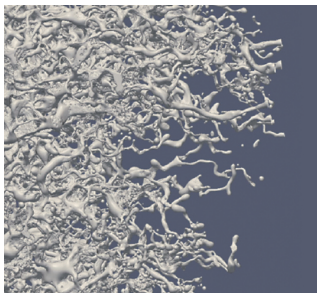
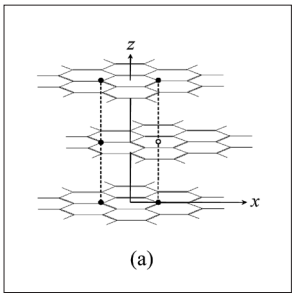
e-Science Journal
Global Scientific Information and Computing Center

Vol. 14

TSUBAME ESJ.





Two-phase Porous Flow Simulation for Natural Sandstone on GPU Supercomputer

Distributed Computing for Machine Learning on Large-Scale Image Dataset

Electronic structure calculation of large nano carbon molecules using multi-GPU massively parallel cluster system

http://www.gsic.titech.ac.jp/TSUBAME_ESJ

Two-phase Porous Flow Simulation for Natural Sandstone on GPU Supercomputer

Takeshi Tsuji Fei Jiang
International Institute for Carbon-Neutral Energy Research (I^2CNER), Kyushu University

We investigated multiphase flow characteristics inside 3D porous rocks by a highly efficient multi-phase lattice Boltzmann method (LBM). Using GPU supercomputer (TSUBAME 2.5), we can conduct two-phase LBM simulation for extremely large-scale digital rock model (1000^3 grids) reconstructed from micro-CT scanned images. These efforts enable us to enlarge the size of digital rock model from the pore-scale (μm scale) to the core-scale (mm scale). As a result, we can compare the results derived from LBM simulation with laboratory experiments. In Carbon Capture and Storage (CCS) project, the CO_2-water (two-phase fluid) migration processes in geologic reservoir are influenced by many reservoir parameters (e.g., temperature, interfacial tension, pore structure, wettability and pressure). Using the two-phase LBM, we calculated CO_2–water behavior under various reservoir conditions and identified suitable conditions for effective CO_2 storage. We further modeled mineral precipitation process of CO_2 inside the rock pore by considering CO_2-water flow.

Introduction 1

The characteristics of fluid flow in porous media have been subject of great interest in a wide range of scientific and engineering disciplines. In particular, the displacement of one fluid by another (i.e., two-phase flow) is a process that is central to enhanced oil recovery (EOR), carbon capture and storage (CCS), remediation of non-aqueous phase liquids (NAPL), and geothermal system. Here we focus on CCS project as an application of multiphase flow simulations.

CCS project involves capturing CO_2 produced by large industrial plants and injecting it deep into a geological formation. This project prevents large amounts of CO_2 emission into the atmosphere [1]. In this project, the behavior of the CO_2 injected into a reservoir can be characterized as two-phase flow in a porous media—specifically, the displacement of formation water (wetting phase) by supercritical CO_2 (nonwetting phase) during CO_2 injection (i.e., drainage process), and CO_2 displacement by water (i.e., imbibition process) after CO_2 injection. Pore-scale interfacial instabilities, such as Haines jumps, snap-off and fingering phenomena, relate to the stability, injectivity, mobility and saturation of CO_2 in the reservoir. Therefore, understanding pore-scale CO_2 flow is crucial to estimating critical CO_2 characteristics in reservoir, including storage capacity, leakage risk, and storage efficiency.

Recently, numerical models based on the lattice Boltzmann method (LBM) have been proposed and are growing in popularity because they provide a convenient means to simulate both single-fluid and multiphase flow behavior within a porous medium system [2]. Unlike traditional computational fluid dynamics (CFD) methods, which solve the conservation equations of macroscopic properties numerically, the basic idea of LBM is that it considers a many-fictive-particle system obeying the same conservation laws. Those particles perform consecutive propagation and collision processes over a discrete lattice mesh.

The conventional multiphase models, such as the level set [3], volume of fluid [4] and phase-field [5] methods, usually need complicated auxiliary algorithms to track and handle the fluid interfaces. Whereas phase separations can be generated automatically from particle dynamics, and no special treatment is needed to manipulate the interfaces in LBM. Moreover, LBM has advantages in dealing with complex boundaries and incorporating microscopic interactions for multi-physics. These features make the LB model well suited to simulate multiphase flow directly on digital rock models with complex pore structure (Fig.1).

However, multiphase LBM simulation for porous medium systems with sufficient resolution and large grid-number remains computationally challenging [6,7]. Since LB models are computationally expensive, without efficient parallel implementations of the algorithms, large amounts of computer time will be required due to the number and complexity of computations required.

Recently, a powerful parallel co-processor, the graphics processor unit (GPU), has drawn much attention due to its outstanding performance for accelerating general scientific computation. GPU computing can be performed using parallel computing architecture CUDA. Over the past few years implementation of single-phase LBM on GPUs has provided excellent performance when compared with their central processing unit (CPU) equivalent [8,9,10].

In this study, we extend and implement multi-phase LBM by adopting GPU computing techniques in order to investigate multiphase flow characteristics inside 3D porous rocks (Fig. 1). We used the micro-CT to scan the rock sample with enough resolution to reveal the correct pore structures in 2D slices. These 2D images are segmented and interpolated to reconstruct the 3D digital rock model (Fig. 1). This binarized porous media data is mapped to a uniform Cartesian grid.

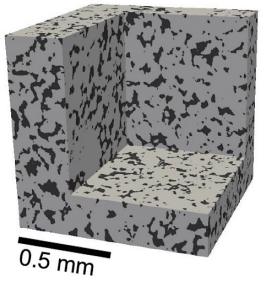


Fig. 1 Pore geometry of the sandstone extracted from micro-X CT images. Black parts indicate pore space. In CO_2 storage, we inject CO_2 into the pore space.

GPU implemented Lattice Boltzmann method 2

Multiphase lattice Boltzmann models have been proposed as an efficient method to solve immiscible two-phase flow systems in porous media. In the present study, we adopt an optimized version of the RK-model [11] which uses a multiple-relaxation-time lattice Boltzmann approach and is able to allow simulations with variations of density and viscosity. The bounce-back scheme is implemented for the non-slip boundary condition at the fluid–solid interfaces. Different wetting conditions such as wetting with a non-zero contact angle or mixed wetting can be easily incorporated by changing the order parameter on the solid wall.

The multi-phase LBM used here has three remarkable advantages. First, almost all the calculations are local; second, complicated procedures for interface tracking are not necessary; and third, the scheme is purely explicit without solving any Poisson equations. These characteristics are very favorable for parallel computing.

The parallel implementations are carried out at two levels: the CUDA core level and the inter-GPU level. Because the microprocessors of the GPU chips are thread-block organized at the logical level, parallel computing inside a single GPU board (CUDA core level) is to essentially define a relationship between thread/block IDs (threadIdx.x/y, blockIdx.x/y) and lattice nodes. Here, we define a grid of blocks in the x- and y-directions, and assign each block a number of threads in each direction corresponding to the number of lattice nodes (Fig. 2). Consequently, one thread is responsible for calculating a column of lattice nodes along the z-direction. The block size (blockDimx, blockDimy) should be determined at the beginning of the calculation, and the numbers of blocks in the x- and y-directions are therefore Nx/blockDimx, Ny/blockDimy, respectively. Consequently, there is no need to loop in the x- and y-direction in the kernel code and all the threads are automatically ordered by the GPU controller and executed simultaneously. We obtain the following relations between thread/block IDs and nodes index jx, jy in the x- and y-direction (Fig. 2):

$$jx = blockDim.x \times blockIdx.x + threadIdx.x\,; \qquad (1)$$

$$jy = blockDim.y \times blockIdx.y + threadIdx.y\,. \qquad (2)$$

Another important point to consider is the data layout of the array to store the particle distribution function. Because the threads do not access the memory individually, but in groups of 32, the GPU will coalesce access to the contiguous memory elements. As a result, continuous memory access is extremely important to exploit the maximum memory bandwidth. Here, to store the particle distribution, a one-dimensional data array is adopted in which the data layout is defined as

$$i \times L_z \times L_y \times L_x + jz \times L_y \times L_x + jy \times L_x + jx \qquad (3)$$

To reduce the data memory access, the collision step and the stream step are combined. As a consequence, the computational efficiency is increased by 30% .

