Chapter 19

High Performance Artificial Intelligence Systems Research Team

19.1 Members

Satoshi Matsuoka (Team Leader)

Jun Igarashi (Senior Scientist)

Aleksandr Drozd (Research Scientist)

Shweta Salaria (Postdoctoral Researcher)

Emil Vatai (Postdoctoral Researcher)

Toshio Endo (Senior Visiting Scientist)

Mohamed Wahib (Visiting Scientist)

Miquel Pericas (Visiting Scientist)

Akihiro Nomura (Visiting Scientist)

Tokyo Tech Matsuoka lab students

19.2 Overview of Research Activities

The High Performance Artificial Intelligence Systems Research Team focuses on convergence of HPC and AI, namely high performance systems, software, and algorithms research for artificial intelligence/machine learning. In collaboration with other research institutes in HPC and AI-related research in Japan as well as globally it seeks to develop next-generation AI technology that will utilize state-of-art HPC facilities, including Fugaku.

For the time being until Mar 31, 2022, team's research work is being sponsored by the JST-CREST "DEEP" AI Project "Fast and cost-effective deep learning algorithm platform for video processing in social infrastructure".

19.2.1 Research Topics

- Extreme speedup and scalability of deep learning: Achieve extreme scalability of deep learning in large-scale supercomputing environments including the Fugaku extending the latest algorithms and frameworks for deep learning.
- Performance analysis of deep learning: Accelerate computational kernels for AI over the state-of-the-art hardware architectures by analyzing algorithms for deep learning and other machine learning/AI, measuring their performance and constructing their performance models.

- Acceleration of modern AI algorithms: Accelerate advanced AI algorithms, such as ultra-deep neural networks and high-resolution GAN over images, those that require massive computational resources, using extreme-scale deep learning systems.
- Acceleration of HPC algorithms using machine learning: Accelerate HPC algorithms, systems, and applications using empirical models based on machine learning.

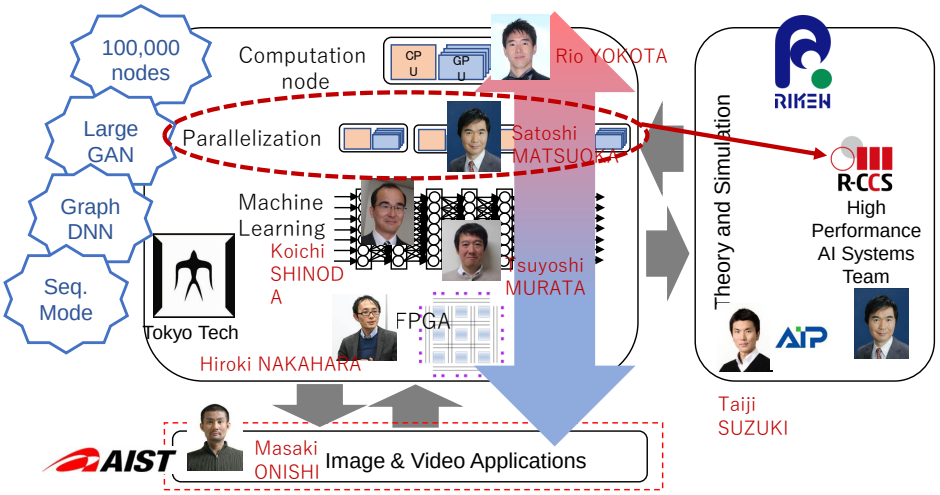


Figure 19.1: Structure of JST-CREST "DEEP" AI project

19.3 Research Results and Achievements

19.3.1 Massive Scale Deep Learning on Fugaku

The Fugaku supercomputer is an attractive platform for deep learning training and inference. A64FX CPU provided high FP16 and int8 compute performance as well as high memory bandwidth for memory-bound kernels such as certain implementations of convolutional operation. Additionally, high performance Tofu fabric enables efficient model-parallel and hybrid training schemes. We are exploring how unique features of the Fukagu supercomputer can be exploited in the most efficient ways in AI workloads, as well doing basic work to create deep learning software ecosystem on Fugaku. The later is done in collaboration with other research teams (Minami team and K. Sato team) as well as with industrial partners such as Arm, Fujitsu, and Linaro.

19.3.2 Accelerating DL with 2nd Order Optimization and Distributed Training

In this work, we proposed a large-scale distributed computational design for the second-order optimization using Kronecker-Factored Approximate Curvature (K-FAC)and showed the advantages of K-FAC over the first order stochastic gradient descent (SGD) for the training of ResNet-50 with ImageNet classification using extremely large mini-batches. We introduced several schemes for the training using K-FAC with mini-batch sizes up to 131,072 and achieved over 75% top-1 accuracy in much fewer number of epochs/iterations compared to the existing work using SGD with large mini-batch. Contrary to prior claims that second order methods do not generalize as well as SGD, we were able to show that this is not at all the case, even for extremely large mini-batches. Data and model hybrid parallelism introduced in our design allowed us to train on 1024 GPUs and achieved 74.9% in 10 minutes by using K-FACwith the stale Fisher information matrix (FIM). This is the first work which observes the relationship between the FIM of ResNet-50 and its training on large mini-batches ranging from 4K to 131K. There is still room for improvement in our distributed design to overcome the bottleneck of computation/communication for K-FAC – the Kronecker factors can be approximated more aggressively without loss of accuracy. One interesting observation is that, whenever we coupled our method with a well known technique that improves the convergence of SGD, it allowed us to approximate the FIM

more aggressively without any loss of accuracy. This suggests that all these seemingly ad hoc techniques to im-prove the convergence of SGD, are actually performing an equivalent role to the FIM in some way. The advantage that we have in designing better optimizers by taking this approach is that we are starting from the most mathematically rigorous form, and every improvement that we make is a systematic design decision based on observation of the FIM. Even if we end up having similar performance to the best known first-order methods, at least we will have a better understanding of why it works by starting from second-order methods. Further analysis of the eigenvalues of FIM and its effect on preconditioning the gradient will allow us to further understand the advantage of second-order meth-ods for the training of deep neural networks with extremely

