High-performance cloud computing for exhaustive protein–protein docking

Masahito Ohue[1,2,a)] Kento Aoyama[1,3] Yutaka Akiyama[1,2]

Abstract: Public cloud computing environments have achieved remarkable improvements in computational performance in recent years, and are also expected to be able to perform massively parallel computing. As the cloud enables users to use thousands of CPU cores and GPU accelerators casually, and various software types can be used very easily by cloud images, the cloud is beginning to be used in the field of bioinformatics. In this study, we ported the original protein–protein interaction prediction (protein–protein docking) software, MEGADOCK, into Microsoft Azure as an example of an HPC cloud environment. A cloud parallel computing environment with up to 1,600 CPU cores and 960 GPUs was constructed using four CPU instance types and two GPU instance types, and the parallel computing performance was evaluated. Our MEGADOCK on Azure system showed a strong scaling value of 0.93 for the CPU instance when H16 instance with 100 instances were used compared to 50, and a strong scaling value of 0.89 for the GPU instance when NC24 instance with 20 were used compared to 5. Moreover, the results of the usage fee and total computation time supported that using a GPU instance reduced the computation time of MEGADOCK and the cloud usage fee required for the computation. The developed environment deployed on the cloud is highly portable, making it suitable for applications in which an on-demand and large-scale HPC environment is desirable.

Keywords: cloud computing, Microsoft Azure, GPU computing, protein–protein docking, MEGADOCK

1. Introduction

The cloud computing environment is regarded as an important computing resource in large-scale data analysis [1]. The cloud computing environment is often used for calculation and analysis accompanied by big data, such as genomics and biomedicine [2, 3]. The development of public clouds such as Microsoft Azure, Amazon AWS, and the Google Cloud Platform has contributed to the performance of large-scale bioinformatics analysis on the cloud environment [4–6]. Bioinformatics problems including sequence homology searches [7–9], similarity searches of tertiary protein structures [10], *ab initio* tertiary protein structure prediction [11], and protein–ligand docking [12, 13] are applied in cloud computing environments as a computing resource.

Among the numerous merits of several existing cloud computing platforms, the pay-as-you-go concept whereby a user can use as much as he/she wishes at any time is the greatest advantage. Large-scale parallel computing using supercomputers enables large-scale simulation and processing of substantial amounts of data, but a user account approval procedure is required according to the institutional rules or the services are available only for the member of the organization possessing the supercomputer. In particular, several barriers exist to use for commercial purposes and owing to factors such as publicness, security and national strategy in supercomputer at public institution. Generally it is difficult for external people to use the public institution supercomputer casually. However, if it is on a cloud, anyone can use computational resources on thousands of cores instantly when necessary.

Distributed computing has mainly been selected as the method for cloud computing. With the development of grid computing, computation on the cloud by Apache Hadoop has been conducted extensively [2, 4, 6, 7] and support tools for constructing Hadoop clusters on the cloud have been established [14]. However, while Hadoop/MapReduce can easily construct a distributed task calculation environment, it is versatile and therefore contains an excessive amount of functions. These tools exhibit limited applicability to certain areas such as data mining, because MapReduce provides poor performance on problems with an iterative structure present in the linear algebra that underlies a substantial amount of data analysis [15]. To improve the performance and enable flexible design according to scientific applications, an original task distribution system has been constructed based on the message passing interface (MPI) in several cases [8].

Fortunately, AWS and Azure provide instances and networks with awareness of parallel high-performance computing (HPC). For example, in Azure, which was used in this research, an instance of a remote direct memory access (RDMA) network (InfiniBand) is also provided. Such an

[1] Department of Computer Science, School of Computing, Tokyo Institute of Technology
[2] Ahead Biocomputing, Co. Ltd.
[3] RWBC-OIL, AIST
[a)] ohue@c.titech.ac.jp

environment is expected to highly effective for parallel computing applications. However, information such as which instance should be used, the amount of scalability obtained, and the price has not been sufficiently clarified in previous studies.

