NATIONAL AUTONOMOUS UNIVERSITY OF MEXICO
UNIVERSITY PROGRAM OF STUDIES ON ASIA AND AFRICA

“WORKING *PAPER 3*”

Parallel Processing and Parallelizing Compilation Techniques for “Green Computing"

Yasutaka Wada

pueaa.unam.mx

The PUEAA Working Paper series disseminates preliminary results of research work on the Asia-Africa region with the aim of fostering the exchange and debate of ideas. The contents of the Working Papers, as well as the conclusions derived from them, are the sole responsibility of the authors and do not necessarily reflect those of the University Program of Studies on Asia and Africa (PUEAA).

All titles are available on the PUEAA website: http://pueaa.unam.mx/observatorio-asia-africa/working-paper

Working Paper PUEAA No. 3. Parallel Processing and Parallelizing Compilation Techniques for "Green Computing"

DOI https://doi.org/10.22201/pueaa.001r.2022

Publication Date January 2022

Programa Universitario de Estudios sobre Asia y África

Calle de Filosofía y Letras 88

04360, Copilco Universidad

Coyoacán, Ciudad de México

This issue was edited by María del Carmen Uribe Rangel. Cover design and illustration: Yussef A. Galicia Galicia. Editorial support: Lesly Abigail Olivares Quintana and María Fernanda Ortiz Castañeda.

Made in Mexico / Hecho en México

Parallel Processing and Parallelizing Compilation Techniques for "Green Computing"

Yasutaka Wada

Resumen

La cuarta revolución tecnológica ha supuesto un gran avance en los procesos de fabricación y las comunicaciones humanas. Si bien los procesadores se han vuelto cada vez más eficientes, tanto en velocidad, capacidad y en consumo de energía, su funcionalidad respecto a este último punto aún debe mejorar. Las más recientes innovaciones representan una oportunidad para crear una "computación verde" y no solo electrónicos y softwares más amigables con el ambiente, sino también usar su nueva eficiencia para mejorar nuestras actividades diarias, así como los diseños de nuestras mismas ciudades para volverlas más sustentables con el ambiente. Estos nuevos sistemas computarizados deberán ser aplicados también conforme a los factores socioeconómicos que deben ser tomados en cuenta para poder ser modificados en favor de la sostenibilidad y la eficiencia.

Abstract

The fourth technological revolution has brought great advances in manufacturing processes and human communications. Although processors have become increasingly efficient, both in speed, capacity and energy consumption, their functionality regarding this last point has yet to improve. The latest innovations represent an opportunity to create "green computing" and not only more environmentally friendly electronics and software, but also to use their new efficiency to improve our daily activities, as well as the designs of our cities themselves to make them more environmentally sustainable. These new computerized systems must also be applied in accordance with the socioeconomic factors that must be taken into account in order to be modified in favor of sustainability and efficiency.