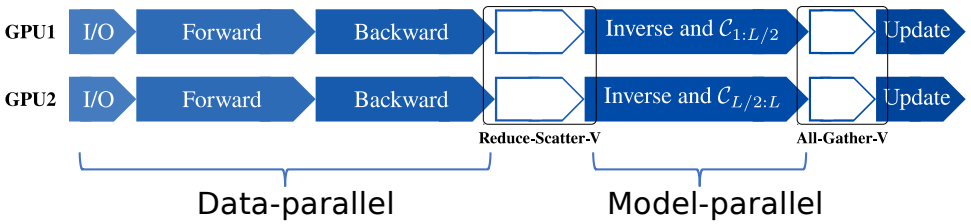


Figure 19.2: Design of hybrid parallel distributed K-FAC

19.3.3 A Software Systolic Array on GPUs

In this study, we build a software systolic array on the top of GPU architecture. Driven by the critical demands of intensive computation in a variety of fields, an old and heavily-researched parallel computer architecture, namely systolic array, is being revived in the post-Moore's law era, e.g. Google's Tensor Processing Unit (TPU), Nvidia's Tensor Core. Such a kind of domain-specific processors can provide extremely high TOPS (TeraOps/Second) and TOPS/Watt for compute-intensive algorithms, i.e. convolution, dense matrix multiplication, AI-purposed computation, Discrete Cosine Transform (DCT). In the last decade, we witnessed the emergence of Graphics Processing Units (GPU). As one of the most popular accelerators recently, GPUs are adopted to speed up the computation in a wide range of fields, i.e. scientific, engineering. Taking the latest TOP500 rankings in High-Performance Computing (HPC) for instance, more than half of the FLOPS comes from the Nvidia Tesla GPUs.

As Figure 19.3 shown, a systolic array is a network of processing elements (PEs) that rhythmically compute, accumulate and transfer data through the system. In typical systolic arrays, all of the PEs are often nested as a two-dimensional mesh. The systolic array architecture is simple yet very effective to achieve both computation and energy efficiencies with very limited memory bandwidth. Inspired by the mechanism of the hard-wired systolic arrays, we innovate a versatile execution model on the top of CUDA architecture for optimizing applications on GPUs. More specifically, we mimic the behavior of systolic arrays by CUDA warp, register files, and shuffle instruction. It is noteworthy that the three key techniques contribute to our model. First, register cache; second, partial sums accumulation and transfer; third, in-register computation. Our model improves the performance of regular memory-bound kernels by taking advantage of thread-private registers as a cache to perform the efficient in-register computation and employing the shuffle intrinsic to exchange partial sums between CUDA threads within a single CUDA Warp. Our model can be viewed as Software Systolic Array Execution Model (SSAM).

Regarding most of the scientific applications (or kernels), the increasing flops-to-bytes ratio of GPUs makes their computational performances bounded by the memory bandwidth. Memory-bound kernels that have a regular pattern of computation are particularly challenging since they appear to be simple, yet they require very complex data reuse schemes to effectively utilize the CUDA memory hierarchy, e.g. global memory, shared memory, register memory, etc. Typically, advanced GPU implementations for memory-bound kernels on structured grids rely on the optimized use of fast on-chip scratchpad memory: the programmer uses this user-managed scratchpad memory for reducing the global memory access. Indeed, there exists a plethora of work proposing variations and combinations of the three locality schemes that rely on scratchpad memory: spatial blocking, temporal blocking, and a wavefront pipeline. Those complex locality schemes enabled strides in performance improvements. However, they essentially moved the bottleneck from the global memory to the faster, yet smaller, scratchpad. The objective of this study is to yet again move the bottleneck from the scratchpad to a faster resource: register files. Hence, we prioritize to use register files to cache data rather than

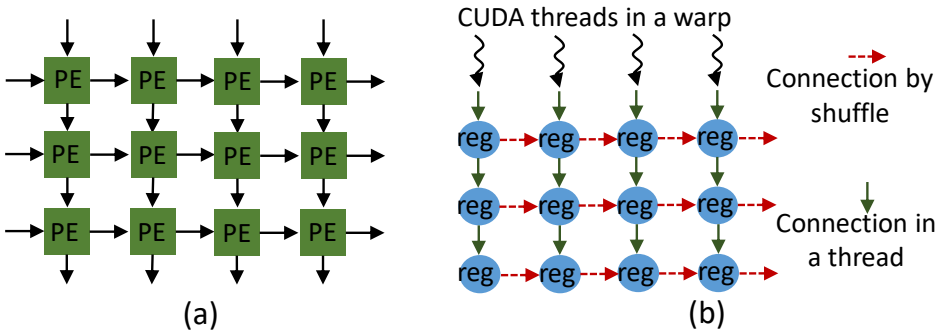
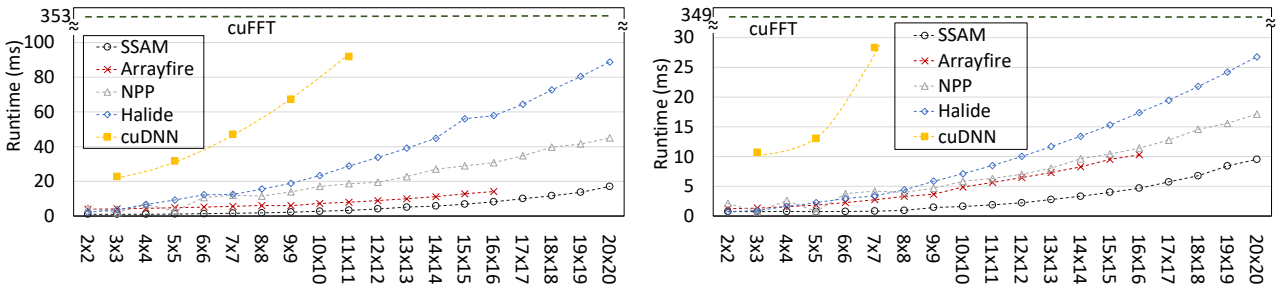


Figure 19.3: (a) Hardware 2D systolic array structure. PE is a processing element. (b) SSAM on CUDA (2D problem illustration: "reg" is a register). In the vertical direction, registers are in the same thread. In the horizontal direction, registers are exchanged by the shuffle instruction.