Therefore, in this study, a large-scale parallel computation of a bioinformatics application was performed on several cloud instances with suggestions for the choice of the public cloud usage environment. We focused on protein–protein interaction predictions, particularly the protein–protein docking problem, as a bioinformatics application. Protein–protein docking, which is a computational method for predicting the structure of a protein complex from known component structures, is a powerful approach that facilitates the discovery of otherwise unattainable protein complex structures. Fast Fourier transform (FFT)-based rigid-body initial protein–protein docking tools are the mainstream of protein–protein docking [16]. Several applications also require a huge number of dockings, such as consensus-based refinement [17], large-scale interactome predictions [18,19], and the identification of protein binders [20,21]. We previously developed the supercomputer-powered software MEGADOCK [18, 22, 23], and we drew on this experience to develop a protein–protein docking tool for efficient HPC computation on the public cloud. A protein–protein docking environment that can achieve large-scale analysis on the cloud is necessary in the current global situation, in which large-scale computing environments are readily available on the cloud.

In this study, we demonstrated the implementation and performance of high-performance cloud protein–protein docking. We evaluated the parallelization efficiency (strong scaling) of MEGADOCK implemented on Microsoft Azure, and verified its usage efficiency for GPU instances.

2. Materials and Methods

2.1 Configuration of Azure cloud computing environment

A unit of computing environment on Azure is called an instance or virtual machine (VM). The machine architecture on Azure is composed of multiple VMs and storage, as illustrated in **Fig. 1**. Each VM and storage is first deployed from AzureCLI and then registered as a resource group in Azure. Thereafter, the computation task is executed on multiple VMs by means of MPI communication. The programs for the bulk VM deployment and bulk undeployment were developed in this study.

2.2 MEGADOCK: protein–protein docking tool

MEGADOCK [23] is our software for protein–protein interaction prediction. The 3D structures (PDB data) of two proteins for predicting interaction are input, and presence or absence of the interaction is output in the form of a score.

MEGADOCK is a multi-threaded implementation that uses OpenMP and runs on a multi-core CPU. Furthermore, a GPU-implemented version is available, which runs on the multiple GPUs using the CUDA library [24]. A multi-node parallel implementation version was also created by hybrid parallelization combined with MPI parallelization [18]. In this work, we constructed parallel implementations for both the CPU VMs and GPU VMs. The details of the parallelization are presented in the following subsection.

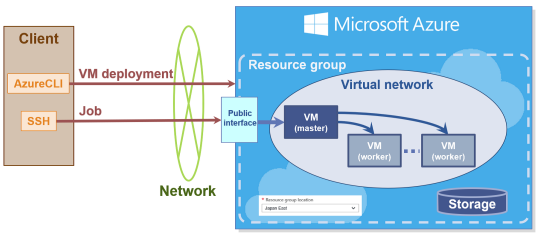

Fig. 1 Configuration of Azure cloud computing environment

2.3 Handling multiple VMs

In the multi-node implementation of MEGADOCK, a master–worker-type task dispatching is performed using MPI. Specifically, one process becomes the master process, and tasks are allocated to the worker processes while the remaining tasks and computing resources are monitored. The tasks are independent for each protein pair and can be data parallelized.

In Azure cloud, we adopted the master–worker-type task dispatching in parallel, whereby one process was the master process and the remaining resources were used to execute multiple worker processes, and MPI communication was used to realize the task dispatching for the protein–protein interaction prediction. Unlike the case in a normal cluster-type computing environment, the distance between real machines in a cloud computing environment tends to be large, and MPI implementation is generally not considered as suitable. However, as MEGADOCK does not require heavy communication between tasks (worker processes), it was expected that the large-scale parallelization would not cause serious slowdowns.

Among the Azure instances available for HPC applications, we targeted A9, DS14, H16, and H16r as CPU instances with 16 CPU cores, as well as NC24 and NC24r as GPU instances equipped with 24 CPU cores and 4 GPU chips. The details of each instance are displayed in **Table 1**. For each process to be able to use one GPU, a task dispatching was performed to run four processes per instance (on a VM). That is, the number of CPU cores allocated to each task was 1/4 of the number of cores in each VM: 4 cores for CPU instances and 6 cores for GPU instances.

2.4 Experimental settings

The dataset was the total of 59 protein heterodimeric complexes in the ZLAB protein–protein docking benchmark (version 1.0) [25]. The 59 heterodimers were divided, and all-to-all (cross) docking calculations were performed on the 59 receptor proteins and 59 ligand proteins.

