DOI: 10.14529/jsfi140206

Extreme Big Data (EBD): Next Generation Big Data Infrastructure Technologies Towards Yottabyte/Year

Satoshi Matsuoka [1], ***Hitoshi Sato*** [1], ***Osamu Tatebe*** [2], ***Fuyumasa Takatsu*** [2], ***Mohamed Amin Jabri*** [2], ***Michihiro Koibuchi*** [3], ***Ikki Fujiwara*** [3], ***Shuji Suzuki*** [1], ***Masanori Kakuta*** [1], ***Takashi Ishida*** [1], ***Yutaka Akiyama*** [1], ***Toyotaro Suzumura*** [4], ***Koji Ueno*** [1], ***Hiroki Kanezashi*** [1], ***and Takemasa Miyoshi*** [5]

Our claim is that so-called "Big Data" will evolve into a new era with proliferation of data from multiple sources such as massive numbers of sensors whose resolution is increasing exponentially, high-resolution simulations generating huge data results, as well as evolution of social infrastructures that allow for "opening up of data silos", i.e., data sources being abundant across the world instead of being confined within an institution, much as how scientific data are being handled in the modern era as a common asset openly accessible within and across disciplines. Such a situation would create the need for not only petabytes to zetabytes of capacity and beyond, but also for extreme scale computing power. Our new project, sponsored under the Japanese JST-CREST program is called "Extreme Big Data", and aims to achieve the *convergence of extreme supercomputing and big data* in order to cope with such explosion of data. The project consists of six teams, three of which deals with defining future EBD convergent SW/HW architecture and system, and the other three the EBD co-design applications that represent different facets of big data, in metagenomics, social simulation, and climate simulation with real-time data assimilation. Although the project is still early in its lifetime, started in Oct. 2013, we have already achieved several notable results, including becoming world #1 on the Green Graph 500, a benchmark to measure the power efficiency of graph processing that appear in typical big data scenarios.

Keywords: Big Data, Supercomputing, Extreme Computing and Big Data Convergence, Data Intensive Computing, Non-Volatile Memory.

1. Introduction

"Big Data" has become a hot topic in mainstream IT; although large scale databases as well as data intensive computing has existed from the past, amassing petabytes to even zetabytes of data is becoming a reality, as overall data generation rate and its traffic, both on the Internet as well as dedicated instrumentation networks, are exploding at exponential ratio as Moore's law continues to speed up processing, enlarge memory, and increase sensor resolution, etc.

However, to most common people "Big Data" is almost synonymous to "mining people's private data (such as purchase history, credit card spending, location tracking, etc.) for marketing purposes". Such a definition is not only extremely narrow, limiting the potential application impact to the global society in areas such as medicine, energy, climate, manufacturing, etc., but also imposes unnecessary underestimation of the infrastructure necessary for future Big Data, causing unnecessary divide between traditional IDCs whose present-day "Big Data Infrastructure" often being a group of re-purposed web-server connected by at best 10GbE or even 1GbE endpoint speeds; this is in contrast with interconnects of supercomputers today offering a few orders of magnitude faster interconnects for massive data exchange for sophisticated processing of data.

[1] Tokyo Institute of Technology, Tokyo, Japan
[2] University of Tsukuba, Tsukuba, Japan
[3] National Institute of Informatics, Tokyo, Japan
[4] IBM Research / University College Dublin, Dublin, Ireland
[5] RIKEN Advanced Institute for Computational Science, Chuo-ku Kobe, Japan

As such, many instances of big data are not so "big" in terms of capacity nor processing complexity, the latter often being simple mining to detect simple statistical trends, and/or associativity with a small number of classes of silo'd datasets within an organization. This is why simple data processing abstractions on simple hardware, such as Hadoop[1] running on commodity servers, is widely employed. However, the future of big data is not expected to be the case. There are various predictions on "breaking down of silos" where organizations will open up their data for public consumption, either for free or for a fee, along with immense increase in varieties of data sources driven by technologies such as IoT[12]. There, meaningful information would be extracted from unstructured and seemingly uncorrelated data spanning exabytes to zetabytes, utilizing higher-order $O(n \times m)$ algorithms on irregular structures such as graphs, as well as conducting data assimilations with massive simulations in petaflops to even exaflops. This is already happening in data-intensive science, in areas such as particle physics[2], cosmology[8], life science[13], where sharing of large capacity research data as "open data" has become domain practice; there is strong likelihood that this will proliferate to the common Internet, just as the Web, which was originally envisioned to share scientific hypertext, took over the world as the mainstream information sharing IT infrastructure.

We refer to such evolved state of big data as "Extreme Big Data", or EBD for short, as a counterpart to extreme computing. An IT infrastructure supporting EBD will involve massive requirements of compute, capacity and bandwidth of resources throughout the system, as well as co-existence of efficiency and real-time resource provisioning, as well as flexible programming environment and adaptability of compute to data locations to minimize the overall data movement Moreover, they have to be extremely power and space efficient, as those factors are the principle parameters nowadays that limit the overall capacity of a given IT system.

Given such requirements, our claim is that, neither existing supercomputers, nor traditional IDC Clouds, are appropriate for the task; rather, we believe that the convergence of the two are necessary, based on the upcoming technology innovations as well as our own R&D to actually achieve such effective convergence. These include extensive and hierarchical use of new generations of non-volatile memory (NVM) as well as processor-in-memory technologies to achieve high capacity in memory and processing with very low power; high-bandwidth many-core processors that can make use of such memory composed in a deep and hierarchical fashion; low latency access of elements of such memory hierarchy, especially NVMs, from all parts of the machine via a scalable, high-bandwidth, hi-bisection network; management of memory objects dispersed and resident throughout the system across application boundaries as *EBD Objects* in the hierarchy, as well as their automated performance tuning and high resiliency; various low-level big-data workload algorithms such as graph algorithms and sorting of various types of keys; various libraries, APIs, languages, as well as other programming abstractions for ease-of-use by the programmer, hiding the complexity of such a large and deep system; finally, resource management to accommodate complex workflows of both batch and real-time processing, being able to schedule tasks balancing the processing requirements versus minimizing data movement.

With such comprehensive overhaul of the entire system stack, coupled with advances in both computing and storage, we expect that we could amplify the EBD processing power of existing Cloud datacenters by several orders of magnitude. Our latest project titled "Extreme Big Data", sponsored by the Japan Science and Technology Agency (JST), under the research area program "Advanced Core Technologies for Big Data Integration" of the Strategic Basic Research Program (CREST), embarked on a 5-year research to develop such EBD technologies

during the period of Oct. 1st, 2013 to Sep. 30th, 2018. The project involves six internal teams, three being in EBD systems area and the other three in EBD applications, in order to pursue co-design of the EBD system stack. Figure 1 describes the overall scheme of the convergence of the HPC and the big data architecture, and the necessity of the co-design with EBD grand challenge applications.