(a) Single precision on Nvidia Tesla P100 GPU. cuFFT method is constant as 353 ms.

(b) Single precision on Nvidia Tesla V100 GPU. cuFFT method is constant as 349 ms.

Figure 19.4: 2D Convolution performance and scalability. The image size is 8192×8192, the x-axis is filter size, the y-axis is execution time. Arrayfire, NPP, Halide, and cuDNN are state-of-the-art libraries.

shared memory, resulting in better data locality and higher computational efficiency.

A wide class of memory-bound kernels (or applications) can benefit from this execution model. In this study, we focus on mapping typical memory-bound kernels in SSAM, i.e. 2D convolution, 2D/3D stencils, Summed Area Tables (SAT). There is a strong motivation to optimize these kernels on GPUs. The 2D convolution and SAT computation become increasingly important in the Deep Learning workload. The stencil is a fundamental computation pattern in most of the scientific applications. To further improve the dependency graph in SSAM, we propose a novel algorithm to transpose the register cache locally (termed as BRLT). Unlike 2D convolution and stencils, we propose a collection of SSAM-based algorithms for SAT computation by the BRLT algorithm. To insight the performance advances of SSAM-based algorithms, we build performance models to analyze their mechanisms while using micro-benchmarking to measure some constant variables of GPUs.

The contributions in this study are as follows. *First*, we design and formulate a software systolic array on the top of CUDA architecture for speeding up the computation of memory-bound kernels with regular access on GPUs. *Second*, we analyze the data reuse and redundancy schemes to quantify the efficiency and limitations of SSAM. *Third*, we evaluate the proposed model for a wide variety of 2D/3D stencils, 2D general convolution (as in Figure 19.4), and SAT kernels on the latest Nvidia Tesla GPUs and demonstrate that SSAM-based algorithms outperform the top reported state-of-the-art libraries, e.g. NPP, Arrayfire, OpenCV. *Fourth*, we propose a novel algorithm to transpose the register matrix (used as register cache) locally and concurrently, resulting in improving the program dependency graph in SSAM for better locality and higher parallelism. *Finally*, we pave a novel way for code automation on GPUs due to the versatility and simplicity of SSAM for optimizing a large variant of algorithms.

19.3.4 Scaling Distributed Deep Learning Workloads beyond the Memory Capacity with KARMA

The dedicated memory of hardware accelerators can be insufficient to store all weights and/or intermediate states of large deep learning models. Although model parallelism is a viable approach to reduce the memory pressure issue, significant modification of the source code and considerations for algorithms are required. We propose a performance model based on the concurrency analysis of out-of-core training behavior, and derive a strategy that combines layer swapping and redundant recomputing. *KARMA* enables distributed training of DNNs beyond memory capacity. KARMA splits the layers to groups of blocks that optimize for reducing the total runtime. We reduce the total runtime by formulating a two-tier constrained optimization problem that maximizes the device occupancy, which in-turn requires an efficient strategy to reduce the stalls due to data movement to minimum. We use a novel scheduling strategy that involves a capacity-based layer swapping policy interleaved with recomputation. Next, KARMA generates an execution plan based on the identified optimal blocking and recompute strategy, and finally replaces the original model code with the new one.

Figure 19.5 shows the training performance (samples/second) of different batch sizes, for different models. As the batch size grows, the out-of-core effects start to appear. The performance begins to drop after the memory footprint exceeds the GPU memory capacity (starting from the second data point on each x-axis). We further conduct an experiment at which we use the same number of GPUs for the original implementation vs. data parallel KARMA (Figure 19.6). To get a fair comparison, we also compare to an optimized version of the original implementation for which we added the phased gradient exchange. Surprisingly, the pure data parallel KARMA outperforms the model-/data-parallel hybrid on 2,048 GPUs. Upon inspection it became clear that increasing the numbers of GPUs also increases the communication cost for the original version. Note that KARMA has fewer iterations (i.e. communication rounds) since it has a larger mini-batch size.

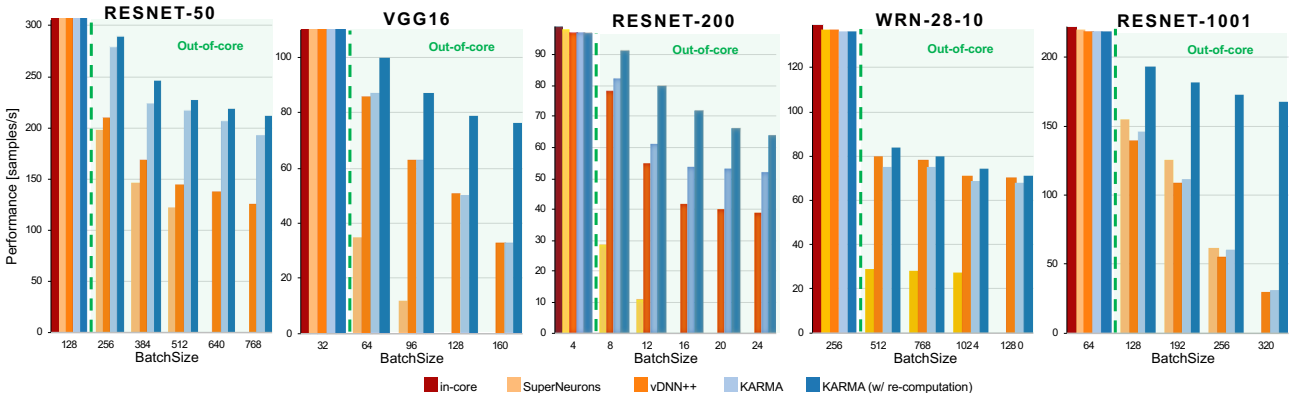


Figure 19.5: Performance using a V100 SMX2 (16*GB*) GPU. For all figures, only the first reported mini-batch size (x-axis) fits in memory for all models (state-of-the-art recompute strategy)