To simulate a large-scale system, two memory-saving schemes have been incorporated into our code due to the limited memory space in a single GPU board. First, we adopted a sparse memory storage method to avoid storing the information of solid nodes. Second, both collision and propagation processes are designed to evolve in a single array with the size of a grid cell at one time step.

Moreover, for extremely large-scale simulation, multi-GPU implementation (inter-GPU level) is necessary. A simple domain decomposition along the z-direction is performed, and each GPU is assigned a subdomain of the total computational volume. For each interface between these subdomains, we use one additional layer of ghost cells (Fig. 3) which are synchronized at every time step to ensure the correct calculation of the color gradient.

Data communications have to be carried out between neighboring GPU boards at each time step. There are two patterns of data transfer mode (Fig. 3). One is the data transfer between GPUs on different motherboards, and the other is between GPUs connected to the same PCI-E bus on one motherboard. For the first case, the data transfer has to be conducted through host CPU sides, whereas the second can be carried out by using direct peer-to-peer communication features

(Fig. 3). Ghost cell data that need to be exchanged between hosts are transferred by an Message Passing Interface (MPI) library. The data exchange between device and host is the most limiting factor. To improve the efficiency, we used pinned-host memory and only exchanged the values that were absolutely necessary corresponding to five distribution functions for the velocity field and two densities for the phase field.

With this highly efficient multi-phase LBM code, large-scale computation with sufficiently high resolution for digital models of natural rocks becomes possible. Using 20 nodes of TSUBAME 2.5 (equipped with total 60 NVIDIA K20X GPUs), we successfully carried out the two-phase flow behaviors in a large-scale digital rock model with a system size of 1000^3 (Fig. 4). These efforts enable us to enlarge the size of digital rock from the pore-scale (μm scale) to the core-scale (mm scale). As a result, we can compare the results derived from LBM simulation with laboratory experiments.

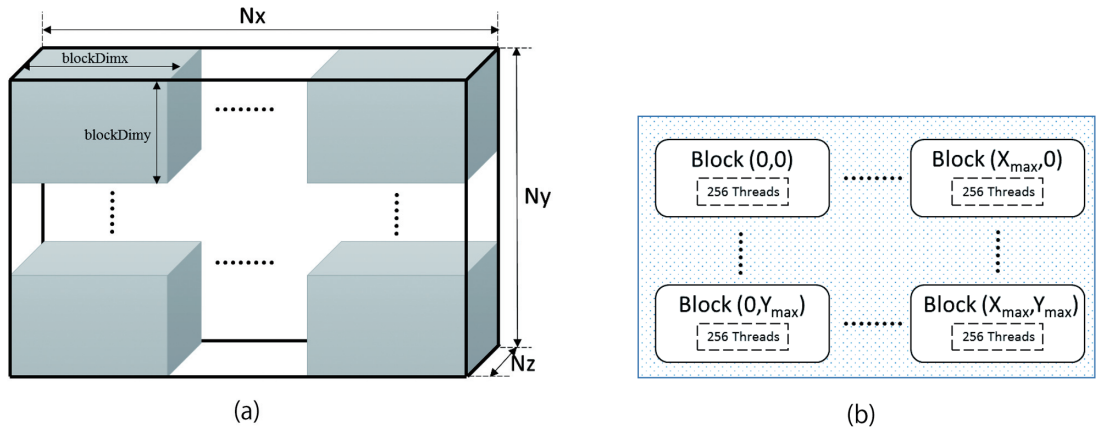


Fig. 2 Domain distribution in the CUDA framework.
(a) Physical node assignment.
(b) Logical block/thread partition.
Each thread is assigned to a single (x,y) value and works on all z values.

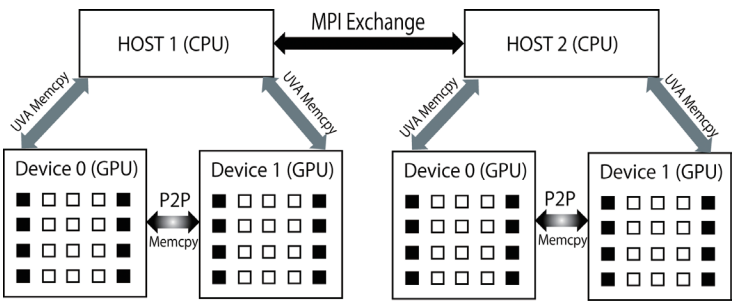


Fig. 3 Communication scheme showing data transfer between different GPU boards: Unified virtual addressing (UVA) CUDA memory copy between host and device; Peer-to-peer memory copy between devices in the same node; MPI data exchange between hosts. Ghost cells are represented by black filled squares.

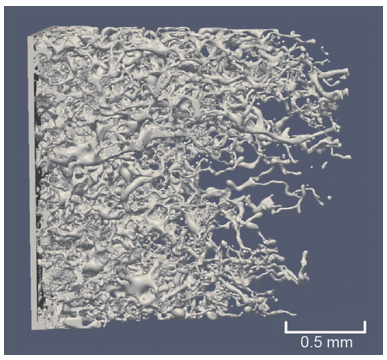


Fig. 4 Example of CO_2 behavior within pore space of Berea sandstone (Fig. 1) using two-phase lattice Boltzmann method. The left is inflow side. White indicates the injected CO_2. The solid grain and water are transparent in this figure. The grid size is 1000^3.

Applications 3

Here we showed a few applications of multiphase LBM scheme; (1) multiphase flow behavior under various reservoir conditions, (2) optimum reservoir conditions for effective residual CO_2 trapping and solubility CO_2 trapping, and (3) mineralization modeling and its influence on the hydrologic properties.

3. 1 Multi-phase flow behavior controlled by reservoir conditions

The CO_2-water behaviors in reservoir are influenced by many reservoir parameters (e.g., temperature, interfacial tension, pore structure, wettability and pressure). Here we quantify hydrological properties under several reservoir conditions (e.g., interfacial tension) from CO_2 behavior within pore space (Fig. 5) [12,13].

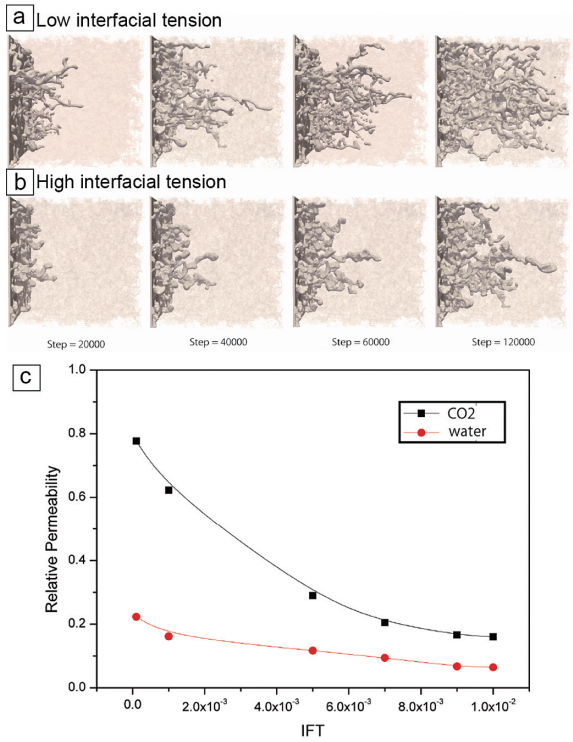


Fig.5 Supercritical CO_2 infiltration into rock as a function of IFT [12]. CO_2 behavior displayed in panel (a) was calculated at lower IFT, compared to panel (b). (c) Relative permeability as a function of IFT at 50% CO_2 saturation.

