Chapter 22

Application Tuning Development Unit

22.1 Members

Kazuo Minami (Unit Leader)

Akiyoshi Kuroda (Research & Development Scientist)

Kiyoshi Kumahata (Research & Development Scientist)

Kazuto Ando (Technical Staff I)

Keigo Nitadori (Technical Scientist)

22.2 Overview of Research Activities

In the case of the K computer, the collaboration between the system and the applications in use is the key to create innovative results. In order to maximize the performance and usability of both the system and such applications, the Application Tuning Development Unit conducted the following activities:

(1) Activities related to the RIKEN Center for Computational Science (R-CCS) software center

(2) Activities to establish a deep learning (DL) environment on the Fugaku supercomputer

(3) Effort for the Fugaku supercomputer development

22.3 Research Results and Achievements

The application tuning development team has been conducting performance evaluation and enhancement aimed at popularizing the applications developed by the RIKEN Center of Computational Science (R-CCS) research team (i.e., R-CCS software) from applications on the K computer and the Fugaku supercomputer. By improving and enhancing the software, we expect industries and communities that have not used the R-CCS software to start using this software. Furthermore, along with the above activities, we are also trying to systemize performance optimization technology. Improving application performance will lead to a shortening of the elapsed time, which makes it possible to use more computational resources. This will help in the more effective utilization of resources. We kept these factors in mind while carrying out the following tasks.

22.3.1 Activities related to the R-CCS software

The RIKEN R-CCS Software Center is developing and deploying high-quality software for numerous high-performance computers (HPCs), including the K computerand the Fugaku supercomputer. Up to now, we have focused on improving and using the following software product developed by R-CCS: NTChem.

22.3.1.1 Enhancement of the R-CCS software (NTChem)

This year, we implemented large-scale parallelization of NTChem in the R-CCS software. NTChem is the molecular chemistry software developed by RIKEN AICS. This application has been provided for the analysis of life science and material science and can be used to examine the electronic state of a material. The molecular orbital method and the density functional method are used in this software. We implemented a number of functions such as the various basis functions and several algorithms for acceleration of calculation. This application works well for calculations using the density functional theory that includes the relativistic effect.

Last year, the calculation of the linear optical response KAIN method was accelerated by the high parallelization of the I/O of the linear response time dependent density functional theory calculations. This revealed a new imbalance problem due to differences in atoms and the cutoff radius. This year, we attempted to improve the massive parallel performance by reducing the load-in balance of the DFT calculation by incorporating rearrangement in the atom division calculation, which was examined last year to solve this problem. Figure22.1 shows the effect of distributing the elapsed time of the hot spots D1Rho_Box and GGA_Box. The total execution time was up to 42.81 [s] for the asis stage and up to 40.78 [s] for the tuned stage, and the imbalance was smoothed and reduced the elapsed time by approximately 4.98%. Based on these results, it is expected that the ratio of hotspot subroutines will increase for larger input data and that the load balancing effect will further increase.

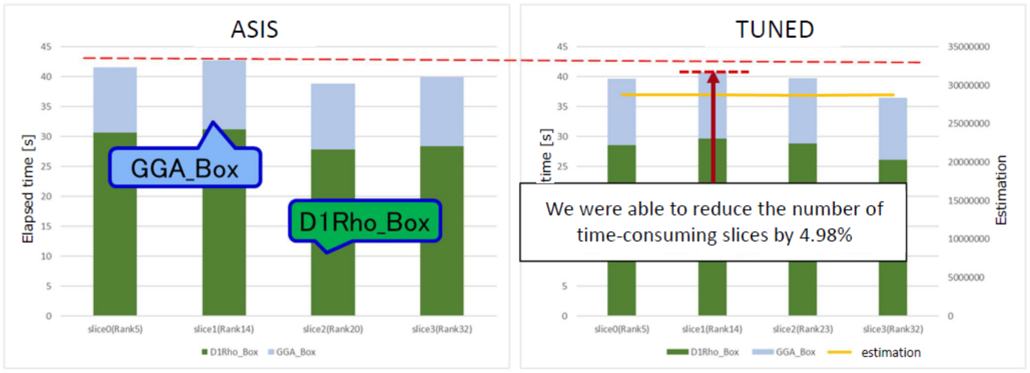


Figure 22.1: Improving imbalance for the KAIN method.