Table 1 Details of Azure instances used in study

Instance	CPU	# cores	Total DP peak (CPU)	GPU	Network	Price (at Mar 2017)
DS14	Xeon E5-2660 @2.20 GHz×2	16	281.6 GFlops	N/A	-	1.39 USD/h
A9	Xeon E5-2670 @2.60 GHz×2	16	332.8 GFlops	N/A	RDMA	1.93 USD/h
H16	Xeon E5-2667v3 @3.20 GHz×2	16	691.2 GFlops	N/A	-	1.75 USD/h
H16r	Xeon E5-2667v3 @3.20 GHz×2	16	691.2 GFlops	N/A	RDMA	1.92 USD/h
NC24	Xeon E5-2690v3 @2.60 GHz×2	24	883.2 GFlops	Tesla K80×4	-	4.32 USD/h
NC24r	Xeon E5-2690v3 @2.60 GHz×2	24	883.2 GFlops	Tesla K80×4	RDMA	4.75 USD/h

Table 2 Results of MEGADOCK on Azure CPU instances (values in parentheses are the ratio of the calculation speed to H16.)

Instance	50 instances	100 instances	Strong scaling
DS14	3,283 s (0.47)	1,696 s (0.48)	0.968
A9	2,369 s (0.64)	1,352 s (0.61)	0.876
H16	1,527 s (1)	820 s (1)	0.931
H16r	1,640 s (0.93)	953 s (0.86)	0.861

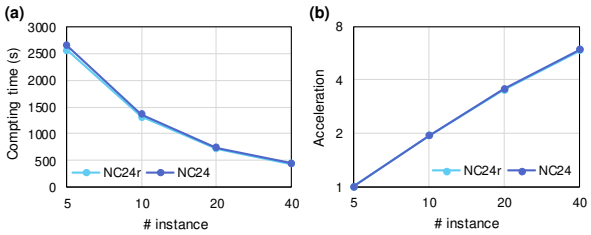

Fig. 2 Results of calculation time measurements on GPU instances: (a) computation time for each number of instances and (b) speed ratio with respect to 5 instances

3. Results and Discussion

3.1 MEGADOCK on multiple CPU instances

The results of the parallel execution of MEGADOCK on 50 and 100 instances using the CPU instances DS14, A9, H16, and H16r are presented in **Table 2**. The calculation time values were the median values measured three times. In this case, strong scaling was the value calculated as strong scaling $= (T_{50}/T_{100})/(100/50)$ when the computation times of 50 and 100 instances were T_{50} and T_{100}, respectively.

The experimental results demonstrated that the computation using the H16 instance was the fastest, followed by H16r, A9, and DS14. This ordering is naturally corresponding to the order of CPU performance (total DP peak) presented in Table 1.

When 100 H16 instances (1,600 CPU cores) were used, the calculation was completed in 820 s. This was the speed at which protein–protein docking calculations could be performed at 255 pairs per minute.

The calculation for the H16r instance was slightly slower than that for H16. The H16r is an instance that can use the RDMA network interface and exhibits higher communication performance than the H16, but MEGADOCK achieves higher performance even without RDMA network. An RDMA network may not be necessary for many bioinformatics applications in which data parallelization is possible. Moreover, as an instance with an RDMA network is more expensive than an instance without it, it is more reasonable not to use an RDMA network from a cost perspective.

The strong scaling was greater than 0.85 in the range of this measurement in all instances.

3.2 MEGADOCK on multiple GPU instances

Using the GPU instances NC24 and NC24r, we measured the computation times with using 5, 10, 20, and 40 instances. **Fig. 2** presents the measured calculation times and speed improvement rates. Owing to the limit of Microsoft Azure on the number of maximum concurrent GPUs (quota limit), the maximum number of allocated instances was 40. In the comparison between the NC24 and NC24r, the NC24r with an RDMA network slightly outperformed the NC24 in terms of speed, but the difference was very small. As with the CPU instance, the GPU instance would not require an RDMA network for this application.

NC24 is discussed below. When using 40 instances of NC24 (960 CPU cores and 160 GPUs), the calculation was completed within 448 s. This was faster than the result for the CPU instance indicated in Table 2 (H16: 1,600 CPU cores), and enabled 466 pairs of protein–protein docking to be performed per minute. For strong scaling, the parallelization efficiency of 20 instances was 0.89 for 5 instances, which was similar to that of the CPU instances. However, when 40 instances were used, the speed improvement was only 5.91-fold faster than that of 5 instances, with a strong scaling value of 0.74.