19.3.5 Optimizing Collective Communication in DL Training

Training models on large-scale GPUs-accelerated clusters are becoming a commonplace due to the increase in complexity and size in deep learning models. One of the main challenges for distributed training is the collective communication overhead for large message sizes: up to hundreds of MB. We have developed two hierarchical distributed memory multileader AllReduce algorithms optimized for GPU-accelerated clusters (named lr_lr and lr_rab), in which GPUs inside a computing node perform an intra-node communication phase to gather and store results of local reduced values to designated GPUs (known as node leaders). Node leaders then keep a role as an inter-node communicator. Each leader exchanges one part of reduced values to the leaders of the other nodes in parallel. Hence, we are capable of significantly reducing the time for injecting data into the inter-node network. We also overlap the inter-node and intra-node communication by implementing our proposal in a pipelined manner. We evaluate those algorithms on the discrete-event simulation Simgrid. We show that our algorithms, lr_lr and lr_rab , can cut down the execution time of an AllReduce microbenchmark that uses the logical ring algorithm (lr) by up to 45% and 51%, respectively. With the pipelined implementation, our lr_lr_pipe achieves 15% performance improvement when compared with lr_lr . In addition, the simulation result also projects power savings for the network devices of up to 23% and 32%.

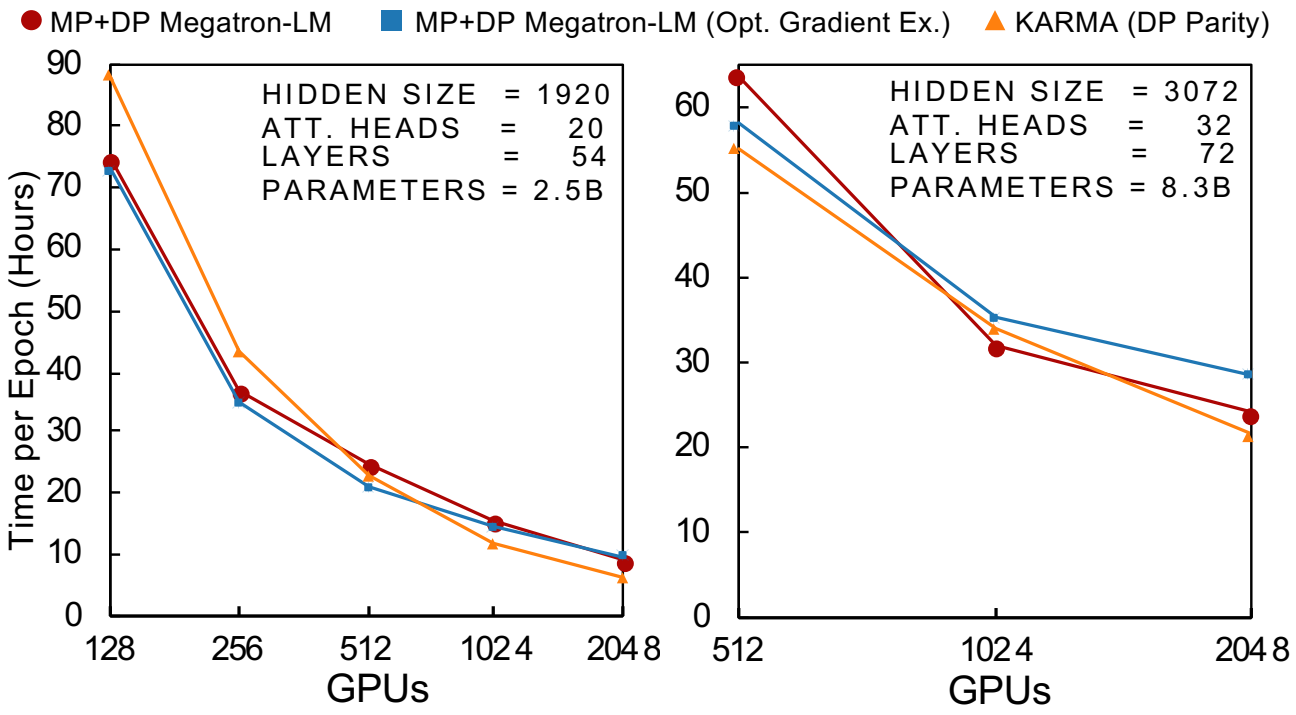
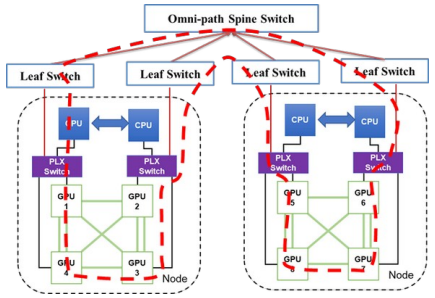
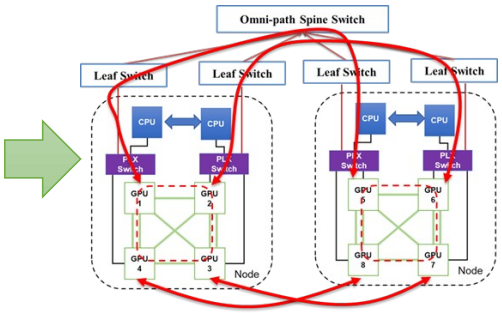


Figure 19.6: Parallelization performance of two Megatron-LM configurations. We compare using the same number of GPUs (parity[1]). Megatron-LM: we compare the original data-/model-parallel hybrid, the original plus our optimized phased gradient exchange, and data parallel KARMA



a) Ring-based algorithm: good for large message size but worse with inter-node comm.



b) Multileader hierarchical algorithm: optimized for inter-node comm.