22.3.2 Activities to establish a deep learning environment on the K computer and Fugaku supercomputer

Recently, machine learning, especially deep learning (DL), has become commonplace. Deep learning calculation consists of a framework for flexibly handling complex network structures and a library (DNNL) for computing typical calculations with high performance. In addition, applications and research using graphics processing units (GPUs) are advancing in DL science fields. The GPU is said to be essential for DL. However, surveys up to last year have shown that even massively parallel computers, such as the K computer and the Fugaku supercomputer, can perform high-performance calculations by taking advantage of the massively parallel characteristics of the CPUs. Focusing on this point, we performed the following work this year, including the performance tuning of Chainer, which is a representative framework, with the goal of contributing to user convenience.

22.3.2.1 Building a development organization

The DL4Fugaku project, which began last year, has been further improved upon this year [8]. Within RIKEN, our unit cooperated to promote the project with the High Performance Big Data Research Team, led by Dr. Kento Sato, and the High Performance Artificial Intelligence Systems Research Team, led by Dr. Satoshi Matsuoka. We also collaborated continuing from last year with the Large-scale Parallel Numerical Computing Technology Research Team, led by Dr. Toshiyuki Imamura, to optimize the library used for caluculations such as convolution. The cooperation with ARM and Linaro has been strengthened, and development is underway with the goal of committing to MLPerf benchmarks. The development of DNNL was promoted as a result of the cooperation with Fujitsu and Fujitsu Laboratories. We continued to contact Preferred Networks, Inc.

and provided technical cooperation on the issue of high parallelization for the utilization of DL on the Fugaku supercomputer.

22.3.2.2 Performance evaluation and tuning of Chainer on the K computer

Continuing from last year, we optimized the performance of Chainer on the K computer this year. The MNIST problem required an original calculation time of approximately 10,300 [s]. Last year, we achieved a performance of 16.2× (635.2 [s]), and a performance efficiency of 15.8%. The floating point underflow and software pipelining inapplicability of Adam optimization were major factors in the efficiency decrease. This year, we found that a further factor is a non-parallel part, such as the Python control part. In addition, we could achieve a performance of 36.4× (283.4 [s]), and a performance efficiency of 35.9% using MPI process parallelization in a CPU. The results of the performance tuning were presented at a poster session [2,3,5,6]. We also provided and supported a tuned Chainer as R-CCS software.

22.3.2.3 Performance evaluation and tuning of Chainer on the FX100

The performance tuning performed on the K computer was applied to FX100, and we identified the problems. The ASIS state on the FX100 took 49,200 [s] to calculate the MNIST sample problem, because the provided NumPy was built using gcc and the SSL2 library could not be linked. By applying Python know-how to the K computer, we finally achieved a performance of approximately 1,030× (47.8 [s]), and a performance efficiency of approximately 13.5%.

22.3.2.4 Performance evaluation and tuning of Chainer on the ThunderX2

We evaluated and tuned the performance of Chainer on the ThunderX2 CPU, which was the previous chip used in the Fugaku supercomputer. The ASIS state on the ThunderX2 took 5,340 [s/core] to calculate the MNIST sample problem, but we finally achieved a performance of approximately 4.8× (1,100 [s/core]), and a performance efficiency of approximately 13.5%. Since the Fujitsu SSL2 library cannot be linked on the ThunderX2, convolution calculations are carried out with OpenBLAS. OpenBLAS tended to have good core performance but not thread scalability, so we evaluated the performance of processing parallelization. We also tuned the performance for the ImageNet problems on the ThunderX2. The calculation costs, such as the cost of batch normalization, have become large for ImageNet problems, so array calculations, which have large costs in NumPy, has been converted to Fortran and prepared as a kernel loop collection. The target kernels of the change were 10 files, 14 sections, and 83 lines, and there were only a few parts that needed to be modified. Based on this effect, ImageNet performance on the ThunderX2 achieved 0.748 [images per seconds (ips)] and a performance efficiency of 54.4% in core performance, and 11.95 [ips] and a performance efficiency of 31.0% in CPU performance.