In the project we plan on developing individual technological elements of EBD as mentioned above, such as designing the future HPC – Big Data convergent architecture, including memory hierarchy and network designs, as well as various algorithms and programing frameworks, along with acceleration of system-wide data and resource management. The co-design applications include life sciences, social simulations, and data assimilation in climate modeling, each with different system-level as well as API requirements for EBD. We will also pursue attaining top-level performance in big data benchmarks, such as the Graph 500[5], which measures the absolute performance of graph processing, as well as the Green Graph 500[6], which ranks the power efficiency of machines when processing the graphs under the Graph 500 rule. In fact, on the November 2013 edition of the Green Graph 500, we achieved No.1 and No.4 in the world, the latter result attained while offloading most of the graph data structures onto a Flash NVM. Sorting is another important benchmark, where we aim to challenge Petabyte sorting speeds on modern many-core processors such as GPUs and Xeon Phis. Finally, we plan on integrating results from other groups working on large-scale big data hardware/software stack. Some of the candidate systems include the Berkeley Spark[9] and ORNL Adios[4]. Figure 2 describes the overall scheme, which instantiates some of the details from Figure 1

The intermediate results of the EBD project will be implemented on our next generation TSUBAME3.0 supercomputer, the successor to the highly successful TSUBAME2.0[24], to be commissioned in the first half of 2016. In the current design of TSUBAME3.0 is being done as the phase 2 of the overall evolution of EBD architecture, and in that sense, TSUBAME3.0 could be the first EBD convergent architecture in production.

The rest of the paper provides an overview of each element of our EBD project, focusing on the candidate architecture, followed by synopsis of the system and application groups and their status quo as of June, 2014, 9 months into the 5-year project.

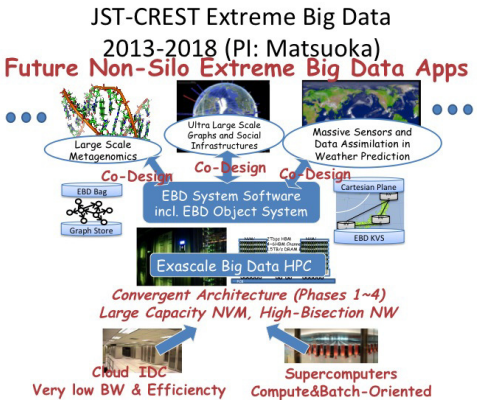

Figure 1. Overview of EBD project

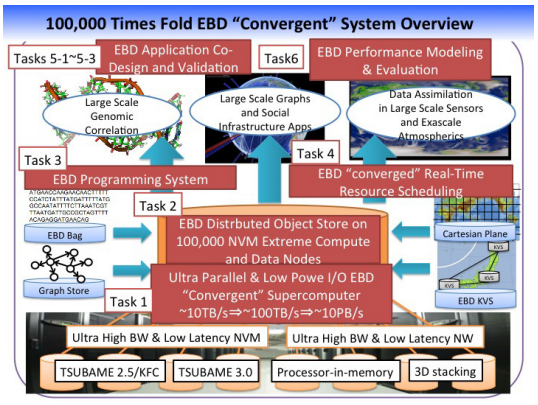

Figure 2. EBD software stacks

2. EBD Architecture

We will first discuss the envisioned EBD architecture. We will construct a series of EBD architecture prototypes, either by adding additional resources to existing machines such as the TSUBAME-KFC, or by building one entirely new. For both types of machines, the intent is to assess the behavior of EBD applications of the future; for this purpose, we will identify the various EBD benchmarks, as well as the EBD co-design applications, in order to model the performance requirements of several classes of EBD applications running on the prototype hardware.

At the same time, we will work with various vendors we have relationships with in our TSUBAME project. Of particular importance is the design of the overall memory hierarchy in relationship to the massively parallel, many-core processors such as the GPU or Xeon Phi that are representative of the current generation of hardware. Non-volatile memory will be used judiciously in order to attain large memory and storage capacity while preserving performance as much as possible. The four phases of the prototypes are being planned as follows: the first phase will involve standard mSATA SSD devices; second generation will involve much faster M.2 flash devices and large number of I/O channels directly utilizing fast I/O busses such as PCI-e; the third generation will incorporate next generation NVM directly connected to memory or some type of coherent bus such as HyperTransport or NVIDIA NV-Link; the third generation will involve placing flash memory or preferably newer generations of NVM on the memory bus of the CPU along with the DRAM, greatly enhancing the bandwidth and lowering the latency; finally, the future, forth generation hardware will integrate NVM, DRAM, and the lightweight data-centric processing core on the same 3-D die-stacking configuration, possibly integrated with heavyweight scalar-centric core that provides for performance in low-threaded workloads (Figure 3). For the last architecture, the traditional I/O bandwidth will equal the memory bandwidth on 3-D.

For each generation, we will be building a prototype, and/or creating a simulation environment to investigate their performance implications.

- Phase1 (2010- TSUBAME2.0-2.5 generation — Flash SSD local burst buffer): Each node will embody one or more server-grade SSD in place of HDDs, resulting in capacity of 100GB-1TB range. Although such configuration is becoming more commonplace, in 2010 the durability of SSDs were substantially questioned under heavy generalized HPC workload for TSUBAME2.0, which for the first time for a petascale supercomputer embodied 60GB $\times$ 2 sever-grade SLC SSDs per each node. Fortunately, such a concern has been disproven, for the nearly 3,000 SSDs in TSUBAME2.0 has seen very small number of failures, despite many of the localized and thus heavy I/O workloads being delegated to the SSDs. The aggregate capacity of SSDs in TSUBAME2.0 was 190 Terabytes, with approximately 400-500GB/s of I/O Bandwidth.
- Phase2 (2015- TSUBAME3.0 generation — large capacity flash with small embedded SSD modules): Each node will be designed from ground-up to embody multiple small high-bandwidth SSD modules such as the M.2 module with PCI-e connection. This will allow for pseudo-direct DMA access of the flash modules, and allows for global management of all the SSDs aggregated as a single namespace entity. Aggregate capacity can be as high as 10 Petabytes, with 10 Terabyte/s I/O bandwidth, which is several times faster compared to all the Top500 machines, whose I/O bandwidth does not exceed 2 Terabytes/s.

- Phase3: (2017-2020 Non Volatile Memory Modules): By placing the non-volatile memory modules onto the memory bus, in the form of DIMM or other modules, will allow for great increase in bandwidth and capacity while lowering the latency considerably, with DIMM modules affording bandwidths of 25 Gigabyte/s or greater. However, since the system will have to be composed of both DRAM and NVM DIMMs due to durability as well as performance issues, effective algorithms to make use of each memory region according to the access characteristics of memory objects must be devised. I/O bandwidth can become as high as Petabyte/s range depending on the NVM technology.
- Phase4: (2020-): (3-D integration of NVM and DRAM and PIM) The ultimate evolution of the architecture is 3-D stacking of NVM and DRAM on a low power processor, which is in effect a PIM (Processor-In-Memory) in modern incarnation3. Optionally, large cores can be integrated using 2.5D interposer technology for low latency, low-threaded processing. Such an architecture will totally unify NVM with DRAM, with tremendous bandwidth while being low power with 3-D stacking. I/O bandwidth, which is essentially on par with the memory bandwidth, can reach 10s to 100s of Petabytes/s, achieving a Yottabyte/year data processing capability.