Introduction

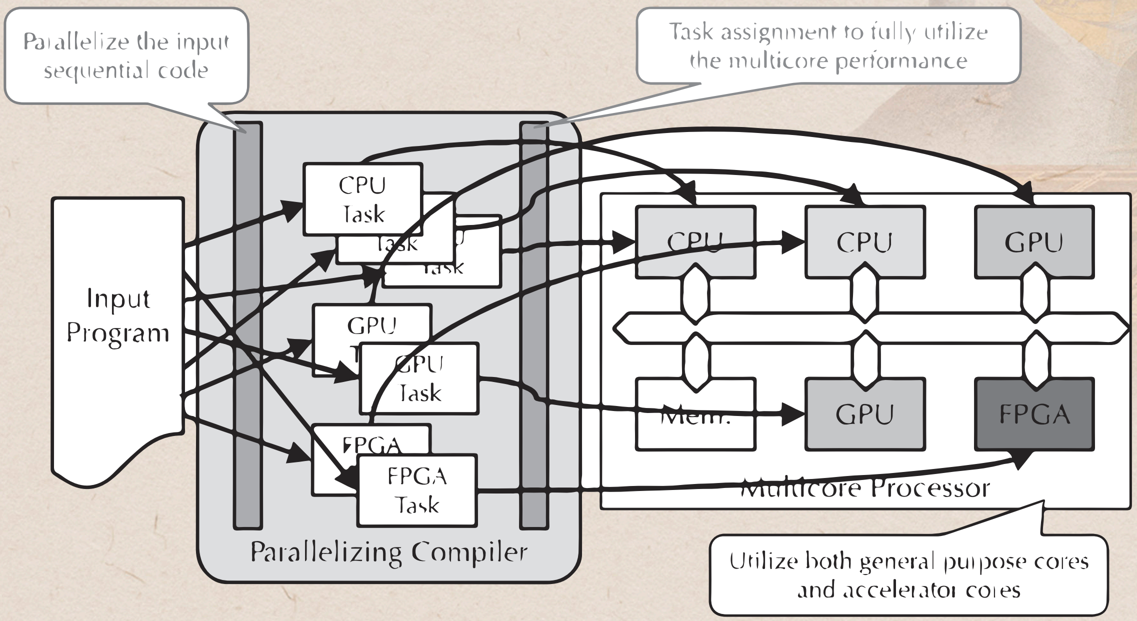
Most of the current processors include functionalities such as DVFS (dynamic voltage/ frequency scaling), clock gating (clock signal supply control), and power gating (power supply control); however, there are trade-offs between their performances and power consumption. Hence, if the power consumption of a computer system is reduced, its performance decreases, in general. Our approaches for utilizing the functionalities based on parallel processing consider cooperation between the software and hardware. The parallel processing technique significantly reduces the execution time of a user application, while distributing the computational tasks in the application to the processor cores in the system, considering the dependencies and task execution order. This speedup enables the selection of the trade-off between the power and performance; DVFS, power gating, and clock gating can be applied for minimizing the power consumption to execute the application with the required performance. These types of software optimization/parallelization approaches render it possible to completely utilize the hardware functionalities. At times, application optimization and parallelization can broaden the gap between the given power budget and the actual power consumption of a computer system. Hardware overprovisioning, which involves the preparation of more hardware than the given power supply can drive, is one of the solutions to this problem. To make this type of computer system, we need to control the system so that it does not consume more power than the given budget. Software techniques that appropriately assign the power budget to each component in the system will play an indispensable role in future computers.

Another aspect of “Green Computing” is taking advantage of the high-performance computational power achieved using parallel processing techniques to realize smarter cities. Although our daily activities consume natural resources, green high-performance computing platforms can provide more efficient social systems, conserving more resources and providing more value than the system consumes. Currently, there are many small computer systems, such as smartphones and sensors, and it is no longer impossible to connect them with platforms to use data and information obtained all over the city. By integrating such high-performance computer platforms with social systems, we believe that we can make safe, good-for-living, and disaster-resistant cities.

This paper surveys and introduces our research effort and ongoing research projects in the field of high-performance computing (Wada *et al*., 2011) (Shimaoka *et al*., 2015) (Inadomi *et al*., 2015) (Wada *et al*., 2018) (Wada, 2017) green computing (Hayashi *et al*., 2012) (Wada *et al*., 2015) (Wada, 2018), and smart cities (Shimaoka *et al*., 2015) (Samra *et al*., 2015). These research activities are mainly focused on improving the energy efficiency of computer systems for completely utilizing the given power to contribute toward the preservation of the global "environment," and improve the computation performance itself to contribute toward the human society "environment." Although parallel processing is a key technology to realize the above mentioned techniques, the development of parallel applications at times involves considerable cost and effort by the programmers and users. Moreover, additional cost and effort are required to optimize the applications for better performance and lower power consumption. To ease these demands of parallel programming, we have been researching automatic parallelization techniques, task scheduling/assignment algorithms, and a program compilation framework that enables parallelization and power-efficiency optimization simultaneously. Utilizing this compilation framework, we can achieve high performance along with better energy efficiency with minimal user/programmer effort. Furthermore, this paper presents the performance improvement that can be achieved with parallel processing for smarter cities through certain example applications.

The rest of this paper is organized as follows: Section 2 discusses parallel processing techniques with automatic parallelization for homogeneous and heterogeneous parallel computer systems. Section 3 describes our power reduction and power management methodologies with parallel processing techniques. Section 4 introduces our research effort and ongoing research project for the development of smarter and sustainable cities regarding computer systems. Finally, Section 5 concludes this paper.

