Journal of Information Processing Vol.25 56–58 (Jan. 2017)

[DOI: 10.2197/ipsjjip.25.56]

Technical Note

Optimum Application Deployment Technology for Heterogeneous IaaS Cloud

Yoji Yamato[1,a)]

Received: July 27, 2016, Accepted: October 4, 2016

Abstract: Recently, cloud systems composed of heterogeneous hardware have been increased to utilize progressed hardware power. However, to program applications for heterogeneous hardware to achieve high performance needs much technical skill and is difficult for users. Therefore, to achieve high performance easily, this paper proposes a PaaS which analyzes application logics and offloads computations to GPU and FPGA automatically when users deploy applications to clouds.

Keywords: cloud computing, IaaS, FPGA, GPU, baremetal, OpenStack, heterogeneous cloud, compiler, PaaS

1. Introduction

Recently, Infrastructure as a Service (IaaS) clouds have been progressed [1], [2], [3], and users can use computer resources or service components on demand (For example, Refs. [4], [5], [6], [7], [8]). Early cloud systems are composed of many low price PC-like servers. Hypervisors, such as Xen or kernel-based virtual machines (KVMs) [9], virtualize these servers to achieve high computational performance using distributed processing technology, such as MapReduce [10].

However, recent cloud systems have changed to make the best use of recent advances in hardware power. For example, to use a large amount of core CPU power, some providers have started to provide baremetal servers which do not virtualize physical servers. Moreover, some cloud providers use special servers with strong graphic processing units (GPUs) to process graphic applications or special servers with field programmable gate arrays (FPGAs) to accelerate specific computation logics. For example, Microsoft's search engine Bing uses FPGAs to optimize search processing [11].

To utilize the recent advances in hardware power, users can benefit from high performance of their applications. However, to achieve high performance, users need to program appropriate applications for heterogeneous hardware and have much technical skill. Therefore, our objective is to enable users to achieve high performances easily. For this objective, cloud PaaS analyzes application logics and offloads computations to GPU and FPGA automatically when users deploy applications. The author previously proposed a Platform as a Service (PaaS) to select appropriate provisioning type of baremetal, container or virtual machine based on user requests [12]. In this paper, we investigate an element technology to offload part logics of applications to GPU and FPGA.

2. Problems of Existing Technologies

Recently, GPU programming, such as the compute unified device architecture (CUDA) [13], that involves GPU computational power not only for graphics processing has become popular. Furthermore, to program without walls between the CPU and GPU, the heterogeneous system architecture (HSA) [14], which allows shared memory access from the CPU and GPU and reduces communication latency between them, has been extensively discussed.

For heterogeneous programming, it is general to add and specify a code line to direct specified hardware processing. PGI Accelerator Compilers with OpenACC Directives [15] can compile C/C++/Fortran codes with OpenACC directives and deploy execution binary to run on GPU and CPU. OpenACC directives indicate parallel processing sections, then PGI compiler creates execution binary for GPU and CPU. Aparapi (A PARallel API) of Java [16] is API to call GPGPU (General Purpose GPU) from Java. Specifying this API, Java byte code is compiled to OpenCL and run when it is executed.

To control FPGA, development tools of OpenCL for FPGA are provided by Altera and Xilinx. For example, Altera SDK for OpenCL is composed of OpenCL C Compiler and OpenCL Runtime Library. OpenCL C Compiler compiles OpenCL C codes to FPGA bit stream and configures FPGA logic, OpenCL Runtime Library controls FPGA from applications on CPU using libraries of OpenCL API. Programmers can describe FPGA logic and control by OpenCL, then configured logic can be offloaded to specified FPGA.

However, these technologies have two problems. A) General language codes of C, C++ or Java need directives such as OpenACC or language extension such as OpenCL C. If we would like to achieve high performance, a timing to specify directives is very important and much technical knowledge of parallel processing is

[1] NTT Software Innovation Center, NTT Corporation, Musashino, Tokyo 180–8585, Japan

[a)] yamato.yoji@lab.ntt.co.jp

needed. B) There is no PaaS to utilize CPU/GPU/FPGA appropriately in clouds and users need to design how much GPU instances are needed.

The author previously proposed a PaaS to provide services based on user requests [12], [17], [18], [19]. The work of Ref. [12] can provision baremetal, container or virtual machine appropriately, thus enhancing the idea [12], we can provide PaaS to select CPU/GPU/FPGA and can partly solve B). This paper's target is to solve A) by an element technology by utilizing GPU and FPGA from general language applications, and this technology is an improved point from Ref. [12]. Complex applications such as synchronous execution of FPGA and GPU are out of the scope of this paper. The work of Ref. [19] is our previous work in a conference paper and this paper improves it.

3. Proposal of Optimum Application Deployment Technology for Heterogeneous IaaS

In this section, we propose PaaS of cloud provider with optimum application deployment technology. Our proposed technology involves a PaaS, an IaaS controller, such as OpenStack [20], heterogeneous cloud hardware with GPU, CPU, FPGA, HDD/SSD storage, and a code patterns database (DB). Our PaaS is one of aPaaS (application PaaS). In general aPaaS provides an application environment. Our PaaS has characteristics for application deployment to offload part logics of applications to GPU and FPGA. And the figure describes OpenStack as an IaaS controller, but OpenStack is not a precondition of the proposed technology and other controllers such as CloudStack can be adopted.