As such, the evolution of the I/O bandwidth can be high as 4-5 orders of magnitude in 10 years, from hundreds of Gigabytes/s for TSUBAME2.0 in 2010, to 10s of Petabytes/s for the 2020-fourth generation EBD machine. The issue then, is what are the system software, programming paradigms, basic EBD algorithms, and applications that could best utilize such immense and rapid increase in performance.

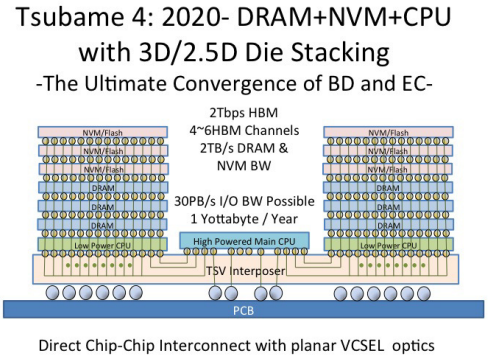

Figure 3. EBD architecture

3. EBD System Software

As described in previous section, EBD machines are based on various novel technologies derived from extreme-scale supercomputing, such as many-core parallel processing, ultra high bandwidth/low latency networks, non-volatile memory techniques, and high performance database techniques, etc.. Therefore, we have to develop completely redesigned new software stacks that support EBD architecture, since existing conventional system software can not support such new hardware device technologies.

3.1. Many-core I/O

Many-core processors such as GPU can provide extremely fast computational power and high memory bandwidth; however, the capacity of memory on GPU devices are limited in size to accommodate EBD applications. In order to mitigate the GPU capacity problem, we have designed a novel I/O prototype machine that consist of multiple GPU accelerators and multiple mini SATA (mSATA) SSD devices. In particular, aggregation of multiple mSATA SSDs and strategy for utilization the GPU and NVM devices are essential to perform high bandwidth, high IOPS as well as large capacity and low energy consumption. Our preliminary results based on basic I/O bandwidth to the prototype with 16 of mSATA SSD devices by fio I/O benchmarks exhibit that the sequential read bandwidth achieves 7.69GB/s (92.4% of theoretical peak), and the write bandwidth achieves 3.8GB/s (90.2% of theoretical peak). The results also exhibit that using multiple mSATA SSDs performs 3.20 to 7.60 times faster than a common PCI-e attached flash memory device, and 11.1 to 17.8 times faster than conventional SSDs attached on a local compute node of TSUBAME 2.5. We also measure the performance of a matrix-vector multiplication benchmark that overlaps file I/O from/to NVMs, memory copy between host and GPUs, and computation on GPUs. The results using 280MB to 140GB of input matrices, which exceed the GPU memory capacity, exhibit that our prototype can achieve 3.06GB/s from 8 mSATA SSDs to a GPU device by using the RAID0 configuration with appropriate stripe size. We also found that using pinned memory and setting small chunk size for overlapping significantly affect data transfer performance.

3.2. Programming Framework

EBD architecture with deep memory hierarchy is encapsulated as EBD objects. In order to operate EBD architecture from applications, we provide a programming framework for the EBD objects based on the system-level programming model and the user-level programming model. The system-level programming model provides specific interfaces and concrete implementations for EBD objects to collect hardware information and to control data layout, while the user-level programming model provides several features to control EBD objects from applications.

As an example of ongoing work on the EBD programming framework, we have developed a MapReduce-style programming framework, called HAMAR (Highly Accelerated MapReduce) for massively data parallel processing on EBD machines with deep hierarchical memory [26]. Our framework automatically handles memory overflows from GPUs by dynamically dividing data into multiple chunks and overlaps CPU-GPU data transfer and computation on GPUs as much as possible. Our experimental results for PageRank graph applications on TSUBAME2.5 using 1024 nodes (12288 CPU cores, 3072 GPUs) exhibit that our GPU-based implementation performs 2.10x faster than running on CPU when the size of graph exceeds the capacity of GPU's device memory.

3.3. Basic Algorithms

Based on novel features of of massively parallel many-core processors and nonvolatile memory devices, we develop basic algorithms, such as indexing, multi dimensional array matching, sorting, unstructured graph search, spatial clustering, etc., to support EBD applications.

As an instance of implementation to overcome deepening memory hierarchy in extreme-scale computing systems, we have developed a fast graph processing implementation with a novel

graph offloading technique using NVMs for Hybrid BFS (Breadth-First Search) algorithm that is widely used in the Graph500 benchmark [21]. Hybrid BFS uses a mixture of two approaches, a conventional top-down approach and a bottom-up approach by changing search directions using parameters. Based on the DRAM-based NUMA-optimized Hybrid BFS implementation called NETAL (NETwork Analysis Library), which achieves 10.5 GTEPS on the Graph500 list (November 2012), we carefully offload infrequent accessed graph data to secondary semi-external memory devices such as NVMs and directly read the data on the devices on demand. The results using Graph500 problems shows that our approach with highly localized data access can achieve competitive performance with the conventional DRAM only approach, although we aggressively extend memory footprints onto NVMs. Using the implementation with some improvements, we have also achieved 4.35MTEPS/Watt on a Scale 30 problem and ranked the 4th position in the big data category in the Green Graph500 (November 2013), which is the 1st position using a single commodity server in the big data category.

Large-scale distributed sorting is another instance for EBD basic algorithms. Splitter-based parallel sorting algorithms are known to be highly efficient for distributed sorting due to their low communication complexity. Although using GPU accelerators could help to reduce the computational cost in general, their effectiveness in distributed sorting algorithms on large-scale heterogeneous GPU-based systems have not been well studied. We accelerate the existing HykSort algorithm by offloading costly computation phases to GPU devices. Preliminary investigations show that the local sort phase is dramatically accelerated by GPUs, while the merge phase achieves negligible performance improvement. We then evaluate the performances of our implementation with only local sort acceleration on the TSUBAME2.5 supercomputer that comprises over 4000 NVIDIA K20x GPUs. Performance evaluation of weak scaling shows that we achieve 389 times speedup with 0.25TB/s throughput when sorting 4TB 64bit integer on 1024 nodes compared to running on 1 node; on the other hand, for CPU vs. GPU comparison, our implementation achieves only 1.40 times speedup using 1024 nodes.

4. Distributed Object Store for EBD Applications

Extreme big data (EBD) applications may involve parallel access from handreds of thousands of processes, which requires not only scalable performance of bandwidth but also I/O operations per seconds (IOPS) to the number of processes. Important issue for scalable I/O bandwidth is to maximize access locality to read and write data, and to accomodate parallel access of metadata such as location of the data, from hundreds of thousands of clients. To improve the read access locality, data location aware process scheduling is essential, which assigns processes depending on the input data location. Regarding the write access, it is necessary to find a space to maximize the locality that may depend on successive processes that read the data as an input.