Figure 1. Parallelizing Compiler


Parallelize the input sequential code
Task assignment to fully utilize the multicore performance
Input Program
CPU Task
CPU Task
CPU Task
GPU Task
GPU Task
FPGA Task
FPGA Task
CPU
CPU
GPU
Mem.
GPU
FPGA
Multicore Processor
Parallelizing Compiler
Utilize both general purpose cores and accelerator cores

Source: Own elaboration.

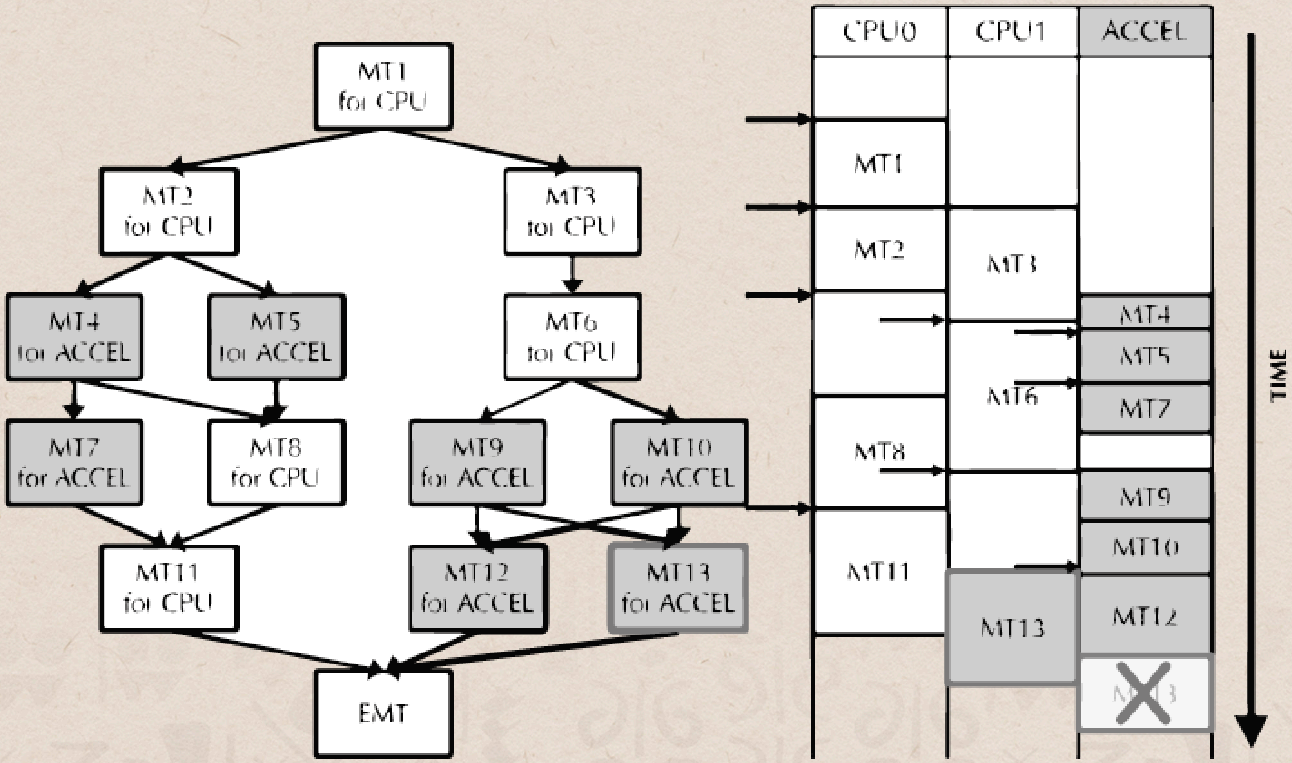
Currently, the software that runs on computer systems includes various applications such as autonomous driving, artificial intelligence, big data processing, and virtual reality. Hence, massive computational power is required, with an ever increasing demand. Previously, computer-system performance was mainly improved by improving a single thread performance and the processor clock frequency. However, now there is almost no room for improving the processor clock frequency and single thread performance due to the difficulty involved in system cooling and the complexity of the processor core design.

