TASK-DECOMPOSITION BASED ANOMALY DETECTION OF MASSIVE AND HIGH-VOLATILITY SESSION DATA ON ACADEMIC BACKBONE NETWORK

Ruo Ando[1], Youki Kadobayashi[2] and Hiroki Takakura[1]

[1]National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku,
Tokyo 101-8430 Japan
[2]Nara Institute of Science and Technology, Graduate School of Science and Technology,
8916-5 Takayama, Ikoma, Nara 630-0192, Japan

ABSTRACT

The Science Information Network (SINET) is a Japanese academic backbone network for more than 800 universities and research institutions. The characteristic of SINET traffic is that it is enormous and highly variable. In this paper, we present a task-decomposition based anomaly detection of massive and highvolatility session data of SINET. Three main features are discussed: Tash scheduling, Traffic discrimination, and Histogramming. We adopt a task-decomposition based dynamic scheduling method to handle the massive session data stream of SINET. In the experiment, we have analysed SINET traffic from 2/27 to 3/8 and detect some anomalies by LSTM based time-series data processing.

KEYWORDS

Anomaly detection, high-volatility traffic, task decomposition, dynamic scheduling, LSTM

1. INTRODUCTION

The Science Information Network (SINET) which the academic backbone network handles Internet traffic of more than 800 universities in Japan. Session data of SINET is increased drastically, which imposes a significant burden on network administrators. The main challenge in this paper is to handle massive and high volatility traffic data.

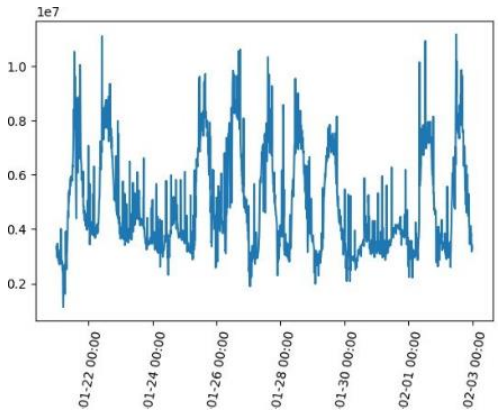

Figure 1. SINET ingoing traffic

DOI : 10.5121/ijdps.2021.12201

First of all, we report the temporal pattern of the SINET ingoing/outgoing traffic patterns. Figures 1 and 2 show the traffic volume of session data. Session data is grouped into one-hour frame bins. In ingoing traffic, we observe a clear diurnal pattern from 01/25/2021 to 01/29/2012. This period is the weekday of the academic campus. On the other hand, 01/30/2021 and 01/31/2021 are weekends, which results in a decrease of traffic and plateau.

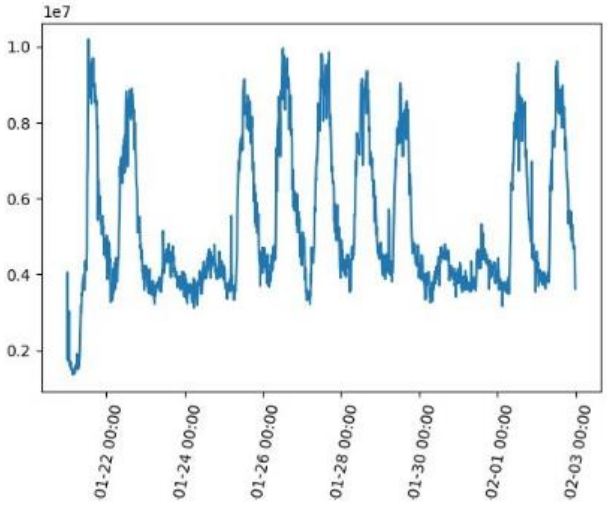

Figure 2. SINET outgoing trafic

More importantly, the session data stream on SINET is high-volatility traffic. In both ingoing and outgoing sessions, the peak traffic is about 2.5 times larger than average traffic. In this paper we present our pipeline for handling massive session data stream adopting taskdecomposition based parallelism. Our pipeline can handle high-volatility session data stream of about 500GB-650GB per day.