In the conditions of lower interfacial tension (Fig. 5a), the CO_2 moves faster. Indeed, the relative permeability calculated from the fluid behavior decreases with increasing IFT because of growing capillary trapping intensity (Fig. 5c). The relative permeability estimated for several reservoir conditions is crucial information in conventional field-scale reservoir fluid simulation. In contrast, equilibrium CO_2 saturation is higher at higher interfacial tension case (Fig. 5b), indicating the CO_2 storage capacity is higher in this condition. This simulation demonstrates that we can estimate CO_2 saturation and permeability at any reservoir conditions using this approach.

3. 2 Optimum conditions for residual and solubility CO2 trapping

We calculate residually trapped CO_2 clusters to quantify CO_2 trapping mechanisms in natural sandstone (Fig.1). The residual CO_2 distribution is generated following simulation of the drainage and imbibition processes (Fig.6) [14]. The characteristics of the residual CO_2 cluster in terms of size distribution, major length, interfacial area, and sphericity are investigated under conditions of different interfacial tension (IFT).

Our results indicate that high interfacial tension increases the residual CO_2 saturation and leads to a large size distribution of residual CO_2 clusters (Fig. 6c). In contrast, low interfacial tension results in a larger interfacial area, which would be beneficial for dissolution and reaction processes during geological CO_2 storage (Fig. 6a). By using this method, we can obtain residual CO_2 distributions under different conditions for optimizing the CO_2 storage capacity.

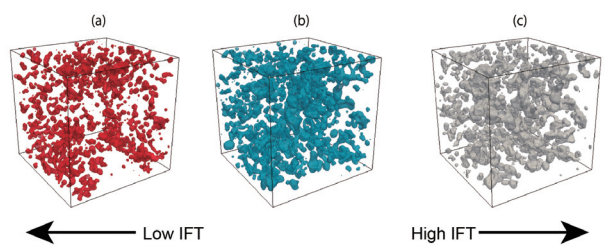


Fig.6 Residual CO_2 distributions [14]. IFT increases from left to right.

3. 3 CO_2 mineralization modeling

In CCS project, the injected CO_2 would be precipitated in rock pore. Here we calculate the mineralization processes of injected CO_2 and its influence upon the hydrological properties [15]. The calcite deposition within the pore space is calculated by using an advection–reaction formulation solved by finite volume method (FVM); we model the precipitated rock by transferring the fluid node to solid node according to the calcium concentration level. The clogging model updates the solid phase according to precipitation process and consequently changes the geometry and porosity of the sample rock. Then fluid solver is called to recalculate the flow field based on the evolved pore microstructures. To validate our method, the carbonate precipitation simulation is carried out on a beads pack model and compared with laboratory experiment. Since the simulation results are in consistent well with the laboratory data, our method can simulate realistic mineralization process.

Our calculation shows that location of mineral deposition depends on not only the fluid velocity field but also the rock structure (Fig.7). The calculated permeability variation due to the carbonate precipitation demonstrates that evolution of pore structure significantly influences the absolute permeability, while it only affects the relative permeability of non-wetting phase. Because the movement of non-wetting fluid is suppressed due to the increased capillary pressure associated with mineralization, permeability of non-wetting phase is significantly influenced by mineralization (pore size reduction) [16]. These observations can be used in the long-term reservoir simulation including geochemical reactions.

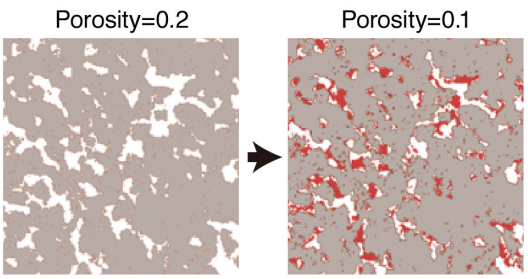


Fig.7 Mineral precipitation calculated from the CO_2 flow within the rock [15]. Red and white parts represent precipitated mineral (carbonate) and pore space, respectively. Left is original rock model.

Summary 4

We have successfully developed two-phase LBM scheme for large-scale digital rock models. Multi-GPU implementation is crucial for the extremely large-scale simulation. Using this approach, we (1) characterize the influence of reservoir conditions upon multiphase flow behavior (permeability) and saturation, (2) reveal optimum conditions for residual and solubility CO_2 trapping in CCS project, and (3) calculate CO_2 mineralization (or carbonate precipitation) by considering CO_2 behavior, and evaluate its influence upon the permeability. Recently, we have calculated elastic properties of CO_2 saturated and mineralized rocks (i.e., digital rock) using dynamic wave propagation simulations [17].

Using these numerical simulations for the large-scale digital rock models, we can accurately model CO_2 behaviors and reactions occurred in rock pore. In near future, large-scale geological formation at CO_2 storage sites could be digitalized and accurately controlled (or managed) by these modeling studies.

Acknowledgements

This study was supported by JSPS through a Grant-in-Aid for Scientific Research on Innovative Areas (no. 15H01143), Grant-in-Aid for Scientific Research (A) (no. 24246148), Grant-in-Aid for Research Activity Startup (no. 26887028) and Bilateral Joint Research Projects with MOSR, and JICA/JST through SATREPS project. We gratefully acknowledge support of I2CNER, sponsored by the World Premier International Research Center Initiative (WPI), Ministry of Education, Culture, Sports, Science, and Technology (MEXT), Japan.

References

[1] Metz, B., et al. eds. (2005), IPCC Special Report, Carbon Dioxide Capture and Storage. Cambridge University Press, Cambridge.

[2] Boek, E.S., M. Venturoli (2010), Lattice -Boltzmann studies of fluid flow in porous media with realistic rock geometries. Comput. Math. Appl. 59, 2305–2314.

[3] Sussman, M., A.S. Almgren, J.B. Bell, P. Colella, L.H. Howell, M.L. Welcome (1999), An adaptive level set approach for incompressible two-phase flows. J. Comput. Phys. 148, 81–124.

[4] Hirt, C.W., B.D. Nichols (1981), Volume of fluid (VOF) method for the dynamics of free boundaries. J. Comput.

Phys. 39(1), 201–225.

[5] Badalassi, V.E., H.D. Ceniceros, S. Banerjee (2003), Computation of multiphase systems with phase field models. J. Comput. Phys. 190(2), 371–397.

[6] Ahrenholz. B, J. Tölke, P. Lehmann, A. Peters, A. Kaestner, M. Krafczyk, W. Durner (2008), Prediction of capillary hysteresis in a porous material using lattice-Boltzmann methods and comparison to experimental data and a morphological pore network model, Advances in Water Resources, 31 (9), 1151–1173.

[7] Ramstad. T, N. Idowu, C. Nardi, P.E. Øren (2012), Relative permeability calculations from two-phase flow simulations directly on digital images of porous rocks, Transport in Porous Media, 94 (2), 487-504.

[8] Tölke, J., M. Krafczyk (2008), TeraFLOP computing on a desktop PC with GPUs for 3D CFD. Int. J. Comput. Fluid Dyn. 22(7), 443–456.

[9] Tölke, J. (2010), Implementation of a lattice Boltzmann kernel using the compute unified device architecture developed by nVIDIA. Comput. Vis. Sci. 13(1), 29–39.