To manage such problems in computer system performance, most of the current computer systems, from embedded systems to supercomputers, employ multiple processors and/or multicore processors that utilize numerous transistors on ICs for improving the system performance. In such parallel computer systems, it is necessary to parallelize applications to obtain sufficient performance. Although parallel processing is an essential technique to maximize computer- system performance, the development of parallel applications is expensive and time consuming. Further, it is required to develop and realize automatic parallelization, which parallelizes a sequential application automatically, to ease this problem.

As a traditional method for parallelizing an application, loop-level parallel processing, which distributes loop iterations in a loop to the processor cores on the system, is often used. Loop-level parallel processing focuses only on the loops in an application and does not parallelize the remaining parts. To obtain increased performance from parallel computer systems, we need to consider the utilization of other types of parallelism in applications.

From the software structure, various elements with various granularities, such as statements, loops, and function calls, can be recognized. If these various parallelism granularities are utilized, it is possible to sufficiently extract parallelism and performance from an application.

Figure 2. Task Scheduling



Source: Based on Fig. 1 in Hayashi *et al.* (2012).

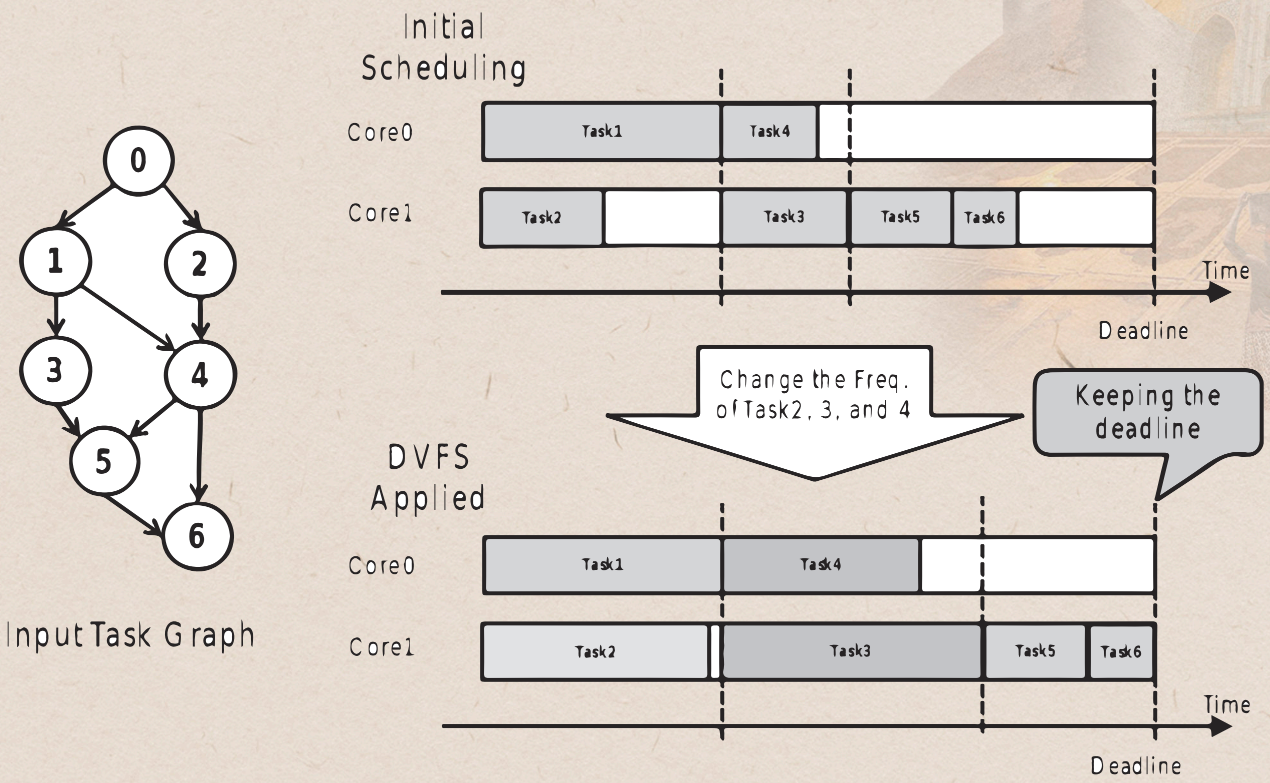
To solve the problems related to the cost of parallelizing an application, we have been developing an automatic parallelizing compiler system, utilizing the various parallelism granularities (Hayashi *et al*., 2012) (Wada *et al*., 2011) (Shimaoka *et al*., 2015). This automatic parallelizing compiler realizes coarse grain task parallel processing, which utilizes parallelism among the function calls in an application. Figure 6 shows the overall structure of our parallelizing compiler. This compiler decomposes the input program into coarse grain tasks and schedule the tasks to process cores on the target processor chip.