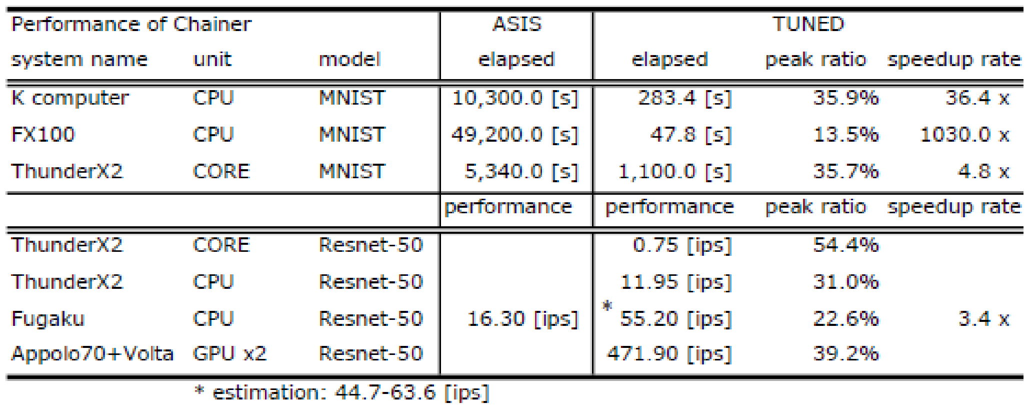
22.3.2.5 Performance evaluation and tuning of Chainer in the environment of the Fugaku early access program

We also evaluated the performance using the early access program environment of the Fugaku supercomputer. By combining the measurement results using one core Chainer overall calculation and thread parallel kernel performance using CMG, we estimated the performance to be approximately 44.7 to 63.6 [ips/cpu] on the Fugaku CPU. When the performance on the CPU was measured, the performance was approximately 16.3 [ips/cpu] in the first measurement. The cause of this performance difference was non-parallelism, such as the calculation of distribution, and we finally achieved a performance of 55.2 [ips/cpu] and a performance efficiency of 22.6% by rewriting and replacing these parts with Fortran kernels. Thus, the performance tuning of Chainer on the Fugaku supercomputer was almost completed, and it was then possible to promote the use of DL on the Fugaku supercomputer [Table 22.1]. In addition, the framework Chainer can now be provided to users on the Fugaku supercomputer. Since the development of Chainer will stop at Ver. 7, our performance tuning work will also be suspended.

22.3.2.6 Performance evaluation and tuning of Chainer on the Apollo70+Volta

We also evaluated the performance of the ThunderX2 machine with two Volta GPUs provided by NVIDIA as a test user. Since ChainerX cannot be executed in parallel, we used Chainer-6.5.0 for evaluation. The maximum performance was 471.9 [ips], and the efficiency was 39.2%. On the Fugaku CPU, this performance is expected to be approximately 102.7 [ips/cpu]. The performance of the Fugaku CPU is currently 55.2 [ips/cpu], but by replacing the convolution algorithm with a faster DNNL, we believe that approximately the same performance will be achieved.

Table 22.1: Summary of Chainer performance.



Performance of Chainer			ASIS	TUNED		
system name	unit	model	elapsed	elapsed	peak ratio	speedup rate
K computer	CPU	MNIST	10,300.0 [s]	283.4 [s]	35.9%	36.4 x
FX100	CPU	MNIST	49,200.0 [s]	47.8 [s]	13.5%	1030.0 x
ThunderX2	CORE	MNIST	5,340.0 [s]	1,100.0 [s]	35.7%	4.8 x
			performance	performance	peak ratio	speedup rate
ThunderX2	CORE	Resnet-50		0.75 [ips]	54.4%	
ThunderX2	CPU	Resnet-50		11.95 [ips]	31.0%	
Fugaku	CPU	Resnet-50	16.30 [ips]	* 55.20 [ips]	22.6%	3.4 x
Appolo70+Volta	GPU x2	Resnet-50		471.90 [ips]	39.2%	

* estimation: 44.7-63.6 [ips]

22.3.3 Power consumption of the job

At the end of the operation of the K computer, we cooperated in research evaluation on temperature, failure rate, and performance. We created a job using the following kernels, executed the job as needed in a resource group with a low priority, and investigated the relationship between temperature, computational load, and failure rate.