This approach is adopted by Google file system [18] and MapReduce [15]. The Google file system stores data blocks at local storage on compute nodes. The data blocks are replicated to improve reliability and availability. Jobs of the MapReduce are allocated on compute nodes considering the data block locations. Open source Apache Hadoop [1] implements the design of Google file system and MapReduce, and plenty of research follows [14, 39, 40]. The Gfarm file system [35] and the Pwrake workflow system [33] cover general workflow execution using this locality aware approach. The Gfarm file system also stores data at local storage on compute nodes. The Pwrake workflow system executes a workflow written in Rakefile in parallel assigning processes on a compute node where the input data is stored. The Pwrake, moreover, allocates

processes to minimize data transfer size among compute nodes during a whole workflow execution by multi-constraint graph partitioning [34], which can reduce data transfer size of intermediate data generated during the workflow execution.

EBD applications do not really require posix file system interface. It requires rather key-value interface but having additional features. The first additional feature is specifying a collection of key-value pairs. We call this collection *a bag*. The second feature is a range query of keys to form a bag. How to form a bag is still under discussion, but the range query is considered to be a basic operation to select key-value pairs. The third feature is to specify parallel operations for one or two bags. When one bag is specified, the operations will be applied to each key in the specified bag in parallel. When two bags are specified, they will be applied to all combinations of two keys in each bag in parallel, which supports $O(mn)$ complexity computation when bags include m and n keys, respectively. The forth feature is a parallel query to analyze the output data.

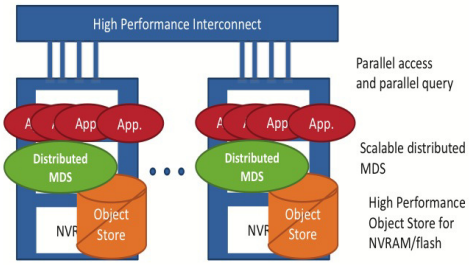

Figure 4. EBD distributed object store

We are designing an EBD distributed object store that has the above features, which consists of distributed metadata server and local object stores (Figure 4). Regarding a local object store, we assume flash storage device and non-volatile storage class memory in which the I/O performance is improved by parallel data access. Hard disk drives have been a major storage device, which performs well for sequential contiguous access, but not for parallel random accesses. On the other hand, in case of flash storage and non-volatile storage class memory, the I/O performance is improved by accessing in parallel. We design a nonblocking local object store that does not require a lock for concurrent accesses using the OpenNVM API [3]. The OpenNVM API provides a sparse address space and an atomic write operation. Traditional file system design manages blocks of file data using multi-level indirect reference. Instead, our design is based on a large-size region using the sparse address space, which avoids the overhead of indirect references. A preliminary evaluation shows scalable IOPS performance when the number of thread increases. When using 16 threads, it achieves 190 Kops/sec, while IOPS of existing directFS and XFS is not improved and stays at 61 and 16 Kops/sec, respectively.

Regarding a range query of keys, we are currently designing in-memory lock-free concurrent B+Tree. Supported operations are search, insert and delete. Dynamic rebalancing of the tree, i.e. tree nodes merge and split, is also supported. Nodes to be split or merged are frozen until replaced by new nodes. Search is not delayed by rebalancing. Search could access old (frozen) and new nodes. New Nodes may be modified for insert or delete only after replacement took

place. Insert and delete operations help finalize the replacement of encountered frozen nodes. Currently the design is under the validation.

5. EBD Interconnection Networks

5.1. Objective

Extreme big data (EBD) processing would generate different access patterns from traditional stencil and uniform communications in parallel scientific applications and lightweight communications in datacenters. It sometimes becomes access to remote flash devices, while traditional HPC applications rely on remote direct memory access. In this section we will design interconnection networks for such EBD processing. Figure 5 shows our quantitative goals of EBD interconnection networks. Our main challenges are (1) communication to remote flash in low latency, i.e. less than 10μs for 4kB transfer, and (2) optimization of non-regular access that may be determined by each execution. In this context we will attempt to make a new network topology and its custom deadlock-free routing, and fine-grain direct communication mechanism to storage. It is free from TCP/IP basis and we will support native communication, e.g. IBverb on InfiniBand.

5.2. Existing Datacenter and Supercomputer Networks

Existing interconnection networks for massively parallel computing can be classified into datacenter and supercomputer networks. Typical datacenter networks (DCNs) are built with a hierarchical structure with so-called top-of-rack (ToR) switches, aggregation switches, cluster routers, and border routers [10]. Since Ethernet has a good economy especially for 1Gbps and 10Gbps links and switches, it commonly used in DCNs. Fat tree or tree topologies are commonly used in DCNs, because they fit with its layered structure and user partitioning. By contrast, supercomputer networks use high-bandwidth links, such as 40Gbps, with custom routing on regular tightly-coupled topologies, such as k-ary n-cubes. Since InfiniBand is frequently used in supercomputer networks, it can support arbitrary topologies with their custom routings. Supercomputer networks are well optimized to regular communication patterns, such as stencil or uniform access by making the best use of regular structure. Our EBD interconnection networks should be different from both DCNs and supercomputer networks, because non-regular unpredictable access patterns are essential for EBD processing (see Figure 5).

5.3. Preliminary Design of EBD Interconnection Networks

5.3.1. Network Topology

Our preliminary design focuses on network topology. We attempt to make a metatopology in order to minimize jitter of network latency and average network latency for EBD direct storage communication that includes little regularity. For a given switch degree, we already propose a design framework of network topology[17]. The proposed topology introduces random connections to achieve low latency, but does so in a way that accounts for the physical layout of the topology so as to lead to further cable length and latency reductions[17]. Figure 6 is an example of the proposed topology and its layout. Graph analysis results[17] showed that the proposed topology has good properties in terms of latency, cable length, and throughput, when

compared to traditional low-degree torus and moderate-degree hypercube topologies, to high-degree fully-connected Dragonfly topologies[22], to the HyperX[11] topology, and to recently proposed fully random topologies[23].

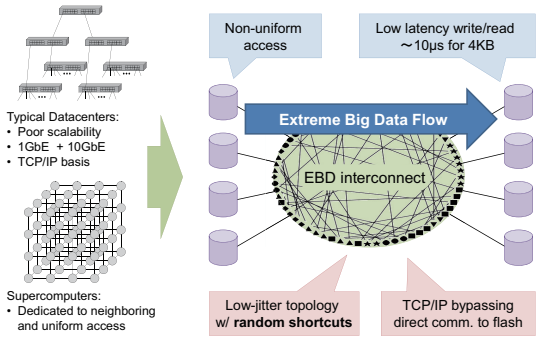

Figure 5. An overview of EBD interconnection networks

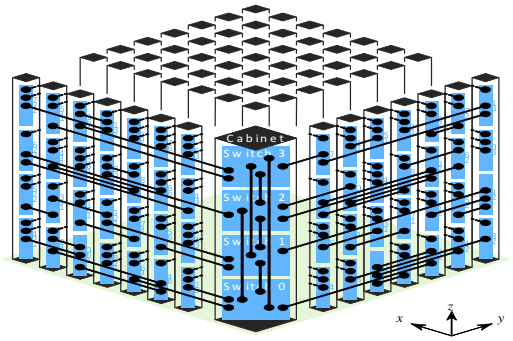

Figure 6. An example of the proposed network topology

5.3.2. Low-jitter Routing

Our key challenge for the network topology that has a little regularity, such as one in the previous subsection, is to minimize the jitter of end-to-end latency for arbitrary pair of compute nodes. Historically a minimal or shortest-path deadlock-free routing has been used for interconnection networks. By contrast, in EBD interconnection networks, to minimize the variation of end-to-end network latency, our design allows to have non-minimal paths. We are also planning to use a quality-of-service (QoS) mechanism for this purpose.