[10] Kuznik, F., C. Obrecht, G. Rusaouen, J.J. Roux (2010), LBM based flow simulation using GPU computing processor. Comput. Math. Appl. 59(7), 2380–2392.

[11] Tölke, J., S. Freudiger, M. Krafczyk (2006), An adaptive scheme using hierarchical grids for lattice Boltzmann multi-phase flow simulations. Comput. Fluid 35, 820–830.

[12] Jiang, F., T. Tsuji, and C. Hu (2014), Elucidating the role of interfacial tension for hydrological properties of two-phase flow in natural sandstone by an improved lattice Boltzmann method, Transport in Porous Media, 104, 1, 205-229.

[13] Yamabe, H., T. Tsuji, Y. Liang, and T. Matsuoka (2015), Lattice Boltzmann simulations of supercritical CO_2-water drainage displacement in porous media: CO_2 saturation and displacement mechanism, Environmental Science & Technology, 49 (1), 537-543.

[14] Jiang, F., and T. Tsuji (2015), Impact of interfacial tension on residual CO_2 clusters in porous sandstone, Water Resources Research, 51, 1710-1722.

[15] Jiang, F., and T. Tsuji (2014), Changes in pore geometry and relative permeability caused by carbonate precipitation in porous media, Physical Review E 90, 053306.

[16] Kitamura, K., F. Jiang, A.J. Valocchi, S. Chiyonobu, T. Tsuji, and K.T. Christensen (2014), The study of heterogeneous two-phase flow around small-scale heterogeneity in porous sandstone by measured elastic wave velocities and lattice Boltzmann method simulation, J. Geophys. Res. 119, 7564-7577.

[17] Yamabe, H., T. Tsuji, Y. Liang, T. Matsuoka (2016), Influence of fluid displacement patterns on seismic velocity during supercritical CO_2 injection: Simulation study for evaluation of the relationship between seismic velocity and CO_2 saturation, International Journal of Greenhouse Gas Control, 46, 197-204.

Distributed Computing for Machine Learning on Large - Scale Image Dataset

Ikuro Sato* Ryutaro Watanabe* Hiroki Nishimura** Akihiro Nomura*** Satoshi Matsuoka***
* DENSO IT LABORATORY, INC. ** DENSO CORPORATION *** Tokyo Institute of Technology

Intensive researches have been revealing that machine-learning methods known as Deep Neural Networks (DNNs) show great classification capabilities through supervised training on massive datasets. This research aims to quantify a condition that primarily controls classification capabilities, and generate a high-performing ensemble classifier consisting of plural DNN models. As for the training, we used node-distributed machine-learning program that we developed from scratch. As many as 96 GPUs are used to train a single DNN model. Most models are trained during the TSUBAME Grand Challenge, using 1146 GPUs simultaneously at peak, reaching about 1 TFLOPS (single) per GPU in the cost derivative parts.

Introduction 1

Image classification is one of the main research fields in Information Technology. Many industrial applications have come out over the years, such as person detection with surveillance cameras, appearance inspection in production equipment, and object detection for automatic emergency braking systems equipped with high-end automobiles, to name a few. Especially for applications that ensure human safety classification accuracy is crucial; therefore, technological progresses or breakthroughs are still in high demand.

Performance of a machine-learning based classifier in general depends on the algorithm, the size of the training dataset, and the computational resources. Krizhevsky et al. showed that far better classification performance is obtained by DNNs, which learn the image features from data, rather than conventional approaches that use predefined image features and only train feature classifier [1]. It is important to know the fact that a large-scale image dataset named as ImageNet was made available behind this success [2]. In fact, there are cases where DNNs perform rather poorly when the size of the dataset is not large [3]. Since optimization of DNN requires a large amount of numerical operations, therefore use of GPUs is a standard practice today. Still, it is not rare to spend a month or even longer for network training on a single GPU.

We have developed a node-distributed, machine-learning program for fast DNN training as a part of project of driving safety application development. We have confirmed that our program runs a few tens of times faster than a single GPU on TSUBAME 2.5.

We applied the above-mentioned program to a large-scale image dataset, generate a high-performing image classifier made of an ensemble of several DNN models selected from a larger set of trained DNN models in a relatively short period of time. We also quantify the relation between the number of free parameters and classification performance from the set of trained DNN. The latter experiment is preliminary at the moment, but it still shows an intriguing trend behind the classification accuracy and the DOF.

Software 2

There are in general two types of approaches in distributed machine learning programs: model parallel and data parallel. Model parallel means to split model parameters and hidden layer representation into two or more GPUs. (Here, we assume the use of GPUs for training phase.) The forward and backward steps require GPU-GPU communications in a synchronous way to compute the cost derivatives [1, 4, 5]. Model parallel is only meaningful when the memory required for all variables necessary to compute cost derivatives exceeds the GPU memory. In other words, it is a technique for training relatively large networks. The other approach, data parallel, is to split data samples into two or more GPUs. In this approach different GPUs work on different images. Only data parallel is implemented in our program. One can implement only model parallel [1] or both [4, 5].

Data parallel approach can further be categorized into synchronous types [6] and asynchronous types [4]. In the former types, once all worker GPUs reads in different images and compute corresponding cost derivatives, the derivatives get added up through communications to update weight parameters and updated parameters are distributed to all workers. This cycle is repeated until convergence. The merits of this method are two-fold: many more images are processed for a given time compared to single machine training, and

convergence is guaranteed. On the other hand, asynchronous data parallel is more advantageous in training speed because it prevent hardware resources from idling, though convergence is not guaranteed. Reference [4] reports asynchronous training cause no problem in convergence in practice.

Throughout this research we used an in-house, machine-learning program, categorized into asynchronous data-parallel approach without special nodes (known as parameter servers). We will discuss the detail of the algorithm in other occasion.

Dataset 3

We used ILSVRC 2012 dataset [2], which is a subset of ImageNet. The ILSVRC 2012 dataset consists of 128M training samples, 50K validation samples and 100K test images. This dataset is one of the largest datasets among public image dataset. Each image has a class label out of 1000 class. (Number of images per class is not unique.) The word "sample" means a pair of image and the true class label. No true classes for the test set are given. Classification accuracy on the test set can be only obtained by submitting the result to the test server.

DNN models 4

We describe our network designing strategy here. The fraction of the number of network parameters over the number of training samples is chosen from (1, 30) to study the relationship between numbers of parameters and the classification scores. Total number of convolutional layers in a network is chosen from 11 up to 48. The filter size is fixed to be 3 × 3, as is seen in [8]. The numbers of feature maps in hidden layers are set by hand. A fully-connected layer is placed at the very last layer before taking softmax normalization in each model. Input image size is 200 × 200 pixel2 or larger for good visibility. This resolution can be realized by inserting pooling layers appropriately. The number of pooling layers in a model ranges from 2 to 5. One of the networks trained in this work is shown in Table 1.

layer	1	2	3	4	5	6	7	8	9
#maps	3	32	38	54	64	90	106	150	178
map size	396^2	394^2	196^2	194^2	96^2	94^2	46^2	44^2	42^2
operation	C	CP	C	CP	C	CP	C	C	CP

layer	10	11	12	13	14	15	-
#maps	212	298	354	421	593	704	1000
map size	20^2	18^2	16^2	7^2	5^2	3^2	1^2
operation	C	C	CP	C	C	C	S

Table 1. An example of neural networks that we used in this work. It has 16M parameters and consists of 15 convolutional, 5 pooling, and 1 fully-connected layers. "C" means convolution, "CP" means convolution and pooling, and "S" means softmax normalization. Activation function symbols are omitted.