In coarse grain task parallel processing, the compiler decomposes the input application into coarse grain tasks such as basic blocks, loops, and function calls. This decomposition can be applied hierarchically, considering the structure of the input application. After coarse grain task decomposition, the compiler can analyze the dependencies among the tasks. These dependencies determine the tasks that can be executed in parallel and the order of task execution. In actual implementation, the compiler represents the dependencies among the coarse grain tasks as a directed acyclic graph (DAG), and utilizes it for task scheduling, to assign the tasks to the processor cores on the computer system for running the application. Figure 2 shows an example of task scheduling. In this figure, our compiler schedules the coarse grain tasks considering the dependencies among them and their processing costs on each core. Through task decomposition and task scheduling, the parallelizing compiler can extract parallelism from the input application and completely utilize the parallelism with the target parallel computer system.

Power/Energy Control with Parallel Processing

Although the primary aim of parallel processing is to improve the performance of application execution, namely, reducing the execution time of the application, this performance improvement with parallel processing reduces and controls the system power consumption, while executing the application. Current computer systems are equipped with various functionalities, such as DVFS and power gating, to reduce and control the power/performance, and parallel processing enables the utilization of such functions.

Figure 3. Finding Power Reduction Opportunities



Source: Hayashi *et al.* (2011).

For embedded systems such as smartphones, automotive systems, and robot controllers, power consumption reduction is crucial for long battery life, natural air cooling, etc. Moreover, the applications that run on such systems should meet the deadline for task completion. Current embedded applications require high performance to realize rich user experiences, and smooth control of cars and robots. Hence, similar to PCs and high-performance computing (HPC) systems, embedded systems need to use multiprocessors and multicore processors. Thus, we need to parallelize embedded applications such that they are sufficiently fast to meet the given deadline. Parallelizing such applications can enable the stable operation of the embedded system. In addition, particularly in the case of coarse grain task parallelization, we can maximize the opportunities for applying DVFS and power gating in the processors to reduce their power consumption.