3.1 Processing Steps

Figure 1 shows system architecture and application deployment steps. There are 7 steps to deploy applications.

1. Users specify their applications to deploy on clouds to PaaS system. Users need to upload source codes of applications to PaaS.

2. PaaS analyzes application source codes, compares codes to code patterns DB and detects similar code patterns. Here, code patterns DB retains codes which are offloadable to GPU and FPGA and corresponding OpenCL patterns. To detect similar codes, we use similar code detection tools such as CCFinderX [21]. Similar code detection tools can detect specified code patterns of FFT (Fast Fourier Transformation), encryption and decryption processing, graphic processing and so on from users' application codes. In these examples, FFT, encryption and decryption processing can be offloaded to FPGA with accelerated configurations of these processing, and graphic processing can be offloaded to GPU.

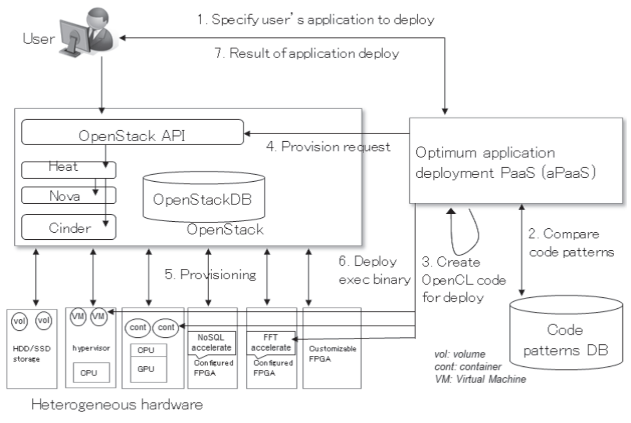

Fig. 1 Proposed optimum application deployment steps for heterogeneous IaaS cloud.

3. PaaS extracts OpenCL language codes for offloadable processing to GPU and FPGA detected in step 2. OpenCL language is the major language for heterogeneous programming and describes processing which runs on FPGA/GPU. Therefore, in this step, PaaS adds specified OpenCL line for offload sections based on code patterns DB.

4. PaaS sends a provisioning request of creating run environment for applications to OpenStack. For example, when we need GPU resources, containers are provisioned on GPU servers because VMs cannot sufficiently control GPUs. And when we need FPGA, servers with FPGA board are provisioned by baremetal provisioning such as Ironic. Basically PaaS selects pre-configured FPGA server for specified logics such as FFT from multiple FPGA servers. However, there is no desired configuration of FPGA, PaaS may provision a non-configured FPGA server and customized configuration may be done before actual applications run.

5. OpenStack creates new computer resources for specified applications. Note that if applications need to create not only one compute server but several resources, such as virtual routers and storage, PaaS sends templates that describe the environment structures by JavaScript Object Notation (JSON) and provisions them at once by OpenStack Heat or other orchestration technology.

6. PaaS deploys application execution binary to provisioned servers. When PaaS deploys execution binary, existing tools of each vendor such as Altera SDK for OpenCL can be used to deploy.

7. PaaS returns application deployment results information of which servers are deployed to process and how much the cost of server usage is and so on. If users agree the deployment results, users start to use applications and usage fees are also started for which users will be charged. If users disagree the results, PaaS deletes resources by OpenStack Heat stack-delete API. After resources deletion, users may re-deploy. And if users would like to reconfigure FPGA, users specify FPGA configuration in this step 7 timing.

By these processing steps, users can deploy applications which are written by general language to heterogeneous cloud. The main contributions of our proposal are comparing source codes with code pattern DB to find offloadable logics and extracting OpenCL codes automatically. There is a merit for users to achieve high performance without programming knowledge of GPU and FPGA.

4. Conclusion

This paper proposed a PaaS to offload application logics to GPU and FPGA automatically when users deploy applications to clouds. Proposed PaaS analyzed source codes of applications, detected offloadable logics to GPU and FPGA using similar code detection technology and predefined code patterns, created OpenCL codes and deployed them. This can enable high performance applications easily for users. In the future, we will verify the proposed technology performance and validity for general application codes. We also need to study how to deploy complex applications.

Acknowledgments The author would like to thank Hideki Hayashi and Shinichi Nakagawa who were managers of this research.

References

[1] Yamato, Y., Nishizawa, Y., Muroi, M. and Tanaka, K.: Development of Resource Management Server for Production IaaS Services Based on OpenStack, *Journal of Information Processing*, Vol.23, No.1, pp.58–66 (Jan. 2015).

[2] Yamato, Y., Nishizawa, Y., Nagao, S. and Sato, K.: Fast and Reliable Restoration Method of Virtual Resources on OpenStack, *IEEE Trans. Cloud Computing*, DOI: 10.1109/TCC.2015.2481392 (Sep. 2015).