A few more sentences on fully-connected layers are in order. It is a regular practice to place a couple of fully-connected layers after map resolution is sufficiently reduced (for example, see [1, 8]). In these layers particular weight parameter is linked to particular pair of neurons, indicating that many parameters are required. As a result, overfitting tends to take place under some conditions. It is a common practice to use a regularization technique such as dropout [10] in fully-connected layers to prevent such behavior. In our model construction, a network has one and only one fully-connected layer and the last feature map has relatively small spatial dimension 3 × 3. We expected that this construction poses structural regularization to some extent, preventing severe overfitting, while reducing the number of parameters. In our preliminary experiment, we confirmed the network similar to that of Table 1 has better generalization ability, while it has fewer parameters, than a network with more than one fully-connected layer.

We summarize all the details about the way of optimization in Table 2. As many (few) as 96 GPUs (24 GPUs) are used to train a single model. Most of the DNN models are trained during the TSUBAME Grand Challenge. Training took roughly 7 days. During the Grand Challenge, we used 1146 GPUs at peak to train 13 models simultaneously. The computational efficiency that we measured on the derivative computation part was 0.98 TFLOPS (single) per GPU, whereas the theoretical computational efficiency of the GPU (Tesla K20X) is 3.95 TFLOPS (single).

Activation function	Rectifier Linear Unit [11]
Cost function	Cross Entropy loss with softmax normalization
Dropout (any variants)	Not used.
Minimization algorithm	Mini-batch Stochastic Gradient Descent
Size of mini-batch	The median value in the ensemble: 132
Training duration	The median value in the ensemble: 515 epoch
Data augmentation	Random homography transformation, cropping, side flipping, radial distortion, weak elastic distortion distortion, color noise
Decision rule	APAC [12]

Table 2. Optimization detail.

Qualitative evaluation 5

Figure 1 shows some of validation images classified successfully. The ground truth of Fig. 1 (a) is "unicycle". What makes the prediction difficult is that the wheel part is out of the frame, and only a part of the shaft is visible. Nevertheless, the model can predict the class with high confidence. Interpretation is that crowd of people looking above and the pose of the performer in the center tell much about the target object. From this sample, DNNs likely capture scene context including the background. Similar interpretation holds in Fig. 1 (b), where the ground truth is "paddle". Figure 1 (c) shows image of "bubble", and again the classifier correctly predicts the class. It used to be a hard problem to recognize semi-transparent object because the background image features admixes through the transparent part. The fact that our model successfully recognizes this sample with high confidence indicates that DNNs easily overcome this type of difficulty.

Figure 2 shows some of validation images classified wrongly. The ground truth of Fig. 2 (a) is "Arabian camel", but the model prediction is "dog sled". The interpretation is that the DNNs over-evaluate the context and belittle the shape of the head. Figure 2 (b) shows image of "triceratops", but our model fails to answer this. Since our model fails to recognize frontal shape of triceratops, the generalization ability is somehow limited.

Fig. 1 Validation images correctly classified by our model.

(a) unicycle

(b) paddle

(c) bubble

Fig.2 Validation images falsely classified by our model.

(a) Arabian camel

(b) triceratops

Quantitative evaluation 6

Figure 3 shows the relationship between the classification score and the number of training parameters. It indicates that classification performance is more-or-less equally good when the number of parameters is in the range of (8×10^6, 40×10^6). It also indicates that classification performance drops drastically when the number of parameters goes below 8×10^6. It is interesting that this kind of trend is seen even though the network architectures vary. This fact implies that the number of parameters is the major factor for classification performance. When distributed machine learning platform is taken into consideration, use of models with less parameters is desirable in terms of training speed. As long as this study is concerned, it is reasonable to use of models with around 8×10^6 parameters.

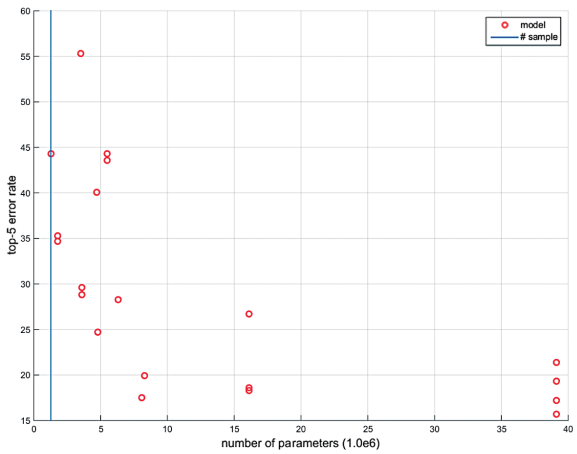


Fig. 3 The relationship between the classification accuracy and the number of training parameters. The blue line indicates the number of training samples.

We have tested every possible combinations of trained DNNs to generate an ensemble classifier. It ended up with six models summarized in Table 3. Here, ensemble classification simply means to take an expectation value of the logarithm of softmax outputs. We confirmed that about 2% reduction in top-5 error rate on the validation set. It means that every model extracts slightly different image features.

Model			Top-5 error rate (%) on validation set	
#convolutional layers	#pooling layers	#parameters	per model	Ensemble
11	4	16M	18.62	13.67
11	4	39M	17.20	
11	4	39M	15.68	
15	5	16M	18.26	
16	4	8M	17.51	
16	5	16M	26.65	

Table 3. Constituents of the ensemble classifier.

We compare the result with representative ones. Classification scores on the test set are summarized in Table 4. The state-of-the-art score is 3.57 % [9], which is better than human classification score 5.1 % [2]. Interestingly their networks have connections that skip some layers, and they claim that these architectures allow fast training even though the network is deeper than 100 layers. GoogLeNet [7] has so-called "Inception Modules" and authors claim that such networks yield high classification performance with considerably reducing the number of parameters. OxfordNet [8] proposes relatively simple architecture with repeated 3 × 3 convolutional layers and dropout-regularized, fully-connected layers. AlexNet [1] is famous for its breakthrough in 2012. GoogLeNet has the lowest number of parameters among these models, and ours is the second. AlexNet has the fewest convolutional layers, and ours is the second. The score of our model is better than AlexNet, lying in the reasonable range.

Model	#constituent models	#layers with product	#parameters	Top-5 error rate (%) on test set
AlexNet [1]	5	8	60M	16.64
GoogLeNet [7]	7	22	5M	6.66
OxfordNet [8]	7	19	144M	7.33
MSRA [9]	6	152	57M*	3.57
ours	6	16	39M	13.89

Table 4. Evaluation of our ensemble model with representative ones. Column 3 and 4 show the largest numbers in the constituent models. (*We estimate the #parameters from their network architecture.)

Outlook 7

Many researches have been revealing the techniques for effective optimization of deep (sometimes extremely deep) neural networks. We are interested in finding ways to exploit representation ability of neural networks while keeping the amount of parameters as low as possible. Reduction of the number of parameters is very important for training speedup. Today's DNNs requires a very large number of parameters compared to the number of training samples. This fact makes us believe there is some room to make the parameter space compact.

Acknowledgements

The node-distributed machine-learning program used in this work was developed during one-year TSUBAME Trial Use Program. Most of the DNN models used in this work were trained during the TSUBAME Grand Challenge 2015. We express our gratitude for such generous opportunities.

References

[1] A. Krizhevsky, et al., "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.
[2] O. Russakovsky, et al., "ImageNet Large Scale Visual Recognition Challenge", IJCV 2015.
[3] M. Oquab, et al., "Learning and transferring mid-level image represention using convolutional neural networks", CVPR 2014.
[4] J. Dean, et al., "Large Scale Distributed Deep Networks", NIPS 2012.
[5] R. Wu, et al., "Deep Image: Scaling up Image Recognition", arxiv:1501.02876, 2015.
[6] F. N. Iandola, et al., "FireCaffe: Near-Linear Acceleration of Deep Neural Network Training on Compute Clusters", arxiv:1511.00175, 2015.
[7] C. Szegedy, et al., "Going Deeper with Convolutions", CVPR 2015.
[8] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", ICLR 2015.
[9] K. He, et al., "Deep Residual Learning for Image Recognition", arxiv:1512.03385, 2015.
[10] G. E. Hinton, et al., "Improving neural networks by preventing co-adaptation of feature detectors", arxiv:1207.0580, 2012.
[11] V. Nair and G. E. Hinton, "Rectified Linear Units Improve

Restricted Boltzmann Machines", ICML 2010.
[12] I. Sato, et al., "APAC: Augmented PAttern Classification with Neural Networks", arxiv:1505.03229, 2015.