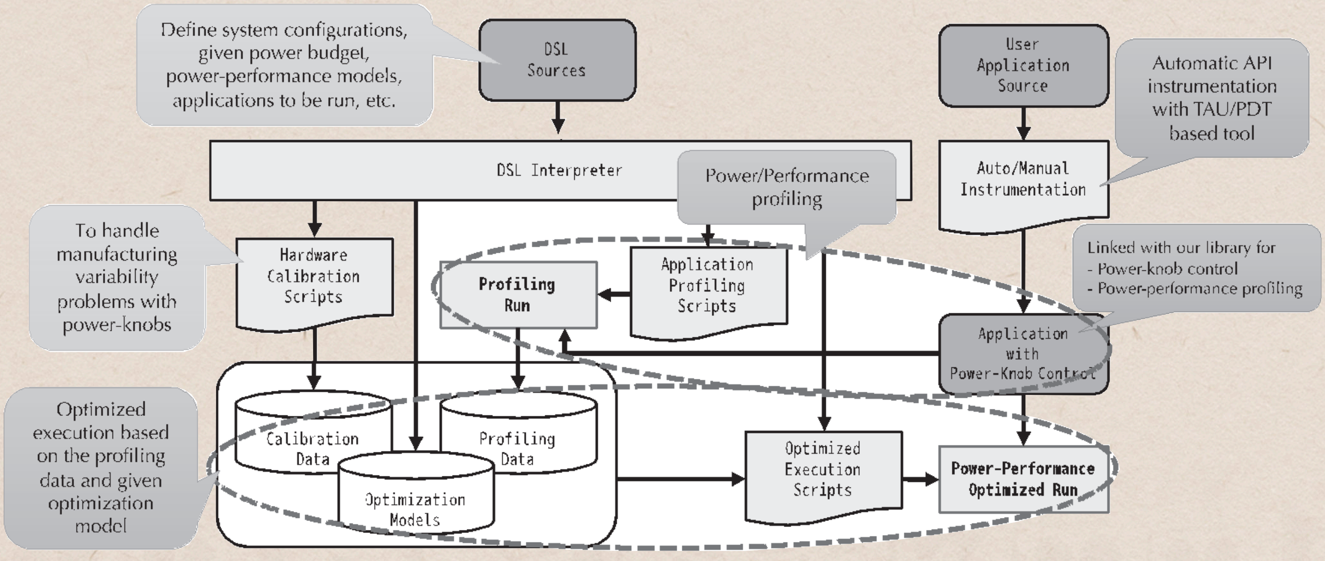
The dependencies among the coarse grain tasks determine the task-scheduling optimization by the compiler to obtain higher performance; opportunities to apply DVFS and power gating can be established, considering the task scheduling result and the given deadline to complete the application. The most straightforward method to apply such power reduction is to execute tasks at the maximum processor core frequency, until the deadline. Another option is to slow down the processor core frequency to meet the given deadline. Among these strategies, our compiler can apply the most suitable, based on the characteristics and parallelism of the application (Hayashi *et al.,* 2012). Figure 3 shows an example of DVFS application. In this figure, we have the given deadline to complete the program shown as the input task graph, and there is some room to apply DVFS while keeping the deadline. In this example, we can slow down the operating frequency for Task 3 and 4 without violating the deadline. Also, we can slow down the operating frequency for Task 2 without any influence for other tasks.

In real applications such as audio encoding and image processing, our power reduction scheme with automatic parallelization resulted in a power reduction of more than 70 [%], compared to the case without the scheme (Hayashi *et al.,* 2012). If power reduction is not considered, we can obtain a speedup of more than 16 times with 12 cores on a RP-X heterogeneous multicore processor by simultaneously utilizing general purpose processor cores and accelerator cores, compared to sequential execution with a general purpose core. In simulations involving synthetic applications on a homogeneous multicore processor, we achieved better power reduction, compared to conventional task scheduling strategies, while maintaining the given deadline for completing the application (Wada *et al.*, 2015).

Software Framework for Power Management

For HPC, power consumption is one of the significant constraints for scaling the system performance. Therefore, it is required to improve the system power efficiency to realize future exascale HPC systems. HPC applications are to be optimized as much as possible for obtaining better performance; however, this optimization expands the gap between the given power budget and actual power usage by the HPC system because software optimization enables more efficient usage of the hardware resources. Hence, hardware overprovisioning can be the mainstream of HPC systems (Inadomi *et al.*, 2015) (Wada *et al.*, 2018). Hardware overprovisioned systems employ considerable hardware resources, which cannot be driven with the given power budget; however, the system software controls the power and usage of hardware resources such that the budget is not exceeded. With cooperation between the hardware and system software, the given power budget can be completely utilized to realize more efficient HPC systems. Such power management is a complicated process involving the collection and analysis of statistics from both hardware and software, power allocation and control by the available power-knobs (hardware resources, whose performance and power can be controlled by software), code instrumentation, etc. on. Due to the numerous hardware components, it would become more complicated to manage the power consumption in future HPC systems. Hence, one of the most important research issues is the development of a power management framework that enables more efficient power consumption and distribution.