A potential scalability issue is the computational cost of path computation for topology-agnostic deadlock-free routing, which is more complex than when routing on structured topologies (see the survey in [16]). However, this computation needs to be performed only when initially deploying the system and after a change in system configuration (e.g. due to link/switch failure). Consequently, concerns about path computation should arise only at extremely large scale[23].

5.3.3. Preliminary Evaluation

We use a cycle-accurate network simulator written in C++ [23]. A header flit transfer requires at least 60 ns delay for a switch, including routing, virtual-channel allocation, switch allocation, and flit transfer from an channel to an output channel through a crossbar. Link delay is assumed to be 5 ns/m on an optical cable and the length of each cable is computed based on the cabinet layout. We use deadlock-free low-jitter routing for the proposed topology. We use three synthetic traffic patterns that determine source-and-destination pairs, namely *random uniform*.

Figure 7 shows latency (averaged over all messages) vs. accepted traffic (the load that is injected into the network in Gbps/host) for all three traffic patterns, for networks with 256 switches located in 64 cabinets. Our EBD interconnection networks, named as “Skywalk” in the figures, achieves latency lower than that of Random for the same degree. Only Dragonfly, resp. HyperX, has latency lower than Skywalk but with a much higher degrees of 19, resp. 17, when compared to Skywalk’s degree of at most 11 in these results[17].

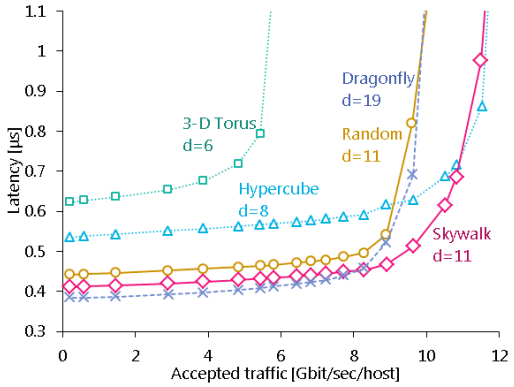

Figure 7. Average latency vs. throughput (results are taken from our previous work [17]). "Skywalk" is the design for our EBD interconnection networks.

6. Proxy EBD Applications

6.1. Metagenome Analysis

Today, the production of biological and medical data has been rapidly increased year by year. Particularly, the increase of genomic data is outstanding because of the improvement of DNA sequencers, which are generally called next-generation sequencers. The latest sequencer, Illumina's HiSeq 2500, can produce several hundred billions of base pairs (bp) of sequence data on a single run of the machine. The throughput of the system is approximately 10,000 times higher than that of previous sequencers. However, current most sequencers only produce information in short fragments of DNA sequences called reads. Thus, it is necessary to perform various analyses based on the DNA reads on a computer for obtaining biologically useful information. The current genome analyses require not only computation power but also huge storage and I/O performance because the analyses have to read huge amount of queries and a large reference sequence database and write the results whose size is often equal to or more than that of queries.

6.1.1. Emerging requirements on metagenome analysis

Metagenome analysis is the study of the genomes of uncultured microbes obtained directly from microbial communities in their natural habitats. The analysis is useful for not only understanding symbiotic systems but also watching environment pollutions [36]. However, the analysis has to deal with multiple organisms simultaneously and thus the size of the query sequences become extremely big data. It is often hundreds times larger than that of general genome analysis. In addition, metagenome analysis requires comparisons of sequence data obtained from a sequencer with sequence data of remote homologues in databases. Therefore, sensitive sequence homology searches and huge reference sequence database, which contains genome sequences of all known organisms, are required in metagenome analysis. As a result, comparison of EBD (queries) versus EBD (reference databases) is required, and there are serious problems both in the requirement of computation power and I/O performance.

For dealing the requirement of computation power on metagenome analysis, we have already developed several novel sequence search algorithms and tools for metagenome analysis. GHOSTM is a GPU-accelerated homology search tool and achieved calculation speeds that were 130 times faster than BLAST [28]. GHOSTX employs suffix array-based fast algorithm and achieved approximately 131-165 times acceleration over BLAST at similar levels of sensi-

tivity [29]. However, by using the fast search algorithm, the problem of I/O performance had become more serious. Figure 8 shows the speedup of the GHOSTM-based metagenome analysis on TSUBAME2.0 of Tokyo Institute of Technology. The speedup drastically deceases at 2,520 GPUs because of insufficient load balancing and I/O performance of the system [27]. The result indicated the more sophisticated system especially for I/O handing is required for large-scale metagenome analysis.

Now, we are developing a novel metagenomic data analysis pipeline on TSUBAME2.5 and K computer at the RIKEN Advanced Institute for Computational Science. To improve the load balancing and I/O processes, we avoided using a simple batch job system, and developed "mpidp" that is a versatile job distribution framework based on Master-Worker model. The system, GHOST-MP, has already showed clearly better scaling than the previous system on TSUBAME2.5 (Figure 9)

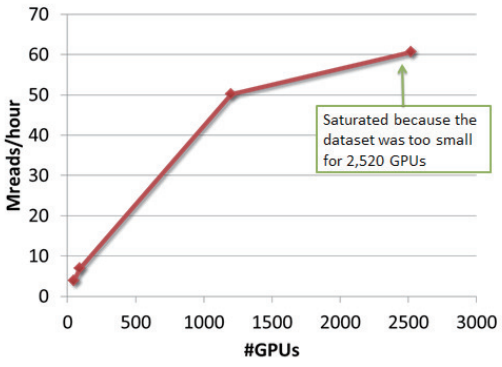

Figure 8. Speedup of GHOSTM-based pipeline on TSUBAME2.0

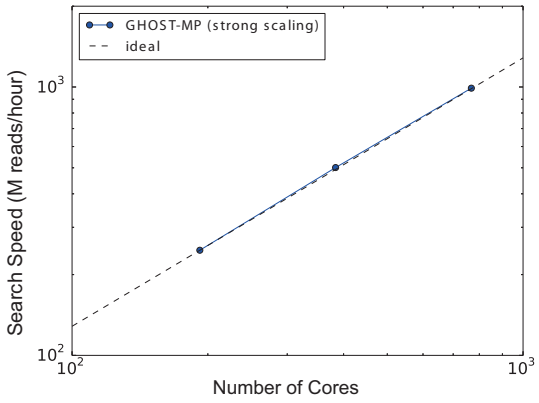

Figure 9. Speedup of GHOST-MP based pipeline on TSUBAME2.5

6.1.2. Metagenome analysis of human oral microbiome

We are now analyzing huge amount of human oral microbiome metagenome data released by human microbiome project (HMP) [19] on K computer. The data consists of 418 samples from various parts of human oral cavity, and includes approximately 26 billion reads, which is more than 5Tb (base pairs) (Table 1). For every query read sequences, we have performed sensitive sequence homology searches to well-annotated reference sequence database (KEGG genes) and revealed evolutionally related proteins. Several biological analyses of the results, which include analysis of active genes on metabolic pathways (Figure 10), have been finished but the most of analyses are now ongoing.