Figure 19.7: Multileader hierarchical reduction algorithm

19.3.6 Training Large 3D CNNs with Hybrid Parallelism

This work is conducted among Yosuke Oyama (Satoshi Matsuoka lab, Tokyo Tech), Center for Applied Scientific Computing (CASC) at Lawrence Livermore National Laboratory, and National Energy Research Scientific Computing Center (NERSC) at Lawrence Berkeley National Laboratory.

Recent advances in deep learning have demonstrated that deep neural networks, especially convolutional neural networks (CNNs), can solve a lot of real-world problems. Specifically, 3D convolutional neural networks have been applied to scientific workflows, including 3D medical images and simulation outputs, to perform end-to-end inference without prior knowledge of underlying research fields. However, the training of 3D CNNs requires much computational resources and time, especially GPU memory capacity, to store the networks' intermediate data due to their gigantic data volume. This problem worsens when higher-resolution data is fed to the networks, which can improve the inference accuracy of the networks. 3D CNNs can consume tens to

hundreds of gigabytes of memory, as exemplified in the cosmology and medical models evaluated in our work.

In our work, we present scalable hybrid-parallel algorithms for training large-scale 3D convolutional neural networks. Deep learning-based emerging scientific workflows often require model training with large, high-dimensional samples, which can make training much more costly and even infeasible due to excessive memory usage. We solve these challenges by extensively applying hybrid parallelism throughout the end-to-end training pipeline, including both computations and I/O. Our hybrid-parallel algorithm extends the standard data parallelism with spatial parallelism, which partitions a single sample in the spatial domain, realizing strong scaling beyond the mini-batch dimension with a larger aggregated memory capacity. We evaluate our proposed training algorithms with two challenging 3D CNNs, CosmoFlow and 3D U-Net. Our comprehensive performance studies show that good weak and strong scaling can be achieved for both networks using up 2K GPUs. More importantly, we enable training of CosmoFlow with much larger samples than previously possible, realizing an order-of-magnitude improvement in prediction accuracy.

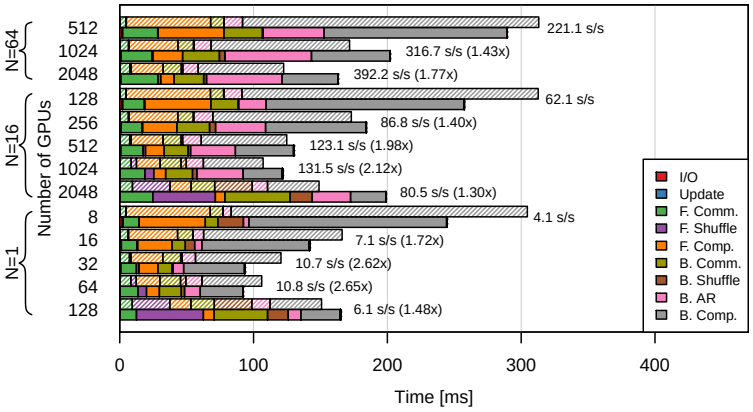


Figure 19.8: Strong scaling of the CosmoFlow network. Shaded bars show iteration time predicted by the performance model.

19.3.7 Advanced and Scalable NLP Models

Subword-level information is crucial for capturing the meaning and morphology of words, especially for out-of-vocabulary entries. We have proposed implementing linguistic compositionality using dedicated neural modules, based on CNN or RNN architecture. Additionally, we propose a hybrid training scheme in which a pure subword-level model is trained jointly with a conventional word-level embedding model based on lookup-tables. We have shown that morphological information can be captured efficiently by extremely compact models. Embeddings generated dynamically from just a few megabytes of parameters significantly outperform conventional (word2vec and FastText) models on morphology related tasks. Additionally, this indicates the vast limitation of the ability of conventional models to capture morphological information.

To model both morphological and semantic information, we have implemented two methods for combining strength of compact subword-level- and lookup-table based models: merging trained embeddings and training jointly. The resulting embeddings achieved high accuracy on a range of benchmarks and are particularly promising for datasets with high OOV rate.

We have showed that for languages with logographic scripts (fig. 19.10), such as Japanese, we can go even deeper than subword level and leverage sub-character information to improve accuracy of NLP applications with rare words.

Finally, we have identified that skewed distribution of different lexical elements in the training corpora is negatively affecting models' convergence dynamics under large minibatch sizes. We have introduced sample mining technique with which we prefetch more uniformly distributed training samples (with respect to involved linguistic phenomena) which allows us to train models with larger batch sizes and, in turn, enables better performance on a single GPU and scalability to multy-GPU clusters.