2. RELATED WORK

Asghari et al. [1] propose a systematic approach for providing comparative infection metrics from large-scale noisy sinkhole data of Conficker botnet. In [2], a large dataset of 350 million HTTP request logs is analyzed for understanding user behavior of mobile cloud storage service. [3] et al. report an empirical analysis for extracting and modeling the traffic patterns of 9600 cellular towers deployed in a metropolitan city. [4] Presents a characterization of Amazon's Web Services (AWS), which reveal that most of the content residing on EC2 and S3 is served by one Amazon data center located in Virginia. [5] et al. proposes a characterization of Dropbox by means of passive measurements of four vantage points in Europe, collected during 42 consecutive days. In [6], a new framework to enable a macroscopic characterization of attacks, attack targets, and DDOS protection is proposed. [7] Presents an analysis of online campus storage systems and data sharing services for more than 19,000 students and 500 student groups. [8] et al. present the large-scale characterization of inbound attacks towards the cloud and outbound attacks from the cloud using three months of NetFlow data in 2013 from a large cloud provider.

Wang et al. show mobile traffic patterns of large scale networks in urban environment by gathering data in cellular towers. Sandiana National Laboratories (SNL) adopts Splunk for managing the Red Sky Supercomputer [11]. Bitincka et al. adopts Splunk for optimizing data analysis with a semi-structured time series database [12]. GPUs were initially designed for graphics rendering, but, because of their cost effectiveness, they were quickly adopted by the HPC community for scientific

computations [13]. GPUs have also been used to accelerate functions such as pattern matching [14], network coding [15]. Ando [16] proposes a lock-free algorithm of data clustering using GPGPU. Ando [17] proposes a Multi-GPU based pipeline system with ELK stack. LSTM algorithm used in our pipeline is partly presented in [18].

3. OVERVIEW

Figure 3 depicts our pipeline for handling massive session data stream on SINET. Our pipeline is divided into three parts.

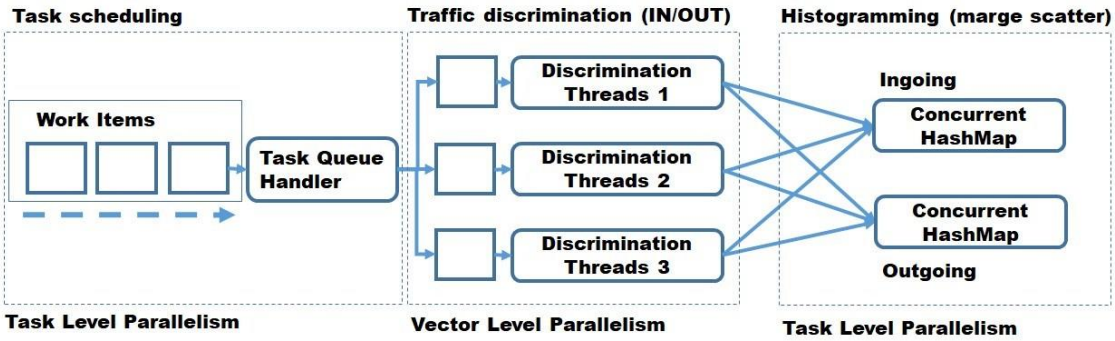

Figure 3. Overview of our data processing pipeline

3.1. Dynamic Task Scheduling

We adopt task decomposition for dynamic allocation. In handling massive and high-volatility traffic data, each work item's amount of processing time is different or even unknown at the beginning of the computation. We use task decomposition-based scheduling for coping with the session stream of SINET (academic backbone network) because the amount of processing time between tasks is usually variable and/or unpredictable.