6.1.3. Novel APIs for EBD processing

Through the large-scale metagenome analysis on K computer and TSUBAME2.5, the bottlenecks of the pipeline have been gradually emerged. In addition, the "mpidp" is a simple tool and is not encapsulated. To use the current "mpidp", the users have to write additional codes and instruct the disk accesses. Therefore, we are now designing novel APIs for processing EBD such as metagenome data. Currently, we focus on metagenome analysis. However, the other applications such as protein-protein interaction prediction [25] and *in silico* screening of

Table 1. The detail of human oral metagenome data used in the analysis

Subsite	# samples	# M reads	# base pairs (Gb)
Saliva	5	278	56
Keratinized gingiva	6	361	73
Buccal mucosa	121	7478	1485
Hard plate	1	54	11
Palatine tonsils	6	373	74
Subgingival plaque	8	517	104
Supragingival plaque	128	7965	1595
Throat	7	393	79
Tongue dorsum	136	8708	1743
Total	418	26131	5220

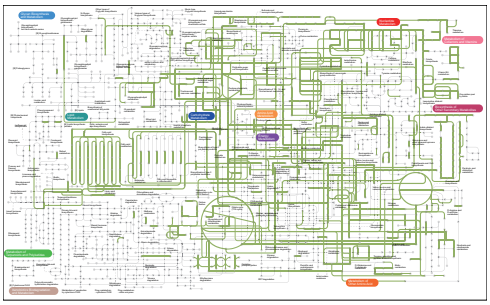

Figure 10. Active pathways in whole supragigival plaque samples

drug compounds also have to read and write huge amount of data, and importantly require a cross matching between EBD (query) versus EBD (reference). We will also perform requirement analysis for these applications in order to make the APIs be more universal.

6.2. Large-Scale Graph Analytics and Billion-Scale Social Simulation

This section introduces some of the example applications handling extremely big data with supercomputers such as large-scale network analysis, X10-based large-scale graph analytics library, Graph500 benchmark, and billion-scale social simulation.

6.2.1. ScaleGraph: X10-Based Large-Scale Graph Analytic Library

Recently, large-scale graph analytics has become a very popular topic owing to the emergence of gigantic graphs whose number of vertices and edges is in millions, billions or even trillions. Many graph analytics libraries and frameworks have been proposed with various computational models and programming languages to deal with such graphs. X10 programming language is a PGAS language that aims at both software performance and programmer's productivity. We introduce ScaleGraph library [7] developed using X10 programming to illustrate the use of X10 for large-scale graph analytics. ScaleGraph library provides XPregel framework that is inspired by Google's Pregel computation model, serving as a building block for implementing graph kernels. We also optimized X10 runtime in some parts such as collective communication, String

data representation, data structure, and file IO, to achieve further performance. As of this writing, the library supports varous parallel and distributed graph kernels such as PageRank, spectral clustering, degree distribution, betweenness centrality, HyperANF, strongly-connected component, maximum flow, single source shortest path, and breadth first search.

We evaluated the performance and scalability of all graph kernels available in ScaleGraph libraries on top of the TSUBAME 2.5 supercomputer with up to 128 nodes. The result shows that most kernels could solve billion-scale artifical data following social network structures and also achieve moderate scalability. We also performed the evaluation using the Twitter network as 2012/10 whose graph is composed of 496 millions of users and 28.5 billions of following relationships. The library successfully handles such a gigantic graph on 128 of machine nodes. To the best of our knowledge, ScaleGraph is the first X10-based library to address performance, scalability and productivity issues in dealing with large-scale graph analytics.

6.2.2. Graph500 Benchmark

As an alternative to Linpack, Graph500 [5] was recently developed. We conducted a thorough study of the algorithms of the reference implementations and their performance in an earlier paper [32]. Based on that work, we implemented a scalable and high-performance implementation of an optimized Graph500 benchmark for large distributed environments [37][38]. In contrast to the computation-intensive benchmark used by TOP500, Graph500 is a data-intensive benchmark. It does breadth-first searches in undirected large graphs generated by a scalable data generator based on a Kronecker graph. There are six problem classes: toy, mini, small, medium, large, and huge. Each problem solves a different size graph defined by a Scale parameter, which is the base 2 logarithm of the number of vertices. For example, the level Scale 26 for toy means 226 and corresponds to 1010 bytes occupying 17 GB of memory. The six Scale values are 26, 29, 32, 36, 39, and 42 for the six classes. The largest problem, huge (Scale 42), needs to handle around 1.1 PB of memory. As of this writing, Scale 40 is the largest that has been solved by a top-ranked supercomputer. Our work [37] proposed an optimized method based on 2D partitioning and other methods such as communication compression and vertex sorting. Our optimized implementation can handle BFS (Breadth First Search) of a large graph with Scale 35 with 462.25 GE/s while using 1366 nodes and 16,392 CPU cores. This competition is greatly challenging since new scalable algorithms have been proposed rapidly. We have been continuously enhancing the scalable algorithm and implementation on various supercomputers.

6.2.3. Billion-Scale Social Simulation

We introduce billion-scale social simulation [30][31] in this section. Towards the contribution to the human society, global economy, ecology, the analysis of human brain characteristics and our daily life, the research in multi-agent simulation is entering into the era of simulating billion-scale agents. Although prior arts tackle distributed agent simulation platform to achieve this goal, it is not sufficient to simulation billion-scale agent behaviors. Based on this observation, we report the first effort for building such an infrastructure platform that handles billion-scale agent simulation platform. In our previous work, we introduce X10-based agent simulation platform for such a purpose and presents its application to traffic simulation. We were able to handle only at maximum 10 millions of agents, but the performance was not scalable due to various reasons such as work load imbalance, global synchronization. Our work in [31] present the

work of purely implementing the whole simulation stack including both the simulation runtime and the application layer such as traffic simulation by the use of the state-of-the-art PGAS language. By implementing the system in such a manner and evaluating the system in highly distributed systems, it is observed that the system can be close to handle billion-scale agents in near real-time. The first experimental result is that the performance scalability is greatly achieved by simulating 1 millions of agents on 1536 CPU cores and 256 nodes in the TSUBAME 2.5 supercomputer. By compiling fully X10-based agent simulation system into C++ and MPI, it only takes 77 seconds for 600 simulation steps which is nearly 10 times faster than real-time. Moreover, by using the entire whole country-wide network of India as the agents underlying infrastructure, we successfully simulated 1 billion agents with 400 nodes. This is the first attempt to deal with such a gigantic number of agents and we believe that this infrastructure would be the basis of large-scale agent simulation in various fields.

6.3. Big Data Assimilation in Weather Forecast Applications

Contemporary weather forecasting relies on numerical simulations, and the accuracy of numerical weather prediction (NWP) is very sensitive to the accuracy of initial conditions or estimates of the current state of the atmosphere. Data assimilation provides the optimal combination of numerical simulations and observational data based on statistical mathematics and finds accurate initial conditions for more accurate forecasts.

