higher computation intensity than CNN training, because we assumed that the whole weight values are synchronously updated at every batch; the size of weight is larger than input data.

C. System Configurations

We now explain the different system configurations for our comparisons. As a reminder, we take the approach of limiting factor analysis. That is, we assume that a data-driven application is applied to *infinitely large* amount of data that is stored in numerous storage devices across the Cloud. We analyze the maximally achievable application throughput for

TABLE I
SUMMARY OF WORKLOADS

Workload	Description	Compute Intensity (FLOPs/Byte)	Data Access Pattern	Source
knn	k-nearest neighbor	0.102	Streaming object data	Rodinia
kmeans	Kmeans	3.840	Streaming object data	Rodinia
lud	LU Decomposition	782.1	Streaming matrix data	Rodinia
AlexNet (inference)	CNN inference	13333	Streaming image	Caffe
GoogLeNet (inference)	CNN inference	16667	Streaming image	Caffe
AlexNet (training)	CNN training	395.7	Streaming image / synchronize weight	Caffe
GoogLeNet (training)	CNN training	8120	Streaming image / synchronize weight	Caffe

different system configurations that have similar electric power budget for computation.

For this purpose, we first chose computation components for different grades: high-end GPU, high-end CPU, embedded CPU and embedded GPU. Our choices and their characteristics are summarized in Table II, where we tried to choose fairly representative ones, for instance by considering popularity. Computation throughput and TDP are from the specification documents of the computation component.

Now, we introduce four system configurations that we consider for Cloud computation. The first is to use a powerful compute-server, as depicted in Fig. 1a. Specifically, we assume that the compute-server is equipped with four high-end GPUs. In this configuration, however, the compute sever needs to stream in the data from storage servers (or devices) before it can proceed with the computation. Therefore, the bottleneck communication bandwidth (for streaming applications) would be the bandwidth of the network connected to this server; we assume that an Infiniband network is available with 3.2GB/s bandwidth. This configuration, referred as Compute-Server-GPU, is the baseline for our comparison.

The second configuration (Storage-Server-CPU) is for more traditional push-down approach where the computation is done by the CPUs in the storage server (Fig. 1a). Here, we assume that a storage server has two high-end CPUs and connected to a lot of (e.g., 32) storage devices through eight channels of SAS disk interface. Therefore, for streaming application, each storage server can get the aggregated bandwidth of these channels which we count as 12.0 GB/s. We compare the electric power of computing component and presume that four storage servers are equivalent to a single compute server with high-end GPUs.

The third and fourth configurations (NDP-eCPU and NDP-eGPU) represent the cases of NDP. As depicted in Fig. 1b, these configurations assume that each storage device (e.g, SSD) is equipped with a computing component so that it can directly execute the given application on the data partition stored in the device. As for the streaming bandwidth, we assume that these devices can exploit the internal bandwidth which is somewhat higher than external bandwidth, amounting to 1.3x of PCIe, as discussed in a previous study [4]. Finally, from TDP of computing components, we presume that a single compute server is equivalent to 128 storage devices with embedded CPU and 64 storage devices with embedded GPU.

TABLE II
SPECIFICATIONS OF COMPUTATION COMPONENTS.

	Class	Compute (GFLOPs/sec)	TDP (watt)
4x Cortex A57	Embedded CPU	55	8
Tegra X1	Embedded GPU	500	15
Xeon 2699v4	Mid-end CPU	1970	145
Titan X	High-end GPU	7000	250

TABLE III
SYSTEM SPECIFICATION

Configuration Name	Computation Component	Num. Units	Stream BW (per-unit)
Compute-Server-GPU	High-end GPU (x4)	x1	3.2 GB/s
Storage-Server-CPU	High-end CPU (x2)	x4	12.0 GB/s
NDP-eCPU	Embedded CPU (x1)	x128	4.55 GB/s
NDP-eGPU	Embedded GPU (x1)	x64	4.55 GB/s

D. Analysis: Streaming Applications

Now we compare different configurations using the roofline model for streaming applications.

The roofline models of all configurations are shown in Fig. 4a where each configuration draws a different line from its own available computation and bandwidth. Since these plots are for streaming applications, the available communication bandwidth in each configuration is simply **(# units in configuration) x (available bandwidth per unit)** in Table III, which dominates the *slanted* regions of the plot.

Similarly, for *flat* regions of the plot, the available computation throughput is again determined by **(# units in configuration) x (# computing components per unit) x (computation throughput per component)**, because we are assuming streaming applications.

Fig. 4b and Fig. 4c are a zoom-ins of slanted region and flat region respectively of Fig. 4a with indications of corresponding application workloads in Table I. For instance, knn application whose computation intensity is about 10^{-1} is shown in Fig. 4b, while AlexNet inference is shown in Fig. 4c with intensity of about 10^4.