3.2. Traffic Discrimination

At the second phase of our pipeline, we divide massive session data into ingoing/outgoing. We use an acronym for Compute Unified Device Architecture (CUDA) Thrust Library for calculating a huge amount of network addresses of session data by massive bit masking. For computing a large number of bit masking in parallel, we adopt vector level parallelism of General Purpose Graphic Processor Unit (GPGPU). To put it simply, we leverage GPGPU for traffic discrimination of session data.

3.3. Histogramming

Finally, our pipeline outputs a time-series of ingoing/outgoing session data stream by mergescatter pattern based histogramming. For generating time series data, we use a concurrent hash map of Intel TBB. The concurrent hash map stores the pairs[i] of <timestamp, data[i]> which are represented as histogram.

Generally, our pipeline adopts a coordinated batch processing pattern. Coordinated batch processing is particularly effective as the workloads increase. Coordinated batch processing is vital

to pull multiple outputs back together to generate some aggregated outputs. For handle huge workloads of session data, we apply coordinated batch processing.

3.4. Coordinated Batch Processing

Coordinated batch processing is another key concept of our pipeline. Coordinated batch processing is vital to pull multiple outputs back together in order to generate some sort of aggregated output. The most canonical example of this aggregation is the reduce part of the map-reduce pattern for generating time-series data. It is easy to see that the map step is an example of sharding a work queue. The reduce step is an example of coordinated processing that eventually reduces a large number of outputs down to single aggregate response.

4. METHODOLOGY

Figure 4 shows another overview of our system which consists of several patterns and parallelism. The basic pattern is master-worker pattern. The master thread enqueues the chunks of session data in the task decomposition manner. From the viewpoint of worker threads, our system run in dynamic scheduling.

The master-worker design pattern divides the roles of computation in the items as follows.

[1] Coping with pieces of work to workers
[2] Collecting the outputs of the computations from the workers
[3] Performing I/O scheduling on behalf of the workers, sending them the data which workers are supposed to process, particularly accessing a file

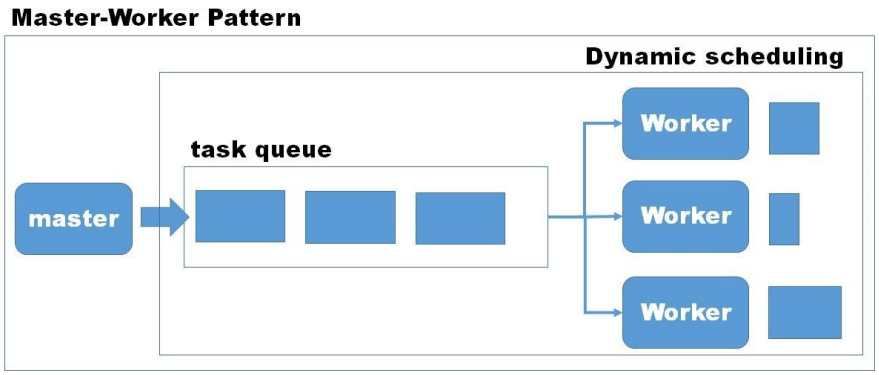

Figure 4. Design pattern of our pipeline

Each simplest form and implementation of the master-worker pattern involves a single master node and multiple worker nodes, as shown in Figure 3. More specifically, the master thread traverses the session data file directory and enqueues the file name. When the queue is complete, the master thread waits until the worker thread processing packets consumes a file name and removes it from the queue.

4.1. Task Decomposition

The two fundamental components of algorithms are tasks and data. A task operates on data, either modifying it in place or creating new data. In a parallel computation, multiple tasks need to be managed and coordinated. If we want to transform code into a concurrent version, there are two

ways - data decomposition and task decomposition. In data decomposition, the program copes with a large collection of data and can independently compute every chunk of the data. In task decomposition, the process is partitioned into a set of independent tasks that threads can execute in any order. Besides, concerning task decomposition, there are several ways to handle tasks.