[3] Yamato, Y., Shigematsu, N. and Miura, N.: Evaluation of Agile Software Development Method for Carrier Cloud Service Platform Development, *IEICE Trans. Information & Systems*, Vol.E97-D, No.11, pp.2959–2962 (Nov. 2014).

[4] Yokohata, Y., Yamato, Y., Takemoto, M., Tanaka, E. and Nishiki, K.: Context-Aware Content-Provision Service for Shopping Malls Based on Ubiquitous Service-Oriented Network Framework and Authentication and Access Control Agent Framework, *IEEE Consumer Communications and Networking Conference* (*CCNC 2006*), pp.1330–1331 (Jan. 2006).

[5] Yamato, Y., Ohnishi, H. and Sunaga, H.: Development of Service Control Server for Web-Telecom Coordination Service, *IEEE International Conference on Web Services* (*ICWS 2008*), pp.600–607 (Sep. 2008).

[6] Nakano, Y., Yamato, Y., Takemoto, M. and Sunaga, H.: Method of creating web services from web applications, *IEEE International Conference on Service-Oriented Computing and Applications* (*SOCA 2007*), pp.65–71 (June 2007).

[7] Sunaga, H., Takemoto, M., Yamato, Y., Yokohata, Y., Nakano, Y. and Hamada, M.: Ubiquitous Life Creation through Service Composition Technologies, *Proc. World Telecommunications Congress 2006* (*WTC2006*) (May 2006).

[8] Yamato, Y., Tanaka, Y. and Sunaga, H.: Context-aware Ubiquitous Service Composition Technology, *The IFIP International Conference on Research and Practical Issues of Enterprise Information Systems* (*CONFENIS 2006*), pp.51–61 (Apr. 2006).

[9] Kivity, A., Kamay, Y., Laor, D., Lublin, U. and Liguori, A.: KVM: The Linux virtual machine monitor, *The 2007 Ottawa Linux Symposium*, pp.225–230 (July 2007).

[10] Dean, J. and Ghemawat, S.: MapReduce: Simplified data processing on large clusters, *Proc. 6th Symposium on Opearting Systems Design and Implementation* (*OSDI '04*), pp.137–150 (Dec. 2004).

[11] Putnam, A., Caulfield, A.M., Chung, E.S., Chiou, D., Constantinides, K., Demme, J., Esmaeilzadeh, H., Fowers, J., Gopal, G.P., Gray, J., Haselman, M., Hauck, S., Heil, S., Hormati, A., Kim, J.-Y., Lanka, S., Larus, J., Peterson, E., Pope, S., Smith, A., Thong, J., Xiao, P.Y. and Burger, D.: A reconfigurable fabric for accelerating large-scale datacenter services, *Proc. 41th Annual International Symposium on Computer Architecture* (*ISCA '14*), pp.13–24 (June 2014).

[12] Yamato, Y.: Server Structure Proposal and Automatic Verification Technology on IaaS Cloud of Plural Type Servers, *International Conference on Internet Studies* (*NETs 2015*), Tokyo (July 2015).

[13] Sanders, J. and Kandrot, E.: CUDA by example: An introduction to general-purpose GPU programming, Addison-Wesley, ISBN-0131387685 (2011).

[14] Kyriazis, G.: Heterogeneous System Architecture: A Technical Review, White paper, HSA Foundation (Aug. 2012). available from ⟨http://developer.amd.com/wordpress/media/2012/10/hsa10.pdf⟩.

[15] PGI compiler web site, available from ⟨https://www.pgroup.com/resources/accel.htm⟩.

[16] Aparapi web site, available from ⟨http://developer.amd.com/tools-and-sdks/opencl-zone/aparapi/⟩.

[17] Yamato, Y., Ohnishi, H. and Sunaga, H.: Study of Service Processing Agent for Context-Aware Service Coordination, *IEEE International Conference on Service Computing* (*SCC 2008*), pp.275–282 (July 2008).

[18] Yamato, Y., Nakano, Y. and Sunaga, H.: Study and Evaluation of Context-Aware Service Composition and Change-Over Using BPEL Engine and Semantic Web Techniques, *IEEE Consumer Communications and Networking Conference* (*CCNC 2008*), pp.863–867 (Jan. 2008).

[19] Yamato, Y.: Proposal of Optimum Application Deployment Technology for Heterogeneous IaaS Cloud, *2016 6th International Workshop on Computer Science and Engineering* (*WCSE 2016*), pp.34–37, Tokyo (June 2016).

[20] OpenStack web site, available from ⟨http://www.openstack.org/⟩.

[21] CCFinder web site, available from ⟨http://www.ccfinder.net/⟩.

Yoji Yamato received his B.S., M.S. degrees in physics and Ph.D. degree in general systems studies from University of Tokyo, Japan in 2000, 2002 and 2009, respectively. He joined NTT Corporation, Japan in 2002. There, he has been engaged in developmental research of Cloud computing platform, Peer-to-Peer computing and Service Delivery Platform. Currently he is a senior research engineer of NTT Software Innovation Center. Dr. Yamato is a member of IEEE and IEICE.