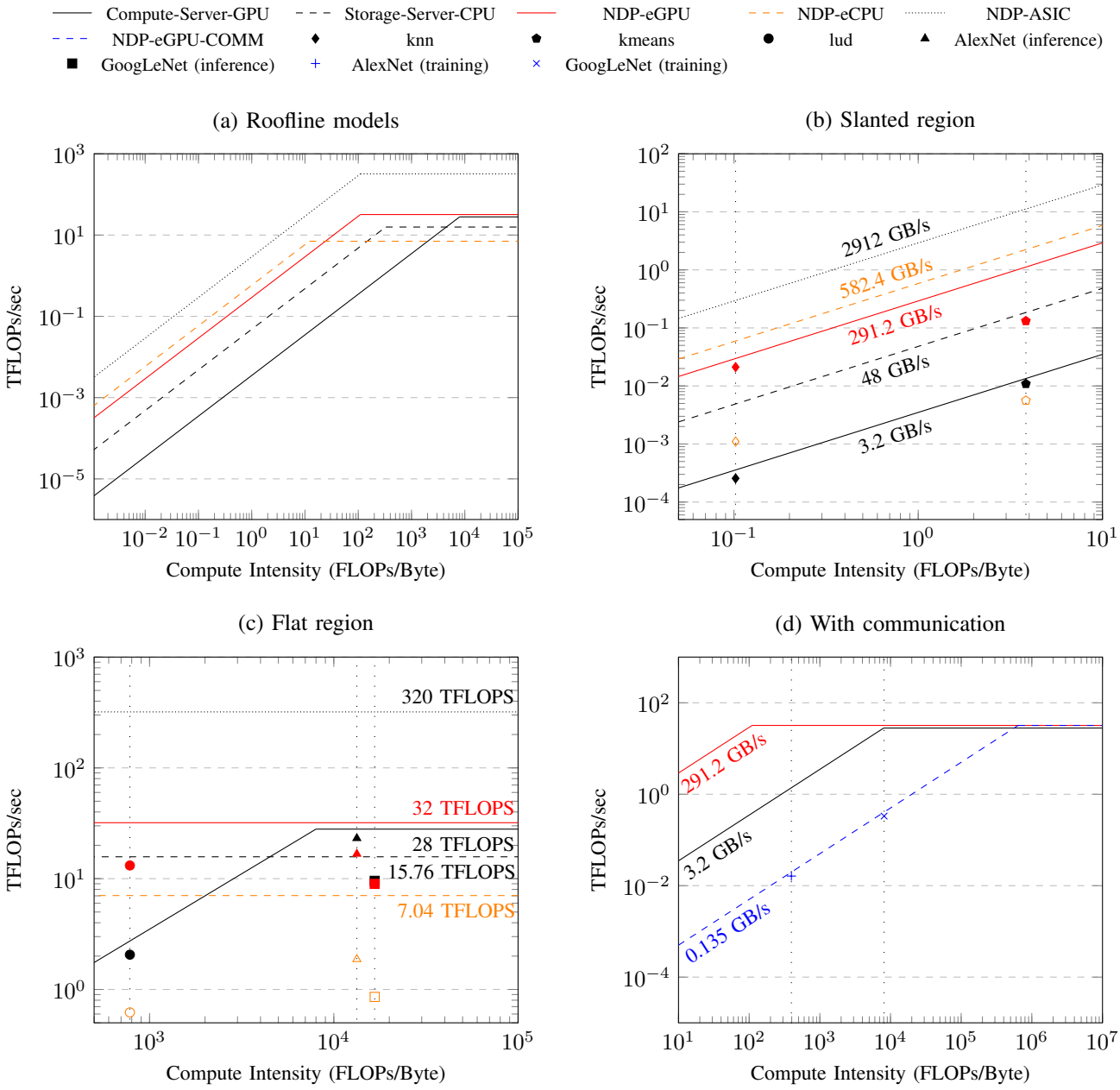


Fig. 4. Roofline models.

The following observations can be made naturally from these plots. (a) For applications with low-computation intensity, spreading out the data into as many devices as possible is always the best. Consequently, among four original configurations, NDP-eCPU performs the best, since the lowest TDP of embedded CPU allowed the largest number of units. (b) However, embedded CPU also has the worst performance-per-power. Therefore the configuration hits the flat region earlier than other configurations and the throughput saturates faster as the computation intensity grows (e.g., lud). (c) To the contrary, Compute-Server-GPU configuration performs least, because the performance is bottlenecked by the network communication bandwidth until the computation intensity of target applications grow significantly. (d) Finally, NDP-eGPU configuration is very interesting. This configuration performs well not only for low computation applications (due to its low TDP), but also for high computation workloads (due it its high performance-per-power). Consequently, this configuration works better than Compute-Server-GPU configuration even for CNN inference applications.

Finally, we added the plot for an extra configuration in above plots, namely NDP-ASIC. This configuration represents the case of using an ASIC for the given (family of) application, which could bring the same amount of computation as embedded GPU but with 10x lower power. For example, neural network accelerators (e.g., PuDianNao [16], Eyeriss [17], and

TPU [18]) are proposing similar performance and energy efficiency over GPUs. Therefore in this configuration, according to our model, the computation can be spread out to 10x larger number of devices, which yields 10x more bandwidth than NDP-eGPU for *slanted* region. Also, since each of the ASIC component gives the same amount of computation as one embedded GPU, the aggregated computation throughput is again 10x than NDP-eGPU.