It is important to note that task decomposition is classified into functional decomposition. The opposite of data parallelism is functional decomposition, an approach that runs different program functions in parallel. At best, functional decomposition improves performance by a constant factor. For example, if a program has functions f, g, and h, running them in parallel at best triples performance, but only if all three functions take the same amount of time to execute and do not depend on each other, and there is no overhead. Otherwise, the improvement will be more minor.

4.2. Dynamic Scheduling

We can allocate tasks to threads in two different ways: static scheduling and dynamic scheduling. Static scheduling of which the cost is known at the outset of computation is not appropriate in daily processing of session data. Although some daily patterns could be observed, traffic of academic backbone networks is changing daily. Besides, drastic traffic increase occurs, particularly on ingoing traffic. In dynamic scheduling, tasks must be assigned to threads for execution. Perhaps the more correct way to say this is that threads must know which tasks to execute.

In either case, you always want to assure that the amount of computation done by threads is roughly equivalent. That is, a load of computation is balanced in each thread.

5. IMPLEMENTATION

5.1. Task Queue

The popular dynamic scheduling method involves setting up a shared container (typically a queue) which can hold tasks and allow threads to pull out a new task once the previous task is complete. Tasks (or adequate descriptions of tasks) must be encapsulated into some structure that can be pushed into the queue. Access the queue must be mutually exclusive between threads to ensure which threads get unique tasks and no tasks are lost through some corruption of the shared container. The task queue design's main purpose is to ensure that each chunk of work is processed within a certain amount of time.

5.2. Merge Scatter Pattern

The merge scatters pattern, associative and commutative operators are provided for merging elements in case of a collision. With the nature of this pattern, scatter could occur in any order. Therefore, both associative and commutative properties are required. An example that uses addition as the merge operator is shown in Figure 5. It is straightforward to adopt a merge scatter pattern to implement histograms with the adding operation.

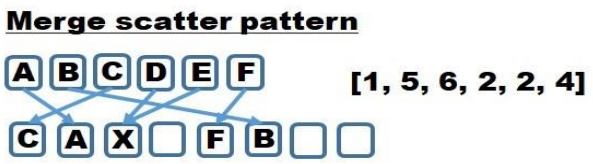

Figure 5. Merge scatter pattern

5.3. Deep Learning for Time-series Analysis

Deep learning for time-series analysis is a new endeavour and promising field. Deep learning is advantageous because deep learning is a highly flexible technique. Specifically, it deep learning provides the capability of modelling highly complex behaviour. Besides, it offers the possibility for nonlinear temporal behaviour without estimating at function forms – which could potentially great change for nonstatistical forecasting techniques. Many of the trouble of preprocessing data to a fit the assumptions can be evaded when deep learning model is applied. The potential advantages of deep learning are as follows:

[1] No requirement of stationarity necessary.
[2] There is no need to develop the art and skill of picking parameters, such as assessing seasonality and order of a seasonal ARIMA model. The art and kill of selecting parameters are not required. For example, assessing seasonality and sequence of a seasonal ARIMA model.
[3] With deep learning, we do not need to develop a hypothesis concerning a dynamics of s system.

However, it is important to notice that deep learning is not a silver bullet. Although there is no requirement or assumption of stationarity for deep learning applied to time series, in practice, deep learning does not work excellently for fitting data with a trend unless the basic architectures are embedded. So still we need to preprocess our data or our technique.

In this paper, we apply Long Short Term Memory (LSTM) for anomaly detection. Long Short Term Memory networks - usually just called LSTMs - are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber [9], and were refined and popularized by many people in the following work.1 They work tremendously well on a large variety of problems, and are now widely used.

6. EXPERIMENTAL RESULT

We report the observation and prediction about SINET traffic during 2/27 - 3/8. In the observation, we use a rack server of Dell(TM) PowerEdge(TM) PE C4140 with 96 core CPUs and 4 GPUs. CPU is Intel(R) Xeon(R) Platinum 8160 CPU @ 2.10GHz. Our PE C4140 has 1.5TB memory. GPU is Tesla V100 PCIe 32GB (NVIDIA Corporation GV100GL). Data stream is forwarded to our pipeline from PA-7000. The PA-7000 Series is powered by a scalable architecture for the purpose of applying the appropriate type and volume of processing power to the key functional tasks of networking, security, and management.