- A basic kernel in which FMA operates at full capacity and is 90
- A kernel that also has large memory access and high power consumption.
- Stride array copy kernel with maximum memory access.
- Sleeping kernel.
- Adventure kernel having high power consumption due to use in real applications.

Although there is a clear relationship between the change in temperature and the calculation efficiency, it was found that there is no relationship between the temperature and the failure rate, which was reported in a poster session [4,7].

22.3.4 Efforts in the development of the Fugaku supercomputer

The FLAGSHIP2020 (FS2020) Project is engaged in research and development for the Japanese national flagship supercomputer ”Fugaku”, which is a post K computer. As part of the FS2020 project, the application tuning development unit applied co-design of the application and Fugaku supercomputer system through the performance optimization and sophistication of target applications. In particular, our unit is responsible for these three applications.

- ADVENTURE: A structural analysis application based on the finite-element method in Priority Issue No. 6
- RSDFT: A first-principles material simulation and optimization application based on density functional theory, in Priority Issue No. 7
- FrontFlow/Blue: A fluid analysis application based on the finite-element method in Priority Issue No. 8

For these applications, we have paid significant attention to the following efforts:

- Application performance tuning

- Establishing methods for estimating application performance on the supercomputer Fugaku.
- Establishing test kernel codes for evaluating the CPU and system of Fugaku.

The following subsections describe application tuning.

22.3.4.1 Enhancement of the co-design application (ADVENTURE)

ADVENTURE, the structural analysis application based on the finite-element method, is composed of two major parts. One is a sub-domain solver, and the other is a coarse grid correction. The whole analysis domain is divided into multiple sub-domains. A sub-domain solver solves each sub-domain deformation. Coarse grid correction solves the inter-sub-domain relationship by representing a sub-domain as one point having six degrees of freedom. In both parts, a hot kernel is the multiplication of a dense matrix and a vector. Until last year, we continued to tune a hot kernel mainly by separating only the hot kernel source. In particular, we focused on the following issues:

- Avoiding unsuitable loop unrolling that caused cache thrashing to occur frequently
- Decreasing the number of data streams for effective cache memory utilization
- The essential procedure of multiplying a dense matrix and a vector is the taking the dot-product of two vectors. In order to apply SIMD instruction to the summation operation of the dot-product calculation effectively, we implemented a recursive summation operation and minimized the number of temporary arrays. Furthermore, we decreased the number of recursive summation operations to just one.

In previous years, the application used for evaluating performance on the Fugaku supercomputer was a limited version. For example, Fugaku omitted some functions and could only treat special input data. This year, we could evaluate a full-function application. Using this version, a successful run of a sample problem, namely, a 4,096-node-sized problem (16,384 processes, 65,536 sub-domains), was conducted on the Fugaku. Subsequently, in order to validate the performance on the Fugaku, timer routines and tuned kernel codes were implemented. As a result, on Fugaku supercomputer, the target problem achieved over 60× faster performance than the K computer. In addition, power consumption was significantly lower than estimated previously.

22.3.4.2 Enhancement of the co-design application (RSDFT)

The main procedures in RSDFT are DGEMM, and collective communications are based on message passing interfaces (MPIs), such as MPI_Allreduce and MPI_Bcast. Owing to performance tuning conducted during K computer development, RSDFT achieved a performance efficiency of approximately 43.6% for the Si nano-wire problem and won the Gordon-Bell Award in 2011.

This year, we combined of the diagonal priority algorithm kernel into the application and confirmed that there are no problems in calculation and performance. We also evaluated the scalability of CPMD using the K computer. By changing the process mapping, MPI_bcast communication in the band direction became four times faster, and scalability was improved. In addition, we verified the small-scale calculations of the full RSDFT using the prototype Fugaku CPU. As a result, two thread parallelization problems and a problem involving parameter setting were found, but all of these problems were solved. For the power consumption evaluation of tread1 region of the EigenExa library, which is concerned with exceeding the power consumption in boost mode, it was confirmed that the full application can be used on the prototype Fugaku CPU using the development version ScaLAPACK. In addition, we compared the performance of the DGEMM kernel with the evaluation results of last year and found performance differences in some parameters. The cause was found to be due to a short measurement time, and the same performance of the evaluation result of last year was obtained by expanding the measurement time.