E. Analysis: Non-Streaming Applications

The analysis in the previous section is, however, for streaming applications where each unit can perform its own computation without communicating with others a lot. However, not all data-driven applications are simple streaming ones. In this section, we analyze the case of CNN training applications, as examples of non-streaming applications.

As discussed in previous subsections, the CNN training applications not only stream the image data, but also need to synchronize weight value updates. In this analysis, we assume simple methodology of distributed learning where each learner communicate with weight server at the end of each batch processing.

Under this communication pattern, we now need to adjust the roofline model of our execution configurations. The flat regions are intact since the aggregated computation bandwidth, determined by computation throughput per each device, stays the same. However, for slanted region, the effective bandwidth has to be re-adjusted, as we can no longer say the internal storage bandwidth is the bottleneck. Rather, the new bottleneck becomes communication with the weight server. Because each unit must communicate with the server, we presume that the effective bandwidth is **(Network Bandwidth of Weight Server) / (# Units in Configuration)**. We use Infiniband bandwidth for the network bandwidth.

Fig. 4d depicts the new roofline plots for this communication assumption. Although, NDP-eGPU-COMM configuration still has high value for the aggregated computation throughput, the effective bandwidth is too low; it requires very high computation density for NDP-eGPU-COMM to become faster than Compute-Server-GPU again.

IV. Conclusion

In this paper, we analyzed and evaluated alternative designs of near-data processing with higher computation power than embedded CPUs. Based on our observation, none of the existing server systems can run data mining applications efficiently. Through higher internal bandwidth of storage devices and computation bandwidth, NDP with embedded GPU can outperform than conventional server systems with high-end GPUs. We also provided insights about which NDP design is the best for each application. Among the systems we compared, NDP with high computation units was the best in general in terms of both performance and energy efficiency. However, the benefit of NDP was limited to the applications that can be distributed without communication between NDP devices. For the applications with communications, NDP systems were not beneficial at all because it cannot take advantages of improved bandwidth anymore.

References

[1] P. Vagata and K. Wilfong. (2014) Scaling the Facebook data warehouse to 300 PB. https://code.facebook.com/posts/229861827208629/scaling-the-facebook-data-warehouse-to-300-pb/

[2] D. Tiwari, S. Boboila, S. Vazhkudai, Y. Kim, X. Ma, P. Desnoyers, and Y. Solihin, "Active Flash: Towards energy-efficient, in-situ data analytics on extreme-scale machines," in *Proceedings of the Conference on File and Storage Technologies*, 2013.

[3] S. Cho, C. Park, H. Oh, S. Kim, Y. Yi, and G. Ganger, "Active disk meets flash: a case for intelligent SSDs." in *Proceedings of the International Conference on Supercomputing*, 2013.

[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," in *Proceedings of the International Symposium on Computer Architecture*, 2016.

[5] J. Do, Y.-S. Kee, J. M. Patel, C. Park, K. Park, and D. J. DeWitt, "Query processing on smart SSDs: Opportunities and challenges," in *Proceedings of the International Conference on Management of Data*, 2013.

[6] E. Riedel, G. A. Gibson, and C. Faloutsos, "Active storage for large-scale data mining and multimedia," in *Proceedings of the International Conference on Very Large Data Bases*, 1998.

[7] K. Keeton, D. A. Patterson, and J. M. Hellerstein, "A case for intelligent disks (IDISKs)," *ACM SIGMOD Record*, vol. 27, pp. 42–52, Sep. 1998.

[8] A. G. Murillo. (2017) The end-to-end refresh of our server hardware fleet. https://code.facebook.com/posts/1241554625959357/the-end-to-end-refresh-of-our-server-hardware-fleet/

[9] K. Lee. (2017) Introducing Big Basin: Our next-generation AI hardware. https://code.facebook.com/posts/1835166200089399/introducing-big-basin-our-next-generation-ai-hardware/

[10] J. Adrian. (2017) Introducing Bryce Canyon: Our next-generation storage platform. https://code.facebook.com/posts/1869788206569924/introducing-bryce-canyon-our-next-generation-storage-platform/

[11] S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Communications of the ACM*, vol. 52, pp. 65–76, Apr. 2009.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the Advances in Neural Information Processing Systems*, 2012.

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *arXiv:1409.4842*, 2014.

[14] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *Proceedings of the International Symposium on Workload Characterization*, 2009.

[15] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv:1408.5093*, 2014.

[16] D. Liu, T. Chen, S. Liu, J. Zhou, S. Zhou, O. Teman, X. Feng, X. Zhou, and Y. Chen, "PuDianNao: A polyvalent machine learning accelerator," in *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, 2015.

[17] Y. H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *Proceedings of the International Symposium on Computer Architecture*, 2016.

[18] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a Tensor Processing Unit™," in *Proceedings of the International Symposium on Computer Architecture*, 2017.