Electronic structure calculation of large nano carbon molecules using multi-GPU massively parallel cluster system

Michio Katouda* Akira Naruse** Takahito Nakajima*

* RIKEN Advanced Institute for Computational Science ** NVIDIA Japan

In this study, we performed electronic structure calculations of large nano carbon molecular assemblers on TSUBAME 2.5 using newly developed multi general purpose computing on graphics processing units (GPGPU) program of the resolution-of-identity second order Møller-Plesset perturbation (RI-MP2) energy calculations. We report the results of performance evaluation on TSUBAME 2.5. The peak performance of the multi-GPU computations of largest nano carbon molecules (nanographene dimer $(C_{96}H_{24})_2$) using 1,349 nodes and 4,047 GPUs of TSUBAME 2.5 are 514.7 TFLOPs. We also report the inter-molecular interaction analysis of two-layer π - π stacked nanographene dimers.

Introduction 1

Nano carbon molecules have been studied actively with experimental, theoretical, and computational approach for the application of organic molecular function materials. Elucidation of local structure and stability of nano carbon molecular assemblers is essential to develop and evaluate those materials. Electronic structure calculation is recognized as the powerful tool for the investigation of those properties. Second order Møller-Plesset perturbation (MP2) theory[1] is the robust method for the accurate treatment of electronic correlations that is important for the description of weak inter-molecular interactions such as the van der Waals interaction, and applicable to the application of calculations of large nano carbon molecular assemblers. However, the computation costs of MP2 calculations are much higher than the Hartree-Fock (HF) calculations, and the sizes of molecules applicable to the MP2 calculations are limited. Thus, the use of large computation systems such as K computer and TSUBAME 2.5 is essential for the allocation of MP2 calculations of large carbon molecular assemblers.

Our research team has been performed the NTChem[2,3] software project for the massively parallel computing of electronic structures of large-scale molecules utilizing the petascale supercomputers such as the K computer. Various efficient theories and algorithms have been implemented into NTChem (Fig. 1) such as (1) electronic structure theories of the ground and excited states of molecules based on HF theory and density functional theory (DFT) and (2) accurate electron correlation theories for ground and excited states such as MP2 theory, coupled-cluster theory, and quantum Monte Carlo method.

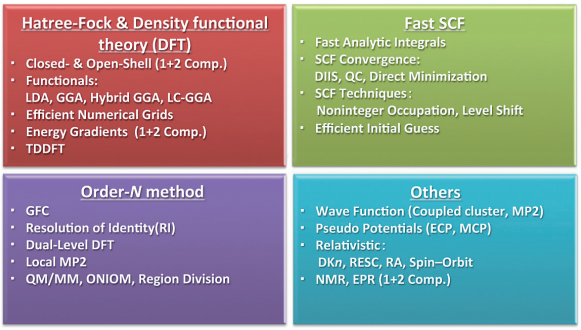


Fig.1 Main feature of NTChem program

Message passing interface (MPI)/OpenMP hybrid parallelization of HF, DFT, and MP2 codes are performed for the massively parallel computing on the K computer. Massively parallel computations using 10-100 thousand nodes of K computer are possible for those calculations. Recently, the authors have been developed an efficient massively parallel algorithm and code for the calculations of electron correlation energy of large molecules based on the resolution-of-identity MP2 (RI-MP2) method[4,5]. The RI-MP2 calculation of large molecules containing 360 atoms and 9,840 atomic orbitals (AOs) was performed successfully using 8,911 nodes of the K computer.[6]

Furthermore, number of heterogeneous system in which the accelerators such as graphics processing units (GPUs) are installed has been increasing year by year in the field of supercomputers. Demands for electronic structure calculations using accelerators have been increasing day by day. In the previous study, we have been developed codes for the general purpose computing on GPU (GPGPU) in NTChem. Recently, we have performed the improvement of parallel algorithm and the GPGPU implementation of RI-MP2 energy calculations utilizing the multi-node and multi-GPU systems.[7]

In this article, we report the results of the multi-node and multi-GPU computation of the RI-MP2 energies of large

nano carbon molecules on TSUBAME 2.5. The overview of GPGPU code of RI-MP2 calculations used in this study is introduced in Section 2. The results of performance evaluation of our GPGPU code of RI-MP2 calculations are presented in Sections 3 and 4. We also report the inter-molecular interaction analysis of π - π stacked nanographene dimers as an example of nano carbon molecular assemblers in Section 5.

Multi GPGPU implementation of RI-MP2 energy calculations on NTChem program 2

2. 1 RI-MP2 method

The RI-MP2 method[4,5] is an efficient computation method to speed up the MP2 calculations introducing the RI approximation (or the density fitting approximation) of four-center electron repulsion integrals (ERIs) as

$$(ia|jb) = \sum_n B_n^{ia} B_n^{jb} \tag{1}$$

$$B_n^{ia} = \sum_{\mu\nu l} (\mu\nu|l)(l|n)^{-1/2} C_{\mu i} C_{\nu a} \tag{2}$$

introducing the auxiliary basis functions (indices l and n) expanding the products of AOs. This approximation reduces the computation costs and the memory requirements considerably due to the reduction of prefactors. The main computation bottleneck of RI-MP2 calculations is the matrix-matrix multiplication in eq. (1). But, the efficient computation and thread-based parallelization is possible utilizing the DGEMM routine in the optimized Basic Linear Algebra Subroutines (BLAS) [8] library for systems. The MP2 correlation energy is evaluated by

$$E^{(2)} = \sum_{ij}^{\text{occ.}} \sum_{ab}^{\text{vir.}} \frac{(ia|jb)\left[2(ia|jb) - (ib|ja)\right]}{\varepsilon_i + \varepsilon_j - \varepsilon_a - \varepsilon_b} \tag{3}$$

using the approximated integrals of eqs (1) and (2).

2. 2 Massively parallel implementation of RI-MP2 method

Before this study, we have performed the development of massively parallel algorithm and MPI/OpenMP hybrid parallel implementation of RI-MP2 energy calculations.[6,7] In this implementation, the matrix-matrix multiplication in eq. (1) is parallelized distributing the indices (a and b) of virtual molecular orbitals (MOs) to the MPI processes. This scheme improves the parallel scalability considerably than the previous schemes where the indices (i and j) of occupied MOs are distributed[5,9]. We also apply the in-core storage scheme for utilization of distributed memory by intermediate data of three-center integrals in eqs. (1) and (2) to eliminate the input-output (I/O) overhead demanding in the conventional MP2 code. Recently, we have improved the MPI parallel scalability applying the two-dimensional (2D) hierarchical MPI parallelization utilizing the local MPI communicator. This improvement attains the massively parallelization utilizing more than 10 thousand processes. In this scheme, we parallelized the matrix-matrix multiplication in eqs. (1) and (2) with the block data distribution of matrices as the second dimension of MPI parallelization as well as the first dimension of MPI parallelization of virtual MO indices (a and b). Using the improved code, the RI-MP2 calculation of large molecules containing 360 atoms and 9,840 AOs was performed successfully up to 80,199 nodes of the K computer.[7]

2. 3 Multi GPGPU implementation of RI-MP2 method

Before this study, we have performed multi GPGPU implementation[7] into RI-MP2 parallel code mentioned in Section 2.2 using the compute unified device architecture (CUDA)[10]. We apply the offloading model in the multi-GPU computing where the matrix-matrix multiplication of eqs. (1) and (2) is offloaded to the multi-GPUs (Fig. 2).