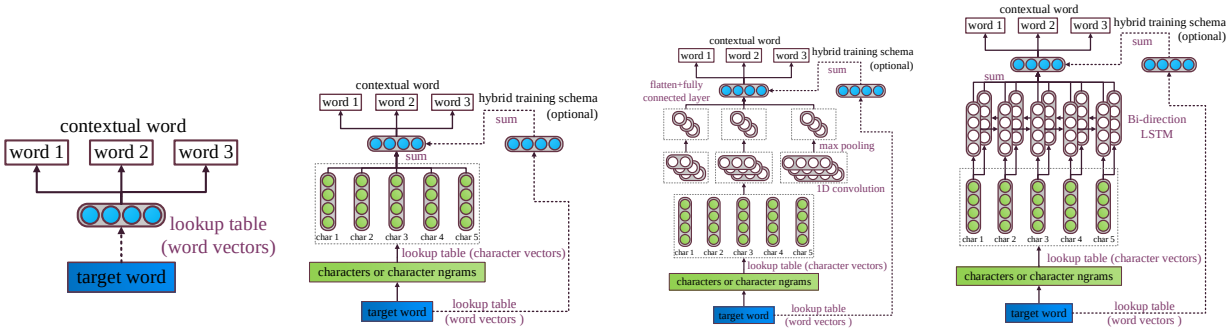


Figure 19.9: different neural nets as a compositional function

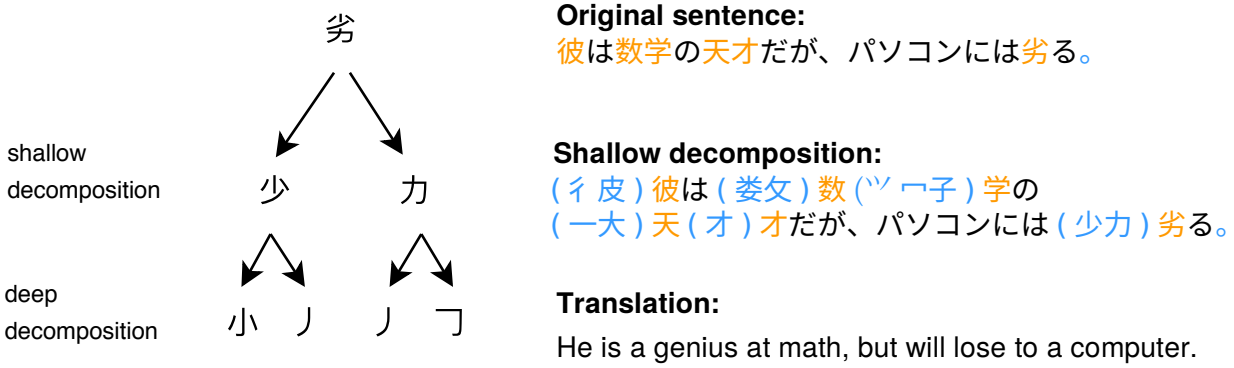


Figure 19.10: example of subcharacter information in Japanese characters

19.3.8 Performance evaluation of deep learning software and systems

The diverse tools used for the development and deployment of machine learning (ML) and deep learning (DL) applications makes benchmarking difficult. This diversity includes: different *hardware* (e.g. GPUs, TPUs, different CPUs such as A64FX of the Fugaku Supercomputer), multiple *frameworks* (e.g. TensorFlow, PyTorch), with numerous *backends* (e.g. BLAS, MKL, cuBLAS, cuDNN) executing different neural network neural network *arcitectures* (e.g. ResNet, BERT) running on various datasets synthetic or real (such as MNIST, ImageNet, COCO) and hyperparameters (e.g. batch size, precision).

We are developing *Benchmarker*, a modular benchmarking software, to tackle this problem. Benchmarker has well-defined interfaces, which enables easy addition of components at all levels mentioned above. It can generates logs in human- and machine-readable JSON format with detailed platform information. From these logs, it can generate visualisations, such as relative comparison over the any dimensions (frameworks, hardware, hyperparameter etc).

The purpose of such fine grained benchmarking, is to help pinpoint bottlenecks in the different implementations, but, more importantly, to provide insight into the reason behind the efficiency (or the lack of it) of certain implementations. Figure 19.11 shows the relative performance of 5 different deep learning models executed on 4 different hardware platforms with mixed and single precision.

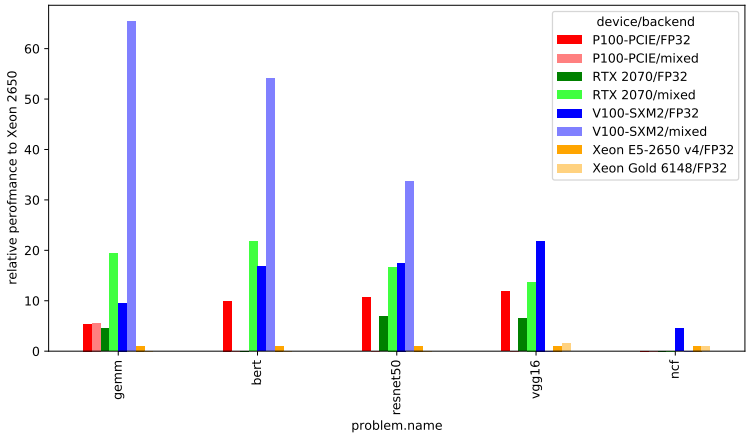


Figure 19.11: Relative performance of different models on various hardware

19.3.9 Predicting GPU Performance Using Collaborative Filtering

Graphical Processing Units (GPUs) are the de-facto source of performance in high performance computing. With the rapid increase in number and types of GPUs available, finding the best hardware accelerator for each application is a challenge. For that matter, it is time consuming and tedious to execute every application on every GPU system to learn the correlation between application properties and hardware characteristics. To address this problem, we use collaborative filtering to build an analytical model which can analyze and predict performance of applications across different GPU systems. Our model learns representations, or embeddings (dense vectors of latent features) for applications and systems and uses them to characterize the performance of various GPU-accelerated applications. We improve state-of-the-art collaborative filtering approach based on matrix factorization (MF) by building a multi-layer perceptron (MLP) as shown in figure 19.12.