Figure 4. Power Performance Management Software Framework


Define system configurations, given power budget, power-performance models, applications to be run, etc.
DSL Sources
User Application Source
Automatic API instrumentation with TAU/PDT based tool
DSL Interpreter
Power/Performance profiling
Auto/Manual Instrumentation
To handle manufacturing variability problems with power-knobs
Hardware Calibration Scripts
Profiling Run
Application Profiling Scripts
Linked with our library for
- Power-knob control
- Power-performance profiling
Application with Power-Knob Control
Optimized execution based on the profiling data and given optimization model
Calibration Data
Profiling Data
Optimization Models
Optimized Execution Scripts
Power-Performance Optimized Run

Source: Based on Fig. 2 in Wada *et al.* (2018).

Currently, we are developing a software framework to automate the power management process in HPC systems (Wada *et al*., 2018) (Wada, 2017). In this framework, we aim to provide a simple user interface, standard structure and design pattern for developing libraries to control the power-knobs, and easy-to-use and straightforward domain specific language (DSL) for developing power-performance models to optimize power allocation among the hardware resources in a system.

Figure 4 shows the power management workflow with our software framework. Our power management framework is equipped with a simple DSL compiler, which generates several scripts:

a) A script to gather information on the hardware resources in the target system.

b) A script to generate the profile data of the input application.

c) A script to run the application under the power management strategy described in the input DSL source code.

Users can obtain power-performance optimized application execution by running these scripts. In addition, our framework instruments API calls into the input application source code to control the power knobs at runtime, according to the decision made based on the power-performance model defined by the DSL. Using the earlier version of our power management software framework, we had succeeded in realizing an easy and straightforward method for controlling the power consumption of application execution, under simple power-performance models.

Currently, we are actively developing the next version of the framework to include more complicated power-performance models, and power management strategies with simpler interfaces (Wada, 2017).

Towards Smart Cities

To create smarter cities with respect to computer systems, one of the essential aspects is their energy efficiency, as previously described. Additionally, higher performance must be realized not only with benchmark applications but also with "real" applications. In this section, we present examples of the acceleration of certain real applications, and our ongoing research project for managing the performance and power consumption for various scales of computer systems, including virtualized systems such as cloud platforms.

Optimizing Real Applications

The evaluation and improvement of computer system performance with benchmark programs is necessary to investigate and develop better computer systems; however, applications for solving real social problems include various characteristics, and the applicability of our parallel processing and power reduction/management methods need to be extended to handle such applications. In the realization of smarter cities, traffic and disaster problems are inevitable. Hence, we have parallelized and optimized some applications in these areas.

The first example of a real application is traffic optimization (Samra *et al.*, 2015) (Sen and Head, 1997). Optimization of traffic signal control is adequate for avoiding traffic jams and can contribute to resource saving as well. Furthermore, appropriate traffic signal control can assist in saving human lives by preventing traffic accidents. In our previous research, we had developed a parallel algorithm for traffic signal control and had obtained a performance of order O(n), even though the performance of the original algorithm that employed a dynamic programming method was of order O(n2). Although additional effort is still required to realize real-time optimized traffic control with multiple intersections, this parallelization and improvement can contribute to various applications for smarter cities.

Another challenging example is earthquake simulation (Shimaoka *et al.*, 2015). In Japan and Mexico, earthquakes and tsunamis are the most severe disasters that require countermeasures. If we can realize “super-realtime” earthquake simulation with parallel processing, an alert system can be realized such that people can be evacuated in time. We applied our coarse grain task parallel processing in an earthquake simulator, GMS, developed by NIED (GMS: Ground Motion Simulator, n.d.), and achieved a speed up of approximately x100, with 128 processor cores, compared to the sequential execution on an IBM server with POWER8 chips. This scalable performance improvement with coarse grain task parallelization can contribute toward the realization of safer and smarter cities.

Towards Computer Systems for Sustainable Cities

As described in the previous sections, our parallel processing and power reduction strategies can achieve effective performance and power reduction in various parallel computing systems. However, the most power consuming computer system is the cloud environment, for which the power consumption needs to be managed. Different from typical computer systems, cloud platforms rely on virtualization techniques. Hence, it is not easy to apply DVFS and power gating in such an environment because various virtual machines can run on the same physical hardware. This situation renders it difficult to find opportunities to apply DVFS and power gating, and our existing parallel processing and power reduction schemes cannot be applied as such.