22.3.4.3 Enhancement of the co-design application (FrontFlow/blue)

FrontFlow/blue is a fluid simulation code based on the finite-element method. Its major kernels are "kernels depending on memory bandwidth". The performance of these kernels is strongly dependent on the data transfer speed between the CPU and memory. Therefore, in order to tune the performance, it is necessary to obtain higher bandwidth rather than higher CPU calculation performance. In previous years, in order to effectively use the increased SIMD width of Fugaku, we performed a tuning in which list (or indirect) access storage, caused by the data structure of the finite-element method, was changed to sequential (or direct) access storage. Furthermore, last year we used the following tunings.

- Hiding the imbalance of cache memory latency, caused by the memory access pattern difference between threads, by prefetching the list access array.
- Obtaining maximum memory throughput using the SIMD prefetch instruction.
- Applying tuning for miscellaneous parts that were not considered in terms of performance.

Thus, the performance was significantly improved compared to the initial version, which is the version when starting co-design. For example, in the gradient calculation kernel, the elapsed time is improved from 27.4 [s] to 5.58 [s], the performance is improved from 11.18 [GFLOPS] to 59.29 [GFLOPS], and the memory throughput is improved from 53.6 [GB/s] to 195 [GB/s]. In addition, we contributed to environmental sophistication to enable standard use of the Fugaku supercomputer by identifying the problems listed below using FFB.

- Performance fluctuations
- Used memory increase corresponding to the calculated time

Finally, we submitted a paper for the ACM Gordon-Bell Prize, which was adopted as a finalist.

22.4 Schedule and Future Plan

22.4.1 R-CCS software center activities

Activities related to the following two areas should be continued for use with the Fugaku supercomputer after the service of the K computer ends this summer. One is application improvements from the viewpoint of both performance and usability, and the other is activities to promote the advancement of a utilization environment for R-CCS software users via demonstrations, tutorials, documents, and other methods. Although we focused on four R-CCS software products in our activities up to this point, it is undisputable that the focus could be expanded to additional software applications.

22.4.2 Activities to establish a DL environment on the Fugaku supercomputer

With regards to DL, we summarized the usage scenarios and performance levels of other computers. In the future, we intend to work to develop a high-performance DL library and scalable DL frameworks for use on the Fugaku supercomputer.

First, in order to improve the performance of the special DNN libraries developed by ARM Scalable Vector Extension (SVE), we intend to feed back knowledge gained during evaluations of the general DNN kernel prototyped herein. Furthermore, we intend to evaluate how much performance can be achieved for real training problems in order to build a general DNN kernel and special DNN library for use in a DL framework.

The elapsed time of a single convolution calculation depends on the DNN library performance, but the total training time of the scalable DL framework depends on the communication method. For this reason, the study of new communication algorithms will also be needed in order to facilitate high-performance training.

22.5 Publications

22.5.1 Articles/Journal

[1] Takahiro Yamasaki, Akiyoshi Kuroda, Toshihiro Kato, Jun Nara, Junichiro Koga, Tsuyoshi Uda, Kazuo Minami and Takahisa Ohno, "Multi-axis Decomposition of Density Functional Program for Strong Scaling up to 82,944 Nodes on the K Computer: Compactly Folded 3D-FFT Communicators in the 6D Torus Network", Computer Physics Communications, Vol.244, No.2019, (2019) pp.264-276, https://doi.org/10.1016/j.cpc.2019.04.008 .

22.5.2 Conference Papers

[1] Chisachi Kato, Yoshinobu Yamade, Katsuhiro Nagano, Kiyoshi Kumahata, Kazuo Minami, Tatsuo Nishikawa, "Toward Realization of Numerical Towing-Tank Tests by Wall-Resolved Large Eddy Simulation based on 32 billion grid Finite-Element", Finalist of ACM Gordonbel Prize for SC20, (2020).