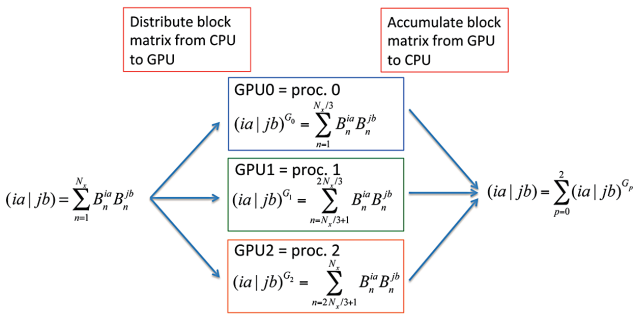


Fig.2 Scheme of GPGPU implementation of eq. (1) in RI-MP2 energy calculation

The task distribution to the multi-GPUs are attained assigning each MPI process for the partial matrix-matrix multiplication of the blocked matrices mentioned in Section 2.2. The matrix-matrix multiplication is implemented using cublasDgemm routine in cuBLAS[11] library optimized by NVIDIA. The data transfer between CPUs and GPUs becomes serious bottleneck in this GPU computing. To reduce this overhead, we have implemented the batch processing of matrix-matrix

multiplication for different indices of virtual MOs (*a* and *b*) where the data in the same batch is transfer between CPUs and GPUs at once utilizing the page-locked (pinned) memory that enables a direct memory access on GPUs to request transfers to and from the host memory without the involvement of CPUs. The CPU code and GPU code were compiled with Intel compiler and CUDA 5.0 compiler, respectively. Intel Math Kernel Library was used for BLAS and LAPACK library and cuBLAS library was used.

Performance evaluation of multi-GPU computation of RI-MP2 energy on TSUBAME 2.5: speedups using GPU and parallel performance 3

We first investigated the elapsed times and parallel performance of the multi-GPU jobs of RI-MP2 energies on TSUBAME 2.5. We performed the RI-MP2 energy calculations of nanographene $C_{96}H_{24}$ using 64-512 nodes of TSUBAME 2.5. cc-pVTZ basis set [12] and corresponding auxiliary basis set [13] are employed (120 atoms, 3,216 AOs, 300 occupied MOs, 2,916 virtual MOs, 8,496 auxiliary basis functions, and no symmetry). HF calculations before MP2 calculations were performed on the K computer, and the results of MOs and MO energies were used for input data. 3 MPI processes and 4 OpenMP threads per node and 1 MPI process and 12 OpenMP threads per node are employed for GPU and CPU jobs, respectively. The elapsed times, parallel scalability, and speedup using GPUs are summarized in Table 1.

Nodes	GPU job		CPU job		Speed-ups using GPUs
	Time [s]	Speed-ups	Time [s]	Speed-ups	
64	198	64	1277	64	6.4
128	123	104	806	101	6.6
256	92	137	531	154	5.8
512	86	148	349	234	4.1

Table 1. Elapsed time and parallel speedups of RI-MP2/cc-pVTZ calculation of nanographene $C_{96}H_{24}$ on TSUBAME 2.5

The GPU jobs are 4.1-6.6 times faster than the CPU jobs using 64-512 nodes. Although the parallel performance of GPU jobs is lower than that of CPU jobs, it is good for both cases with the number of nodes increases up to 512 nodes (148 and 234 times of speedups for the GPU and CPU jobs, respectively, using 512 nodes).

Multi-GPU computation of RI-MP2 energy of large carbon molecules 4

We then performed the multi-GPU computation of large carbon molecules using nearly entire system of TSUBAME2.5. We performed the RI-MP2 calculations of nanographene dimer $(C_{96}H_{24})_2$ using 1,349 nodes and 4,047 GPUs of TSUBAME 2.5. cc-pVTZ basis set and corresponding auxiliary basis functions are employed (240 atoms, 6,432 AOs, 600 occupied MOs, 5,832 virtual MOs, 16,992 auxiliary basis functions, and no symmetry). 3 MPI processes and 4 OpenMP threads per node and 1 MPI process and 12 OpenMP threads per node are employed for GPU and CPU jobs, respectively.

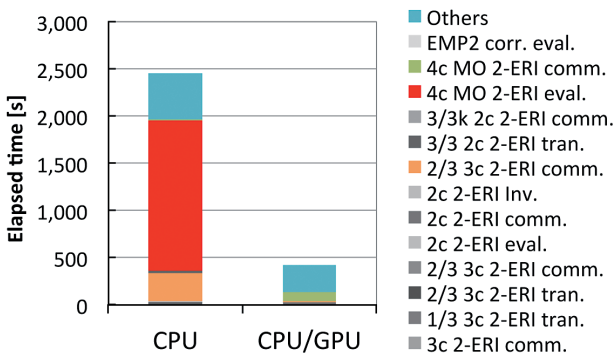


Fig.3 Detailed profile of elapsed time of RI-MP2/cc-pVTZ calculations of nanographene dimer $(C_{96}H_{24})_2$ using GPUs and CPUs on TSUBAME 2.5

Fig. 3 presents a detailed profile of the elapsed time of GPU and CPU jobs. Considerable improvement of performance is attained using GPUs. The GPU computing is 5.9 times faster than the CPU computing due to the reduction of time for matrix-matrix multiplication in DGEMM routine decreases to very short times, as shown in Fig. 3 (see the red color for this part). The elapsed times of GPU job and CPU job are, respectively, 419 s and 2,467 s. Considerable improvement of the measured peak performance value was attained using GPUs. The measured peak performance values of GPU and CPU codes are, respectively, 514.7 TFLOPs, and 87.5 TFLOPs. However, the ratio of measured

and theoretical peak performances of the GPU implementation (9.3%) is lower than that of the CPU implementation (42%). This is because the most time consuming part of the GPU jobs is no longer matrix-matrix multiplication performed on GPUs but other operations such as the intra-node collective communication of four-center MO ERIs and the inter-node point-to-point communication of three-center ERIs.

Analysis of π - π stacking interactions of nanographene dimers 5

We analyzed the π - π stacking interaction energies of two-layer nanographene dimers from the results of calculations performed in this study. The model molecules of dimers employed in this analysis are $(C_{24}H_{12})_2$, $(C_{54}H_{18})_2$, and $(C_{96}H_{24})_2$ (Fig. 4). We consider the model of AB stacking structure (Fig. 4(a)). Experimental C–C bond (1.45 Å) and inter-layer (3.35 Å) distances of bulk graphite, and C–H bond distance (1.1 Å) of benzene are used to build the models.

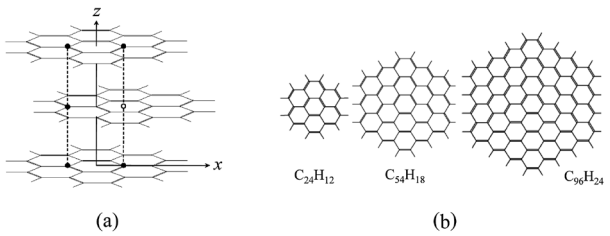


Fig.4 Model molecules of two-layer nanographene dimers.

The interaction energies are calculated using the MP2 method, the spin-component-scaled MP2 (SCS-MP2) [14] method, and the double-hybrid DFT with the empirical dispersion correction (B2PLYP-D3[15]). The cc-pVTZ basis set and corresponding auxiliary basis set are used. Basis set superposition errors were corrected using the counterpoise method[16]. The interaction energies per one carbon atom are summarized in Table 2.