Thus, we are developing a new software environment to connect a compiler that analyzes the individual application and the cloud platform, which manages the virtual machines and underlying hardware. By orchestrating various scales/types of systems, including IoT devices, PCs, servers, and supercomputers, we can contribute to the realization of smarter and sustainable societies.

Conclusions

Our research efforts and ongoing research projects were surveyed and summarized in this paper. The main focus of this research was to achieve adequate computer system performance by applying parallel processing techniques. The utilization of various types (granularities) of parallelism in applications is the key in our parallel processing techniques, and considerable effort has been made in realizing an automatic parallelizing compiler to easily generate parallel applications. With coarse grain task parallel processing in particular, opportunities can be extracted to not only achieve fast execution time of the input application but also to apply DVFS and power gating for optimizing the power/energy consumption of computer systems in running the application.

In our ongoing research projects related to power/energy efficient computer systems, we are mainly focusing on the development of a software framework to provide an easy-to-use and productive power management environment for HPC systems, and for applying our power reduction/optimization strategies in virtualized systems, such as cloud platforms. These projects can contribute toward realizing smarter and sustainable cities, from the telecommunications, information, and communication technology (ICT) point of view.

Acknowledgements

This work was partially supported by JSPS KAKENHI, Grant Numbers: 24700055 and 17K12665, and by the Japan Science and Technology Agency (JST) CREST program for the research project, "Power Management Framework for Post-Petascale Supercomputers". The author would like to thank all the collaborators in the research projects mentioned in this paper for their kind support and fruitful discussions.

References

Ground Motion Simulator (GMS). (n.d.), "GMS: Ground Motion Simulator". Available: http://www.gms.bosai.go.jp/GMS/

Hayashi, A. *et. al.* (2011), "Parallelizing Compiler Framework and API for Power Reduction and Software Productivity of Real-Time Heterogeneous Multicores". In: Cooper K. *et. al.*(eds) *Languages and Compilers for Parallel Computing. LCPC 2010*. Lecture Notes in Computer Science, vol. 6548. *Springer.*

Hayashi, A. *et. al.* (2012), "Parallelizing Compiler Framework and API for Heterogeneous Multicores", *IPSJ Transactions on Advanced Computing Systems [in Japanese]*, vol. 5, no. 1, pp. 68-79.

Inadomi, Y. *et. al.* (2015), "Analyzing and Mitigating the Impact of Manufacturing Variability in Power-Constrained Supercomputing", in the *International Conference for High Performance Computing, Networking, Storage and Analysis (SC'15).*

Samra, S. *et. al.* (2015), "A Linear Time and Space Algorithm for Optimal Traffic-Signal Duration at an Intersection", *IEEE Transactions on Intelligent Transportation Systems,* vol. 16, no. 1, pp. 387-395.

Sen, S. and Head, K. L. (1997), "Controlled Optimization of Phases at an Intersection," *Transportation Science*, vol. 31, no. 1, pp. 5-17.

Shimaoka, M. *et. al.* (2015), "Coarse Grain Task Parallelization of Earthquake Simulator GMS Using OSCAR Compiler on Various cc-NUMA Servers", in *The 25th International Workshop on Languages and Compilers for Parallel Computing.*

Wada Y. (2018), "Toward Software-Hardware Cooperative Systems for Energy Efficiency with Virtualization Platforms", in *Exhibition, The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC18).*

Wada, Y. (2017), "PomPP Library and Tools". GitHub. Available: https://github.com/pompp/pompp_tools

Wada, Y. *et. al.* (2011), "A Parallelizing Compiler Cooperative Heterogeneous Multicore Processor Architecture", *Transactions on High-Performance Embedded Architectures and Compilers IV*, pp. 215-233.

Wada, Y. *et. al.* (2015), "Energy-aware Task Scheduling for a Manycore Processor with Coarse Grain Voltage Domains", *IPSJ Transactions on Advanced Computing Systems [in Japanese]*, vol. 8, no. 1, pp. 34-50.

Wada, Y. *et. al.* (2018), "A Power Management Framework with Simple DSL for Automatic Power-Performance Optimization on Power-Constrained HPC Systems", *Springer International Publishing.*