[2] Hisashi Yashiro, Koji Terasaki, Yuta Kawai, Shuhei Kudo, Takemasa Miyoshi, Toshiyuki Imamura, Kazuo Minami, Hikaru Inoue, Tatsuo Nishiki, Takayuki Saji, Masaki Satoh and Hirofumi Tomita, ”A 1024-member ensemble data assimilation with 3.5-km mesh global weather simulations”, Finalist of ACM Gordonbel Prize for SC20, (2020).

22.5.3 Posters

[1] Kiyoshi Kumahata, Kazuo Minami, ”Co-Designing of FEM based CFD code FrontFlow/blue for the Supercomputer Fugaku”, HPC Asia 2020, Fukuoka, Japan, (2020).
http://sighpc.ipsj.or.jp/HPCAsia2020/hpcasia2020_poster_abstracts/poster_abstract_18.pdf
[2] Akiyoshi Kuroda, Kiyoshi Kumahata, Syuichi Chiba, Katsutoshi Takashina and Kazuo Minami, ”Performance Tuning of Deep Learning Framework Chainer on the K computer.”, Frontiers of Statistical Physics (FSP2019), Koshiba-Hall in Hongo Campus, The University of Tokyo, JAPAN, P-20 (2019.06.07-08).
[3] Akiyoshi Kuroda, Kiyoshi Kumahata, Syuichi Chiba, Katsutoshi Takashina and Kazuo Minami, ”Performance Tuning of Deep Learning Framework Chainer on the K computer.”, ISC High Performance (ISC2019), Research Posters/HPC in ASIA Research Posters, Frankfurt Messe, Germany, PR-28 (2019,06.18-19).
[4] Jorji Nonaka, Keiji Yamamoto, Akiyoshi Kuroda, Toshiyuki Tsukamoto, Kazuki Koiso, Naohisa Sakamoto, ”A View from the Facility Operations Side on the Water/Air Cooling System of the K Computer”, The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC19), Research Posters, Denver, USA, Poster 82 (2019.11.17-22).
[5] Akiyoshi Kuroda, Kiyoshi Kumahata, Syuichi Chiba, Katsutoshi Takashina and Kazuo Minami, ”Performance Tuning of Deep Learning Framework Chainer on the K computer.”, HPC Asia 2020, ACROS Fukuoka, Japan, No.32 (2020.01.15-17).
[6] Akiyoshi Kuroda, Kiyoshi Kumahata, Syuichi Chiba, Katsutoshi Takashina and Kazuo Minami, ”Performance Tuning of Deep Learning Framework Chainer on the K computer.”, The 2nd R-CCS International Symposium, Kobe, JAPAN, No.26 (2020.02.17-18).
[7] Jorji Nonaka, Toshiyuki Tsukamoto, Motohiko Matsuda, Keiji Yamamoto, Akiyoshi Kuroda, Atsuya Uno and Naohisa Sakamoto, ”A Brief Analysis of the K Computer by using the HPC Facility’s Water Cooling Subsystem”, The 2nd R-CCS International Symposium, Kobe, JAPAN, No.34 (2020.02.17-18).
[8] Kento Sato, Akiyoshi Kuroda, Kazuo Minami, Jens Domke, Aleksandr Drozd, Mohamed Wahib, Shuhei Kudo, Toshiyuki Imamura, Kiyoshi Kumahata, Keigo Nitadori, Kazuo Ando and Satoshi Matsuoka, ”DL4Fugaku: Deep learning for Fugaku - Scalability Performance Extrapolation -”, The 2nd R-CCS International Symposium, Kobe, JAPAN, No.42 (2020.02.17-18).

22.5.4 Invited Talks

[1] Kiyoshi Kumahata, ”Achievement of the priority issue No.8 -summary of FFB tuning-”, The 2nd meeting of application cooperation development, Institute of Industrial Science, The University of Tokyo, (2019).

22.5.5 Oral Talks

22.5.6 Software

22.5.7 Patents

22.5.8 Book

[1] Kazuo Minami, Kiyoshi Kumahata, ”Case Studies of Performance Optimization of Applications”, The Art of High-Performance Computing for Computational Science Vol.2, ed. Masaaki Geshi (Springer, Singapore, 2019), Chap. 1-3. https://doi.org/10.1007/978-981-13-9802-5_3, (2019).