Figures 6 and 7 show the traffic volume of session data. Session data is grouped into one-hour frame bins. In ingoing traffic, we observe a clear diurnal pattern from 01/25/2021 to 01/29/2012. This period is the weekday of the academic campus. On the other hand, 01/30/2021 and 01/31/2021 are weekends, which results in the decrease in traffic and plateau. Concerning outgoing traffic, we observed more clear diurnal patterns during 03/02-03/06. The key findings are that in outgoing traffic, there is no drastic spike compared with ongoing traffic.

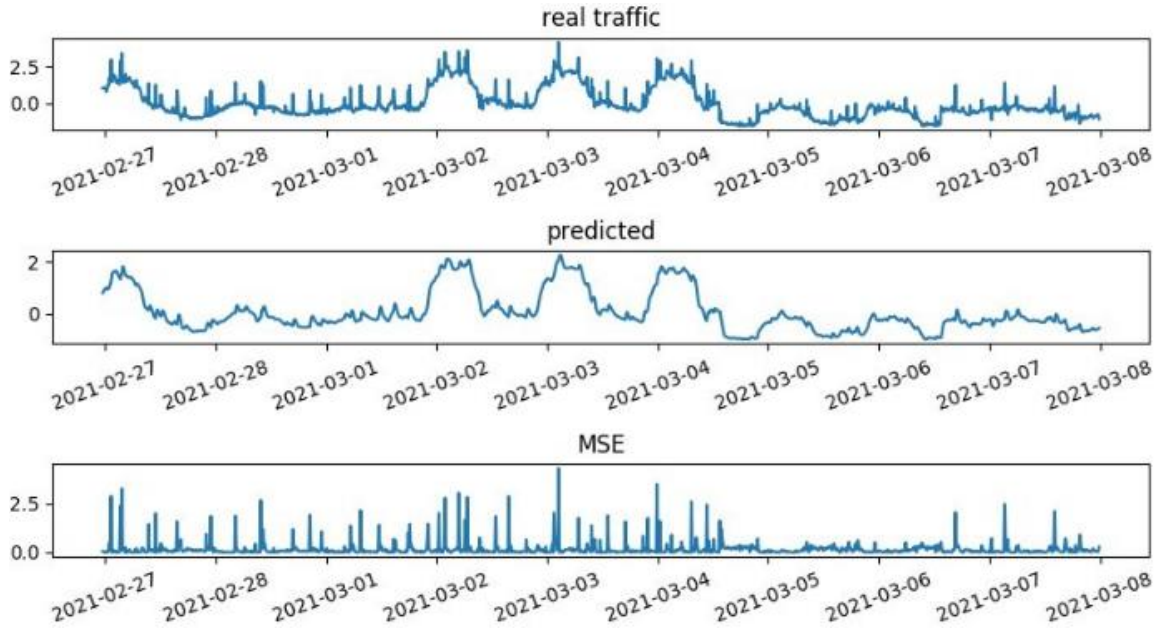

Figure 6. Anomaly detection of SINET outgoing traffic

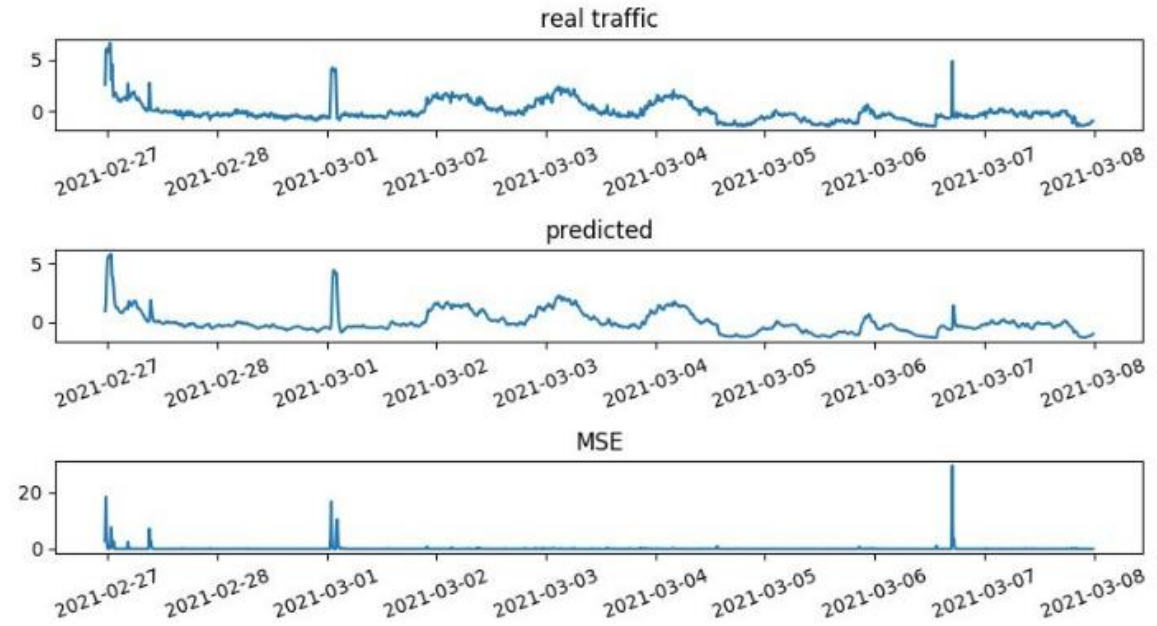

Figure 7. Anomaly detection of SINET ingoing traffic

In summary, our multi-GPU-driven pipeline has succeeded in processing huge workloads of about 1.2 to 1.6 billion of session stream (500GB-650GB) within 24 hours during 03/02-03/06.

Our key observations are as follows:

[1] We observed some drastic spikes in ingoing sessions.
[2] There have been no extreme spikes in outgoing sessions.
[3] We have succeeded in detecting anomaly (several spikes in ingoing sessions) using MSE.

7. DISCUSSION

In this paper, we give priority to handle huge session data over any other matter. In 2009 researchers from Google wrote a paper entitled The Unreasonable Effectiveness of Data [10]. According to this paper, if machine learning algorithm A using a messy dataset with a trillion lines can be highly effective in tasks, algorithm A is downright useless in coping with a clean dataset with a mere million lines. If algorithm A does not work with a dataset comprising a million examples, our intuitive conclusion is that it does not work at all. Our pipeline system is based on the concept of this paper. However, at the same time, simple time-series analysis algorithm is deployed on currently our system. This point could be the limitations or deficiencies of our system.

8. CONCLUSION

Science Information Network (SINET) is the Japanese academic backbone network for more than 800 research institutions and universities. In our operational experience, SINET suffers the drastic traffic increase by about five times larger than average. This paper presents the methodology and system implementation for handling massive and high-volatility session data streams. We adopt two technologies: dynamic scheduling and LTSM based anomaly detection. Besides, three main features are discussed: Tash scheduling, Traffic discrimination, and Histogramming. We adopt a task-decomposition based dynamic scheduling method to handle the massive session data stream of SINET. In the experiment, we have analysed SINET traffic from 2/27 to 3/8 and detect some anomalies by LSTM based time-series data processing. We can conclude that we have succeeded in handling rapid session increase during 2/27-3/08. For further work, we are building more sophisticated algorithm for time-series analysis. For example, more finegrained tuning of hyper parameters and bi-directional RNN could be applied for our system.