These results demonstrated that the π - π stacking interaction energies converge to the value of bulk two-layer graphite sheets with the size of model increases. The results of SCS-MP2 (68.0 meV/atom) and B2PLYP-D3 (61.2 meV/atom) reproduces the experimentally estimated dispersion interaction energy of bulk graphite[17] (52 ± 5 meV/atom) well, although that of MP2 (85.7 meV/atom) yielded a much overestimated interaction energy than the other methods and the experimental value.

Model	MP2	SCS-MP2	B2PLYP-D3
$(C_{24}H_{12})_2$	68.6	44.4	42.7
$(C_{54}H_{18})_2$	87.6	59.4	54.6
$(C_{96}H_{24})_2$	85.7	68.0	61.2

Table 2. π - π stacking interaction energies of nanographene dimers per one carbon atom (meV/atom)

Concluding remarks 6

In this study, we performed the RI-MP2 energy calculations of nano carbon molecular assemblers using up to 1,349 nodes and 4,047 GPUs on TSUBAME 2.5. Speedups using GPUs and parallel scalability were investigated on TSUBAME 2.5 using up to 512 nodes. The GPU computation speeds up considerably (4.1-6.6 times) the RI-MP2 calculations. Parallel scalability of present GPU implementation is good with the number of nodes. 514.7 TFLOPs of the measured peak performance is attained for the GPU job of $(C_{96}H_{24})_2$ using 1,349 nodes and 4,047 GPUs of TSUBAME 2.5, which is much higher than that of CPU jobs (87.5 TFLOPs).

We also analyzed the π - π stacking interaction energies of two-layer nanographene dimers. The π - π stacking interaction energies converge to the value of bulk two-layer graphite sheets with the size of model increases. The SCS-MP2 and B2PLYP-D3 methods reproduced the experimentally estimated dispersion interaction energy of bulk graphite well, although the MP2 method yielded much larger dispersion energies than the other methods.

To date, we have not yet performed the GPGPU implementation of the three-center AO ERIs routine and optimized our code for network communications to match the network of TSUBAME 2.5. We are planning these developments to match the TSUBAME 2.5 architecture.

Acknowledgements

The authors would like to thank Mr. Yukihiko Hirano at NVIDIA Japan for fruitful discussions and his warm encouragement. This work was supported by by the TSUBAME grand challenge program, category A and the MEXT, Japan, Next-Generation Supercomputer project (the K computer project). Benchmark calculations were performed on TSUBAME 2.5 system (Tokyo Institute of Technology, Tokyo, Japan). Preliminary calculations were performed on the K computer (RIKEN, Kobe, Japan), the supercomputer system of the Research Center for Computational Science (National Institute of Natural Sciences, Okazaki, Japan).

References

[1] C. Møller and M. Plesset: Note on an Approximation Treatment for Many-Electron Systems, Phys. Rev., Vol. 46, pp. 618-622 (1934)

[2] T. Nakajima, M. Katouda, M. Kamiya, and Y. Nakatsuka: NTChem: A high-performance software package for quantum molecular simulation, Int. J. Quant. Chem., Vol. 115, pp. 349-359 (2015)

[3] NTChem, http://labs.aics.riken.jp/nakajimat_top/ntchem_e.html

[4] M. Feyereisen, G. Fitzgerald, and A. Komornicki: Use of approximate integrals in ab initio theory. An application in MP2 energy calculations, Chem. Phys. Lett., Vol. 208, pp. 359-363 (1993)

[5] D. E. Bernholdt and R. J. Harrison, Large-scale correlated electronic structure calculations: the RI-MP2 method on parallel computers, Chem. Phys. Lett., Vol. 250, pp. 477-484 (1996)

[6] M. Katouda and T. Nakajima: MPI/OpenMP hybrid parallel algorithm of resolution of identity second-order Møller–Plesset perturbation calculation for massively parallel multicore supercomputers, J. Chem. Theory Comput., Vol. 9, pp. 5373-5380 (2013)

[7] M. Katouda, A. Naruse, and T. Nakajima: Massively parallel algorithm and implementation of RI-MP2 energy calculation for peta-scale many-core supercomputers, manuscript in preparation.

[8] BLAS, http://www.netlib.org/blas/

[9] M. Katouda and S. Nagase: Efficient parallel algorithm of second-order Møller–Plesset perturbation theory with resolution-of-identity approximation (RI-MP2), Int. J. Quant. Chem., Vol. 109, pp. 2121-2130 (2009)

[10] NVIDIA CUDA ZONE, https://developer.nvidia.com/cuda-zone/

[11] cuBLAS http://docs.nvidia.com/cuda/cublas/

[12] T. H. Dunning Jr.: Gaussian basis sets for use in correlated molecular calculations. I. The atoms boron through neon and hydrogen, J. Chem. Phys., Vol. 90, pp. 1007-1022 (1989)

[13] F. Weigend, A. Köhn, and C. Hättig: Efficient use of the correlation consistent basis sets in resolution of the identity MP2 calculations, J. Chem. Phys., Vol. 116, pp. 3175-3183 (2002)

[14] S. Grimme: Improved second-order Møller–Plesset perturbation theory by separate scaling of parallel-and antiparallel-spin pair correlation energies, J. Chem. Phys., Vol. 118, pp. 9095-9102 (2003)

[15] L. Goerigk and S. Grimme: Efficient and Accurate Double-Hybrid-Meta-GGA Density Functionals—Evaluation with the Extended GMTKN30 Database for General Main Group Thermochemistry, Kinetics, and Noncovalent Interactions, J. Chem. Theory Comput., Vol. 7, pp. 291-309 (2011)

[16] S. F. Boys and F. Bernardi: The calculation of small molecular interactions by the differences of separate total energies. Some procedures with reduced errors, Mol. Phys., Vol. 19, pp. 553-566 (1970)

[17] R. Zacharia, H. Ulbricht, and T. Hertel: Interlayer cohesive energy of graphite from thermal desorption of polyaromatic hydrocarbons, Phys. Rev. B, Vol. 69, pp. 155406 (2004)

● TSUBAME e-Science Journal vol.14
Published 2/29/2016 by GSIC, Tokyo Institute of Technology ©
ISSN 2185-6028
Design & Layout: Kick and Punch
Editor: TSUBAME e-Science Journal - Editorial room
Takayuki AOKI, Toshio WATANABE,
Atsushi SASAKI, Yuki ITAKURA
Address: 2-12-1-E2-6 O-okayama, Meguro-ku, Tokyo 152-8550
Tel: +81-3-5734-2085 Fax: +81-3-5734-3198
E-mail: tsubame_ j@sim.gsic.titech.ac.jp
URL: http://www.gsic.titech.ac.jp/

TSUBAME

vol. 14

International Research Collaboration

The high performance of supercomputer TSUBAME has been extended to the international arena. We promote international research collaborations using TSUBAME between researchers of Tokyo Institute of Technology and overseas research institutions as well as research groups worldwide.

Recent research collaborations using TSUBAME

1. Simulation of Tsunamis Generated by Earthquakes using Parallel Computing Technique
2. Numerical Simulation of Energy Conversion with MHD Plasma-fluid Flow
3. GPU computing for Computational Fluid Dynamics

Application Guidance

Candidates to initiate research collaborations are expected to conclude MOU (Memorandum of Understanding) with the partner organizations/departments. Committee reviews the "Agreement for Collaboration" for joint research to ensure that the proposed research meet academic qualifications and contributions to international society. Overseas users must observe rules and regulations on using TSUBAME. User fees are paid by Tokyo Tech's researcher as part of research collaboration. The results of joint research are expected to be released for academic publication.

Inquiry

Please see the following website for more details.
http://www.gsic.titech.ac.jp/en/InternationalCollaboration

GSIC
Global Scientific Information and Computing Center