3.3 Which instance should be used from a cost perspective

CPU instance According to the comparison of CPU instances, the computation speed of the H16 instance was the most favorable. Comparing the H16 with the less expensive DS14, the speed improvement ratio was 1,696 s/820 s = 2.07. The price ratio between H16 (1.75 USD/h) and DS14 (1.39 USD/h) was 1.75 USD/ 1.39 USD = 1.26. As a result, it is more reasonable to use the H16 than the DS14, as the value of the speed improvement ratio is larger than the price ratio. Both A9 and H16r are slightly more expensive because they have an RDMA network, but MEGADOCK does not need to use these instances because no increase obtained in the computation speed when using an RDMA network. When using applications that require a powerful network, we recommend the H16r, which is approximately the same price as the A9, but provides higher CPU performance.

GPU instance A significant increase in the speed was achieved when using the GPU instance. However, unlike the H16 and DS14, the NC24 has 24 CPU cores, making a direct comparison difficult. In the following, we consider the max-

Table 3 Summary of results for H16 and NC24 instances

Instance	# inst.	CPU cores	GPUs	Time	Price (1 inst.)	Total fee*
H16	100	1,600	N/A	820 s	1.75 USD/h	39.9 USD
NC24	40	960	160	448 s	4.32 USD/h	21.5 USD

* The total fee was obtained by Price × Time (h) × # inst.

imum measurements at H16 (100 instances, 1,600 cores, and 820 s) and NC24 (40 instances, 960 cores and 160 GPUs, and 448 s) in terms of the cost. **Table 3** provides a summary of these results. In Table 3, the total fee was calculated by ignoring the time required for factors such as VM deployment and assuming that the product of {calculation time × number of instances} used was the total cloud usage time. Consequently, the same calculation could be performed for 21.5 USD for NC24, compared to 39.9 USD for H16. The NC24 has a shorter execution time and is almost twice as advantageous in terms of usage fees. For GPU-enabled applications, the use of GPU instances offers the potential to yield computational results rapidly and inexpensively, and active consideration thereof is recommended.

4. Conclusions

We constructed a computing environment for large-scale protein–protein docking calculations with the MEGADOCK software on the public cloud of Microsoft Azure, and performed large-scale parallel calculations on approximately 1,000 GPUs. We found that MEGADOCK provided the fastest GPU computation on the NC24 instance and the cloud computing cost was lower than that of using CPU instances. Large-scale data analysis with MEGADOCK requires high CPU and GPU performance, but does not require high communication performance. For bioinformatics applications similar in properties to MEGADOCK, it would be most cost-effective to use the NC24 instance or the similar instance without high-bandwidth network, like RDMA.

The use of the public cloud environment is advantageous owing to the portability and reproducibility of computing applications, and it allows for the rapid construction of large-scale applications such as the one investigated in this study. In addition to the protein–protein docking calculations demonstrated in this study, various other bioinformatics applications operating on the public cloud will certainly contribute to accelerating the research in this field.

Acknowledgments This work was supported by JSPS KAKENHI (20H04280), JST CREST (JPMJCR1303), AMED BINDS (JP18am0101112), Microsoft Corp, and Leave a Nest Co., Ltd.