REFERENCES

[1] Hadi Asghari, Michael Ciere, Michel J. G. van Eeten, ``Post-Mortem of a Zombie: Conficker Cleanup After Six Years", USENIX Security Symposium 2015: pp.1-16

[2] Zhenyu Li, Xiaohui Wang, Ningjing Huang, Mohamed Ali Kaafar, Zhenhua Li, Jian Zhou, Gaogang Xie, Peter Steenkiste: An Empirical Analysis of a Large-scale Mobile Cloud Storage Service. Internet Measurement Conference 2016: pp.287-301

[3] Fengli Xu, Yong Li, Huandong Wang, Pengyu Zhang, and Depeng Jin, ``Understanding Mobile Traffic Patterns of Large Scale Cellular Towers in Urban Environment", IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 25, NO. 2, APRIL 2017

[4] Ignacio Bermudez, Stefano Traverso, Marco Mellia, Maurizio M. Munaf?: Exploring the cloud from passive measurements: The Amazon AWS case. INFOCOM 2013: pp.230-234

[5] Idilio Drago, Marco Mellia, Maurizio M. Munafo, Anna Sperotto, Ramin Sadre, Aiko Pras: Inside dropbox: understanding personal cloud storage services. Internet Measurement Conference 2012: pp.481

[6] Mattijs Jonker, Alistair King, Johannes Krupp, Christian Rossow, Anna Sperotto, Alberto Dainotti: Millions of targets under attack: a macroscopic characterization of the DoS ecosystem. Internet Measurement Conference 2017: pp.100-113

[7] Song in Liu, Xiaomeng Huang, Haohuan Fu, Guangwen Yang: Understanding Data Characteristics andAccess Patterns in a Cloud Storage System. CCGRID 2013: pp.327-334

[8] Rui Miao, Rahul Potharaju, Milan Yu, Navendu Jain: The Dark Menace: Characterizing Network-based Attacks in the Cloud. Internet Measurement Conference 2015: 169-182

[9] Sepp Hochreiter, Jürgen Schmidhuber: Long Short-Term Memory. Neural Comput. 9(8): 1735-1780 (1997)

[10] Alon Halevy and Peter Norvig and Fernando Pereira, "The Unreasonable Effectiveness of Data", IEEE Intelligent Systems, vol.24, pp8-12, 2009

[11] Jon Stearley, Sophia Corwell, and Ken Lord. Bridging the gaps: Joining information sources with splunk. In Workshop on Managing Systems via Log Analysis and Machine Learning Techniques, 2010.

[12] Ledion Bitincka, Archana Ganapathi, Stephen Sorkin,and Steve Zhang. Optimizing data analysis with a semistructured time series database. In Workshop on Managing Systems via Log Analysis and Machine Learning Techniques, 2010.

[13] J. D. LUEBKE OWENS, D.GOVINDARAJU, N.HARRIS, M.KRGER, J.LEFOHN, , and T. J. PURCELL. A survey of general-purpose computation on graphics hardware. In Computer Graphics Forum 26, 1 (2007), pages 80–113, 2007.

[14] SHOJANIA H., LI B, , and WANG X. Nuclei. Gpuaccelerated many-core network coding. In IEEE Infocom, pages 459–467, 2009.

[15] R. SMITH, GOYAL N., ORMONT J., SANKARALINGAM. K, and ESTAN. C. Evaluating gpus for network packet signature matching. In Ie International Symposium on Performance Analysis of Systems and Software, 2009.

[16] Ruo Ando, A lock-free algorithm of tree-based reduction for large scale clustering on GPGPU. In Proceedings of the 2nd International Conference on Artificial Intelligence and Pattern Recognition, ACM 2019, ISBN 978-1-4503-7229-9, AIPR 2019, pp129-133, 2019

[17] Ruo Ando, "Multi-GPU Accelerated Processing of Time-Series Data of Huge Academic Backbone Network in ELK Stack", Usenix LISA 2019 (33th Large Installation System Administration Conference Large Installation System Administration Conference) October 28– 30, 2019 Portland, OR, USA 10 2019

[18] Ruo Ando, Yoshiyasu Takefuji: “A constrained recursion algorithm for batch normalization of tree-sturctured LSTM”, CoRR abs/2008.09409 (2020)