The EBD issue arises in NWP due to the rapid increase of data volume in numerical simulations and observations. With leading-edge high-performance computing and sensing technologies, we may design an extremely demanding NWP system by a factor of 100 or more, that is what we call the "Big Data Assimilation (BDA)" revolution. We foresee that the future of NWP would be directed to the BDA, that is considered to be an important EBD application.

The rapid development of high-performance computing technologies enables higher-resolution simulations, which in turn enable an effective use of dense and frequent observations that have not been used well previously. In fact, dense observations from satellites are thinned if the numerical simulations cannot represent the small-scale phenomena captured by satellites. Also, new sensing technologies enable new types of sensors that produce data at a rate greater by two orders of magnitude than the current sensors. For example, the phased array weather radars (PAWR) implemented at Osaka and Kobe provide three-dimensional scanning of raindrops every 30 seconds at a 100-m resolution with 100 elevation angles, these numbers compared with about 15 elevation angles in 5 minutes or so in conventional Doppler radars. The next-generation geostationary satellite "Himawari-8" of the Japan Meteorological Agency (JMA) is capable of scanning a few limited areas in 30 seconds, compared with the current rapid scan in 5 minutes.

The data produced by the new types of sensors are extremely rapid and useful for capturing smaller-scale phenomena. Severe weather events may be caused by a single cumulonimbus cloud at a few km scale with lifetime of the order of 10 minutes. For such short phenomena, the linear tangent assumption made in most data assimilation methods would be valid only in the order of less than a minute. This motivates us to design a super-rapid NWP system using the every-30-second data from the new types of sensors. At this moment, only the leading-edge supercomputers can run simulations at a 100-m or higher resolution to make use of the dense and frequent data. About 2 Peta floating point operations are required to run 100 30-second simulations at a 100-m resolution. 100 simulations provide an approximate representation of the forecast uncertainties in the local ensemble transform Kalman filter [20], an advanced

data assimilation algorithm. We may spend only 10 seconds for the 100 simulations for timely forecasts, requiring 200 TFLOPS effective performance; this is possible using a quarter of 10-Peta-FLOPS "K computer" if we assume relatively optimistic 10 % efficiency.

The 100 simulations produce 200 GB data, that are transferred to the data assimilation module in a few seconds. Data assimilation merges the 200 GB forecast data with observations in the past 30 seconds, and produces another 200 GB data for the next 30-second forecasts. The tandem forecast-assimilation procedure are repeated every 30 seconds as new data keep coming so frequently.

To build the revolutionary rapid-update NWP system with high reliability, a co-design of software and hardware is essential. The rapid data transfer between forecast and data assimilation modules requires sufficient hardware capacity with specialized software design. Also, we expect that massively parallel high-performance EBD hardware would fail at a certain rate, while the data will keep coming every 30 seconds. In case of hardware failure, we may be forced to skip several steps of data assimilation. We could take advantage of four-dimensional LETKF [20], in which the past data are treated simultaneously in a single data assimilation step. Namely, we could skip several steps during the hardware failure, but still keep all data used effectively.

The BDA system with 30-second update cycles is two orders of magnitude more rapid than the currently used NWP systems. The operational global NWP is updated typically every 6 hours for synoptic weather at 2000 km scale. At the highest frequency for mesoscale weather at 20 to 200 km scale, forecasts are updated every hour in regional NWP systems including the Rapid Refresh (RAP) at the National Centers for Environmental Prediction (NCEP) and the Local Forecasting Model (LFM) at JMA. The 100-times more rapid processing of BDA is extremely demanding, but it is worth exploring such a rapid system to prevent and mitigate disasters caused by local severe weather such as heavy rainstorm and tornadoes.

7. Conclusions

We have introduced and gave an overview of our latest EBD (Extreme Big Data) project, which aims to achieve future convergence of big data and extreme supercomputing. By utilizing existing as well as future technologies gearing towards exascale, such as massively parallel processors, non-volatile memory, 3-D memory stacking, as well as high-bandwidth optics network, a new EBD convergent architecture can be constructed with which exabytes of data can be effectively processed. On top, a new system software stack that maximizes the underlying hardware to allow for massive data to be processed fast, including components such as bottom-level communication layer, distributed object management, basic algorithmic and programming frameworks, resource management, etc., is being developed, with considerations for exploiting other big data software framework. In order to assure that the EBD application requirements are being properly reflected, the system group is engaged in a co-design activity with three EBD application groups, each encompassing different type of application area as well as requirements. The results of the research will be deployed not only onto prototype hardware, but also production machines such as TSUBAME3.0, which is slated to be deployed in the first half of 2016 as one of the first "EBD" convergent production machine. As mentioned the project has only begun, still in its first year, but we have already achieved some notable results such as being world #1 on the Green Graph 500 list.

This research was supported by JST, CREST.

References

1. *Apache Hadoop*. http://hadoop.apache.org.
2. *Worldwide LHC Computing Grid*. http://wlcg-public.web.cern.ch/.
3. *OpenNVM*. https://opennvm.gitbug.io.
4. *The Adaptable IO System (ADIOS)*. https://www.olcf.ornl.gov/center-projects/adios/.
5. *Graph500*. http://www.graph500.org/.
6. *Green Graph500*. http://green.graph500.org.
7. *ScaleGraph Library*. http://www.scalegraph.org/.
8. *Sloan Digital Sky Survey*. http://www.sdss.org/.
9. *Spark*. http://spark.apache.org.
10. Dennis Abts and Bob Felderman. A guided tour of data-center networking. *Commun. ACM*, 55(6):44–51, 2012.
11. Jung Ho Ahn, Nathan Binkert, Al Davis, Moray McLaren, and Robert S. Schreiber. HyperX: topology, routing, and packaging of efficient large-scale networks. In *Proc. of the Conference on High Performance Computing Networking, Storage and Analysis (SC)*, pages 1–11, 2009.
12. Kevin Ashton. That 'internet of things' thing. *RFID Journal*, 2009.
13. Nathan Blow. Metagenomics: exploring unseen communities. *Nature*, 453 (7195):687–90, 2008. ISSN 1476-4687. URL `http://www.biomedsearch.com/nih/Metagenomics-exploring-unseen-communities/18509446.html`.
14. Quan Chen, Daqiang Zhang, Minyi Guo, Qianni Deng, and Song Guo. Samr: A self-adaptive mapreduce scheduling algorithm in heterogeneous environment. In *Proceedings of the 2010 10th IEEE International Conference on Computer and Information Technology*, CIT '10, pages 2736–2743, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4108-2. DOI: `10.1109/CIT.2010.458`.
15. Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, OSDI'04, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.
16. Jose Flich, Tor Skeie, Andres Mejia, Olav Lysne, Pedro Lopez, Antonio Robles, Jose Duato, Michihiro Koibuchi, Tomas Rokicki, and Jose Carlos Sancho. A Survey and Evaluation of Topology Agnostic Deterministic Routing Algorithms. *IEEE Transactions on Parallel and Distributed Systems*, 23(3):405–425, 2012.
17. Ikki Fujiwara, Michihiro Koibuchi, Hiroki Matsutani, and Henri Casanova. Skywalk: a Topology for HPC Networks with Low-delay Switches. In *IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, May 2014.
18. Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, SOSP

'03, pages 29–43, New York, NY, USA, 2003. ACM. ISBN 1-58113-757-5. DOI: 10.1145/945445.945450.