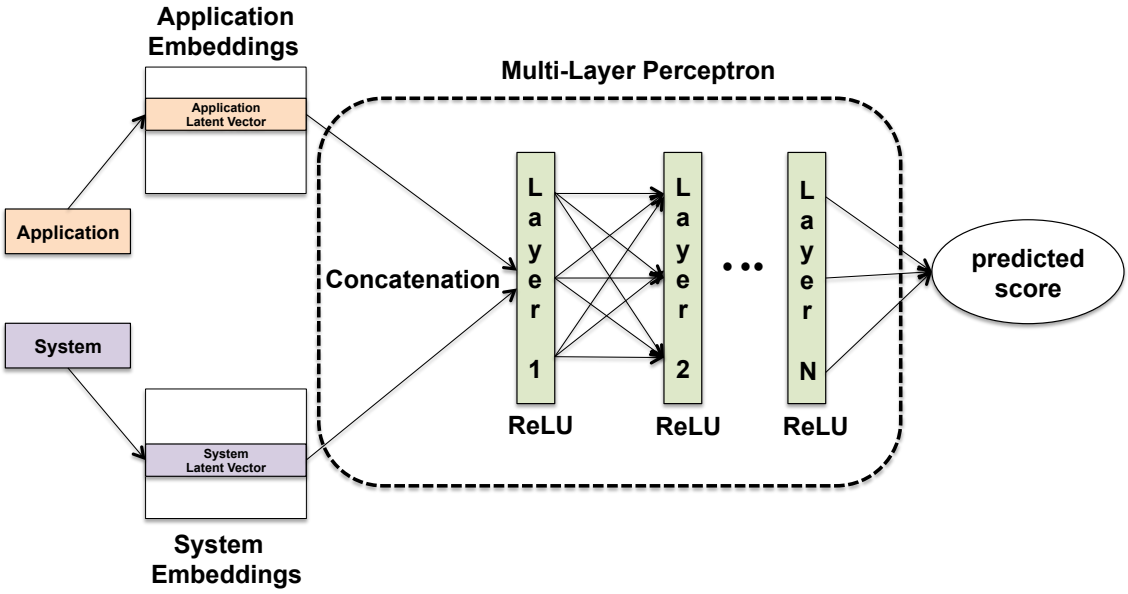


Figure 19.12: Multi-layer perceptron model using latent features

We evaluate our approach on a set of 30 well-known micro-applications and seven Nvidia GPUs of multiple generations. Figure 19.13 shows the prediction accuracy for MF and two variants of MLP when predicting instructions per second (IPS). As a result, we can predict expected IPS values with 90.6% accuracy in average.

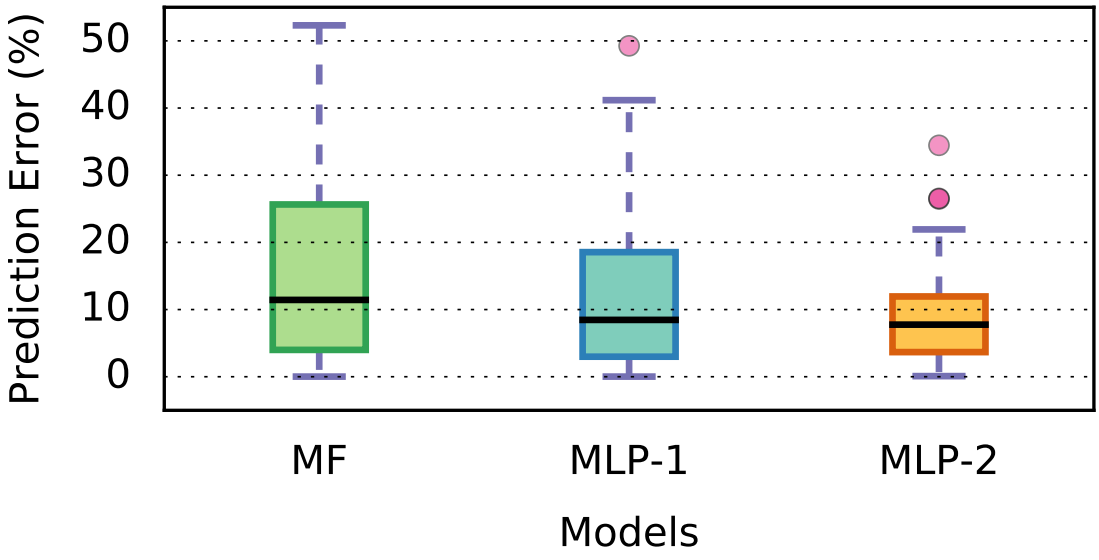


Figure 19.13: Prediction performance of MF, MLP-1 and MLP-2 using IPS dataset

19.4 Schedule and Future Plan

Activity within some of the existing research directions by the team are scheduled according to objective factors. For instance, creating deep learning ecosystem for Fugaku supercomputer is paced so that working solutions are ready for the general acceptance of the machine. Other directions are of more exploratory in nature and are being implemented depending on the current progress of the team and in the field in general. Such, withing natural language processing direction we incorporate large-scale experiments with transformer-based language models to reflect current trends in the field of NLP.

Within the general paradigm of the team, we are exploring new promising research direction as well. Two of such new directions are approximate computing and quantum computing.

19.5 Publications

19.5.1 Articles/Journal

[1] Nagasaka, Yusuke; Matsuoka, Satoshi; Azad, Ariful; Buluc, Aydin "Performance optimization, modeling and analysis of sparse matrix-matrix products on multi-core and many-core processors" Parallel Computing

[2] Bofang Li, Aleksandr Drozd, Yuhe Guo, Tao Liu, Satoshi Matsuoka and Xiaoyong Du "Scaling Word2Vec on Big Corpus"

19.5.2 Conference Papers

[3] Jens Domke, Kazuaki Matsumura, Mohamed Wahib, Haoyu Zhang, Keita Yashima,Toshiki Tsuchikawa, Yohei Tsuji, Artur Podobas and Satoshi Matsuoka "Double-precision FPUs in High-Performance Computing: an Embarrassment of Riches?" IPDPS'19

[4] Shweta Salaria, Aleksandr Drozd, Artur Podobas and Satoshi Matsuoka "Learning Neural Representations for Predicting GPU Performance"

[5] Yusuke Nagasaka, Akira Nukada, Ryosuke Kojima and Satoshi Matsuoka "Batched Sparse Matrix Multiplication for Accelerating Graph Convolutional Networks"

[6] Jens Domke, Satoshi Matsuoka, Ivan R. Ivanov, Yuki Tsushima, Tomoya Yuki, Akihiro Nomura, Shin'ichi Miura, Nic McDonald, Dennis L. Floyd and Nicolas Dube "The First Supercomputer with HyperX Topology: A Viable Alternative to Fat-Trees?"