References

[1] Hashem IAT, Yaqoob I, Anuar NB, et al. (2015) The rise of "big data" on cloud computing: Review and open research issues. Inf Syst 47:98–115.
[2] O'Driscoll A, Daugelaite J, Sleator RD (2013) 'Big data', Hadoop and cloud computing in genomics. J Biomed Inform 46(5):774–781.
[3] Sobeslav V, Maresova P, Krejcar O, et al. (2016) Use of cloud computing in biomedicine. J Biomol Struct Dyn 34(12):2688–2697.
[4] Karlsson J, Torreno O, Ramet D, et al. (2012) Enabling Large-Scale Bioinformatics Data Analysis with Cloud Computing. Proc IEEE ISPA2012:640–645.
[5] Shanahan HP, Owen AM, Harrison AP (2014) Bioinformatics on the Cloud Computing Platform Azure. PLoS ONE 9(7):e102642.
[6] Ekanayake J, Gunarathne T, Qiu J (2011) Cloud Technologies for Bioinformatics Applications. IEEE Trans Parallel Distrib Syst 22(6):998–1011.
[7] Matsunaga A, Tsugawa M, Fortes J (2008) CloudBLAST: Combining MapReduce and Virtualization on Distributed Resources for Bioinformatics Applications. Proc IEEE eScience2008:222–229.
[8] Lu W, Jackson J, Barga R (2010) AzureBlast: A Case Study of Developing Science Applications on the Cloud. Proc ACM HPDC'10:413–420.
[9] Gunarathne T, Wu T-L, Choi JY, et al. (2011) Cloud computing paradigms for pleasingly parallel biomedical applications. Concurr Comput Pract Exp 23(17):2338–2354.
[10] Mrozek D, Kutyła T, Małysiak-Mrozek B (2016) Accelerating 3D Protein Structure Similarity Searching on Microsoft Azure Cloud with Local Replicas of Macromolecular Data. Proc PPAM2015, LNCS 9574:254–265.
[11] Mrozek D, Gosk P, Małysiak-Mrozek B (2015) Scaling *Ab Initio* Predictions of 3D Protein Structures in Microsoft Azure Cloud. J Grid Comput 13:561–585.
[12] Farkas Z, Kacsuk P, Kiss T, et al. (2015) AutoDock Gateway for Molecular Docking Simulations in Cloud Systems. Cloud Computing with e-Science Applications:217–236.
[13] De Paris R, Ruiz DAD, de Souza ON (2015) A Cloud-Based Workflow Approach for Optimizing Molecular Docking Simulations of Fully-Flexible Receptor Models and Multiple Ligands. Proc IEEE CloudCom2015:495–498.
[14] Hodor P, Chawla A, Clark A, et al. (2015) cl-dash: rapid configuration and deployment of Hadoop clusters for bioinformatics research in the cloud. Bioinformatics 32(2):301–303.
[15] Qiu J, Ekanayake J, Gunarathne T, et al. (2010) Hybrid cloud and cluster computing paradigms for life science applications. BMC Bioinform 11:S3.
[16] Matsuzaki Y, Uchikoga N, Ohue M, et al. (2017) Rigid-Docking Approaches to Explore Protein-Protein Interaction Space. Adv Biochem Eng Biotechnol 160:33–55.
[17] Launay G, Ohue M, Santero JP, et al. (2020) Rescoring ensembles of protein-protein docking poses using consensus approaches. bioRxiv 2020.04.24.059469.
[18] Ohue M, Shimoda T, Suzuki S, et al. (2014) MEGADOCK 4.0: an ultra-high-performance protein-protein docking software for heterogeneous supercomputers. Bioinformatics 30(22):3281–3283.
[19] Hayashi T, Matsuzaki Y, Yanagisawa K, et al. (2018) MEGADOCK-Web: an integrated database of high-throughput structure-based protein-protein interaction predictions. BMC Bioinform 19:62.
[20] Wass MN, Fuentes G, Pons C, et al. (2011) Towards the prediction of protein interaction partners using physical docking. Mol Syst Biol 7:469.
[21] Zhang C, Tang B, Wang Q, et al. (2014) Discovery of binding proteins for a protein target using protein-protein docking-based virtual screening. Proteins 82(10):2472–2482.
[22] Matsuzaki Y, Uchikoga N, Ohue M, et al. (2013) MEGADOCK 3.0: a high-performance protein-protein interaction prediction software using hybrid parallel computing for petascale supercomputing environments. Source Code Biol Med 8:18.
[23] Ohue M, Matsuzaki Y, Uchikoga N, et al. (2014) MEGADOCK: An All-to-All Protein-Protein Interaction Prediction System Using Tertiary Structure Data. Protein Pept Lett 21(8):766–778.
[24] Shimoda T, Suzuki S, Ohue M, et al. (2015) Protein-protein docking on hardware accelerators: comparison of GPU and MIC architectures. BMC Syst Biol 9(Suppl 1):S6.
[25] Chen R, Mintseris J, Janin J, et al. (2003) A Protein-Protein Docking Benchmark. Proteins 52:88–91.