19. The Human and Microbiome Project. A framework for human microbiome research. *Nature*, 486(7402):215–21, June 2012. ISSN 1476-4687. DOI: 10.1038/nature11209.

20. B. R. HUNT, E. KALNAY, E. J. KOSTELICH, E. OTT, D. J. PATIL, T. SAUER, I. SZUNYOGH, J. A. YORKE, and A. V. ZIMIN. Four-dimensional ensemble kalman filtering. *Tellus A*, 56(4):273–277, 2004. ISSN 1600-0870. DOI: 10.1111/j.1600-0870.2004.00066.x.

21. Ryo Mizote Yuichiro Yasui Katsuki Fujisawa Keita Iwabuchi, Hitoshi Sato and Satoshi Matsuoka. Hybrid bfs approach using semi-external memory. In *International Workshop on High Performance Data Intensive Computing (HPDIC2014)*, May 2014.

22. John Kim, Wiliam J. Dally, Steve Scott, and Dennis Abts. Technology-Driven, Highly-Scalable Dragonfly Topology. In *Proc. of the International Symposium on Computer Architecture (ISCA)*, pages 77–88, 2008.

23. Michihiro Koibuchi, Hiroki Matsutani, Hideharu Amano, D. Frank Hsu, and Henri Casanova. A Case for Random Shortcut Topologies for HPC Interconnects. In *Proc. of the International Symposium on Computer Architecture (ISCA)*, pages 177–188, 2012.

24. Satoshi Matsuoka, Takayuki Aoki, Toshio Endo, Hitoshi Sato, Shin'ichiro Takizawa, Akihiko Nomura, and Kento Sato. TSUBAME2.0. In *Contemporary High Performance Computing*, Chapman & Hall/CRC Computational Science, pages 525–555. Chapman and Hall/CRC, April 2013. ISBN 978-1-4665-6834-1. DOI: doi:10.1201/b14677-23.

25. Yuri Matsuzaki, Nobuyuki Uchikoga, Masahito Ohue, Takehiro Shimoda, Toshiyuki Sato, Takashi Ishida, and Yutaka Akiyama. MEGADOCK 3.0: a high-performance protein-protein interaction prediction software using hybrid parallel computing for petascale supercomputing environments. *Source code for biology and medicine*, 8(1):18, January 2013. ISSN 1751-0473. DOI: 10.1186/1751-0473-8-18.

26. Koichi Shirahata, Hitoshi Sato, Toyotaro Suzumura, and Satoshi Matsuoka. A scalable implementation of a mapreduce-based graph processing algorithm for large-scale heterogeneous supercomputers. *Cluster Computing and the Grid, IEEE International Symposium on*, 0: 277–284, 2013. DOI: http://doi.ieeecomputersociety.org/10.1109/CCGrid.2013.85.

27. Shuji Suzuki, Takashi Ishida, and Yutaka Akiyama. An ultra-fast computing pipeline for metagenome analysis with next-generation dna sequencers. In *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, pages 1549–1550, Nov 2012. DOI: 10.1109/SC.Companion.2012.320.

28. Shuji Suzuki, Takashi Ishida, Ken Kurokawa, and Yutaka Akiyama. GHOSTM: a GPU-accelerated homology search tool for metagenomics. *PLoS One*, 7(5):e36060, January 2012. ISSN 1932-6203. DOI: 10.1371/journal.pone.0036060.

29. Shuji Suzuki, Masanori Kakuta, Takashi Ishida, and Yutaka Akiyama. GHOSTX: An improved sequence homology search algorithm using a query suffix array and a database suffix array. submitted.

30. Toyotaro Suzumura and Hiroki Kanezashi. Highly scalable x10-based agent simulation platform and its application to large-scale traffic simulation. In *Proceedings of the 2012*

IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications, DS-RT '12, pages 243–250, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-0-7695-4846-3. DOI: 10.1109/DS-RT.2012.44.

31. Toyotaro Suzumura and Hiroki Kanezashi. A holistic architecture for super real-time multiagent simulation platform. In *Winter Simulation Conference 2013*, 2013.

32. Toyotaro Suzumura, Koji Ueno, Hitoshi Sato, Katsuki Fujisawa, and Satoshi Matsuoka. Performance characteristics of graph500 on large-scale distributed environment. In *Proceedings of the 2011 IEEE International Symposium on Workload Characterization*, IISWC '11, pages 149–158, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-1-4577-2063-5. DOI: 10.1109/IISWC.2011.6114175.

33. Masahiro Tanaka and Osamu Tatebe. Pwrake: A parallel and distributed flexible workflow management tool for wide-area data intensive computing. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, pages 356–359, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-942-8. DOI: 10.1145/1851476.1851529.

34. Masahiro Tanaka and Osamu Tatebe. Workflow scheduling to minimize data movement using multi-constraint graph partitioning. *Cluster Computing and the Grid, IEEE International Symposium on*, 0:65–72, 2012. DOI: http://doi.ieeecomputersociety.org/10.1109/CCGrid.2012.134.

35. Osamu Tatebe, Kohei Hiraga, and Noriyuki Soda. Gfarm grid file system. *New Generation Computing*, 28(3):257–275, 2010. ISSN 0288-3635. DOI: 10.1007/s00354-009-0089-5.

36. Susannah G Tringe, Tao Zhang, Xuguo Liu, Yiting Yu, Wah Heng Lee, Jennifer Yap, Fei Yao, Sim Tiow Suan, Seah Keng Ing, Matthew Haynes, Forest Rohwer, Chia Lin Wei, Patrick Tan, James Bristow, Edward M Rubin, and Yijun Ruan. The airborne metagenome in an indoor urban environment. *PloS one*, 3(4):e1862, January 2008. ISSN 1932-6203. DOI: 10.1371/journal.pone.0001862.

37. Koji Ueno and Toyotaro Suzumura. Highly scalable graph search for the graph500 benchmark. In *Proceedings of the 21st International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '12, pages 149–160, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-0805-2. DOI: 10.1145/2287076.2287104.

38. Koji Ueno and Toyotaro Suzumura. Parallel distributed breadth first search on gpu. In *High Performance Computing (HiPC), 2013 20th International Conference on*, pages 314–323, Dec 2013. DOI: 10.1109/HiPC.2013.6799136.

39. Matei Zaharia, Andy Konwinski, Anthony D. Joseph, Randy Katz, and Ion Stoica. Improving mapreduce performance in heterogeneous environments. In *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation*, OSDI'08, pages 29–42, Berkeley, CA, USA, 2008. USENIX Association.

40. Matei Zaharia, Dhruba Borthakur, Joydeep Sen Sarma, Khaled Elmeleegy, Scott Shenker, and Ion Stoica. Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling. In *Proceedings of the 5th European Conference on Computer Systems*, EuroSys '10, pages 265–278, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-577-2. DOI: 10.1145/1755913.1755940.