[7] Chen Peng, Wahib Mohamed, Takizawa Shinichiro and Matsuoka Satoshi “A Versatile Software Systolic Execution Model for GPU Memory Bound Kernels”

[8] Chen Peng, Wahib Mohamed, Takizawa Shinichiro and Matsuoka Satoshi “iFDK: A Scalable Framework for Instant High-Resolution Image Reconstruction”

[9] Hamid Reza Zohouri, Satoshi Matsuoka “The Memory Controller Wall: Benchmarking the Intel FPGA SDK for OpenCL Memory Interface”

[10] Kazuki Oosawa, Youhei Tsuji, Yuichiro Ueno, Akira Naruse, Rio Yokota and Satoshi Matsuoka “Second-order Optimization Method for Large Mini-batch: Training ResNet-50 on ImageNet in 35 Epochs”

[11] Youhei Tsuji, Kazuki Oosawa, Yuichiro Ueno, Akira Naruse, Rio Yokota, Satoshi Matsuoka “Performance Optimizations and Analysis of Distributed Deep Learning with Approximated Second-Order Optimization Method”

[12] Hideyuki Jitsumoto, Yuya Kobayashi, Akihiro Nomura, Satoshi Matsuoka “MH-QEMU: Memory-State-Aware Fault Injection Platform”

19.5.3 Posters

[13] Yosuke Oyama, Naoya Maruyama, Nikoli Dryden, Peter Harrington, Jan Balewski, Satoshi Matsuoka, Marc Snir, Peter Nugent, Brian Van Essen “Toward Training a Large 3D Cosmological CNN with Hybrid Parallelization” ICPP 2019 Kyoto, Japan

[14] Haoyu Zhang,Wahib Mohamed, Pen Chen, Satoshi Matsuoka “Can Local Binary Convolutions Make Neural Networks Models Smaller?” ICPP 2019 Kyoto, Japan

[15] Lingqi Zhang,Wahib Mohamed, Satoshi Matsuoka “Understanding the Overheads of Launching CUDA Kernels” ICPP 2019 Kyoto, Japan

[16] Chen Peng, Wahib Mohamed, Takizawa Shinichiro, Matsuoka Satoshi “High resolution Image Reconstruction on Super computers” GTC 2020 Online (COVID-19), Originally San Jose, CA, USA

[17] Ryan Barton, Wahib Mohamed, Artur Podobas, Satoshi Matsuoka “BITFLEX: A Dynamic Runtime Library for Bit-Level Precision Manipulation and Approximate Computing” HPCAsia 2020 Fukuoka, Japan

[18] Chen Peng, Wahib Mohamed, Takizawa Shinichiro, Matsuoka Satoshi “A Software Systolic Array on GPUs GTC 2020” Online (COVID-19), Originally San Jose, CA, USA

19.5.4 Invited Talks

[19] Satoshi Matsuoka, “Riken R-CCS, Fugaku and AI”, Shanghai Arm Symposium, 2019/07/11, Shanghai, China

[20] Satoshi Matsuoka, “Fugaku: Co-Designing the first ‘Exascale’ Supercomputer with Real Application Performance as the Primary Target”, French-Germany-Japan HPC Workshop, 2019/11/06, Tokyo, Japan

[21] Satoshi Matsuoka, “The first ”exascale” supercomputer Fugaku & beyond”, HPC China Keynote, 2019/08/22, Hohhot, Inner Mongolia, China

[22] Satoshi Matsuoka, “A64fx and Fugaku - A Game Changing, HPC / AI Optimized Arm CPU to enable Exascale Performance”, HPC User Forum, 2019/10/11, Edinburgh, UK

[23] Satoshi Matsuoka, “Arm A64fx and Post-K: Game Changing CPU & Supercomputer for HPC and its Convergence of with Big Data / AI”, Hyperion HPC User’s Forum, 2019/04/03, Santa Fe, New Mexico

[24] Satoshi Matsuoka, “Fugaku and its Facilities”, ICPP 2019 EEHC Workshop, 2019/08/05, Kyoto, Japan

[25] Satoshi Matsuoka, “スーパーコンピュータ「京」「富岳」の概要”, JST-CRD 「社会革新を先導する量子科学技術」ワークショップ, 2019/06/14, 東京

[26] Satoshi Matsuoka, "The first "exascale" supercomputer Fugaku – HPC, BD & AI", JST Singapore 10th Anniversary Symposium, 2020/01/10, Biopolis, Singapore

[27] Satoshi Matsuoka, "The first "exascale" supercomputer Fugaku & beyond", DoE Workshop on Modeling and Simulation (ModSim) 2019, 2019/08/15, Seattle, WA

[28] Satoshi Matsuoka, "Fugaku as the Centerpiece of Society5.0 Revolution", Multicore World 2020, 2020/02/20, Wellington, New Zealand

[29] Satoshi Matsuoka, "RIKEN R-CCS OpenACC Presentation", Annual OpenACC Meeting Presentation, 2019/09/02, Kobe, Japan

[30] Satoshi Matsuoka, "Fugaku: The first 'Exascale' Supercomputer with Real Application Performance as the Primary Target, and Towards the Future", PPAM 2019 Keynote, 2019/09/08, Bialystok, Poland

[31] Satoshi Matsuoka, "Fugaku as the Centerpiece of Society5.0 Revolution", 2nd Annual R-CCS Symposium Keynote, 2020/02/18, Kobe, Japan

[32] Satoshi Matsuoka, "Post-K: the first 'exascale' supercomputer for convergence of HPC and big data/AI Numerical algorithms for high-performance computational science", The Royal Society, 2019/04/09, London, UK

[33] Satoshi Matsuoka, "AI for HPC and HPC for AI: Bidirectional Convergence Efforts of HPC and AI on the Fugaku Supercomputer", Deep Learning on Supercomputers Workshop Keynote, Supercomputing 2019, 2019/11/18, Denver, Colorado, USA

[34] Satoshi Matsuoka, "Toward Scaling Deep Learning to 100,000 Processors – The Fugaku Challenge", ScalA 2019 Workship Keynote, Supercomputing 2019, 2019/11/18, Denver, Colorado, USA

[35] Satoshi Matsuoka, "Fugaku: Co-Designing the first 'Exascale' Supercomputer with Real Application Performance as the Primary Target", SPPEXA Final Symposium, 2019/10/23, Dresden, Germany

[36] Satoshi Matsuoka, "Fugaku and AI", HPC Summer School, 2019/07/08, Kobe, Japan

[37] Aleksandr Drozd, "Deep Learning Ecosystem for ARM-Based Flagship Supercomputer", Linaro Connect 2019, 2019/10/20, San Diego