Trend of Software R&D for Numerical Simulation — Hardware for parallel and distributed computing and software automatic tuning —

TAKAO FURUKAWA
Promoted Fields Unit

MINORU NOMURA
Information and Communications Unit

1 Introduction

Computer performance has been improved by increase of CPU (central processing unit) clock frequency based on miniaturization of semiconductor manufacturing process, however this reaches the upper limit caused by increase of heat generation, power consumption and leakage current since around 2005. To overcome these problems, in recent years, improvement of computer performance by parallel processing using multi-core processors which have multiple cores executing general operations at lower clock frequency has become a central issue.[8] Since the year 2000, various processors have come on the market, for example multi-core CPUs, GPU (Graphics Processing Units) which enable fast numerical computations, and heterogeneous processors which combine an ordinary core for general oprerations with special cores for numerical operations. While such high performance processors based on parallel computing have come, ironically, development of numerical simulation software with high execution efficiency has been extremely difficult.

In order to make full use of parallel hardware, we must select suitable calculation algorithms for the hardware architecture, and adjust programs to raise execution efficiency. This is called software tuning, which is an essential element for software development in high performance computing. There are technical difficulties as well as tremendous work in manual software tuning for extremely complicated hardware architecture in recent years. Furthermore, developed software must be rewritten and tuned in accordance with frequently updated hardware architecture to keep existing software resources. This increases software maintenance costs. It is a serious obstacle of efficient progress of R&D in numerical simulation to update software for the latest hardware architecture.

This paper first describes trends in hardware architecture and software applications, next provides an overview of layer structure in parallel and distributed software for high performance numerical simulation. We then discuss automatic tuning, which plays an important role to develop software for high performance computing. Finally, we introduce new trends in research organizations promoting fundamental software technology.

2 Trends in Hardware Architecture and Software Applications

2-1 Diversifying Commodity Processors and their Trends

Commodity processors (also called microprocessors, MPUs) are mass produced low cost processors for PCs, servers and game consoles. In recent years, high performance computers for simulation in science and technology are also increasingly adopting parallel and distributed systems using commodity processors.

Figure 1(a) to (d) show typical architectures of commodity processors. (a) Single core CPU and (b) Multi-core CPU designed for general operations that have a large instruction set enabling complicated processing is comprised of a core1 processing unit with memory storing data. For example, it has acceleration functions such as a special instruction for sequential data multiplication and out of order execution which invokes independent processes for fast overall processing. The bottom half of Figure

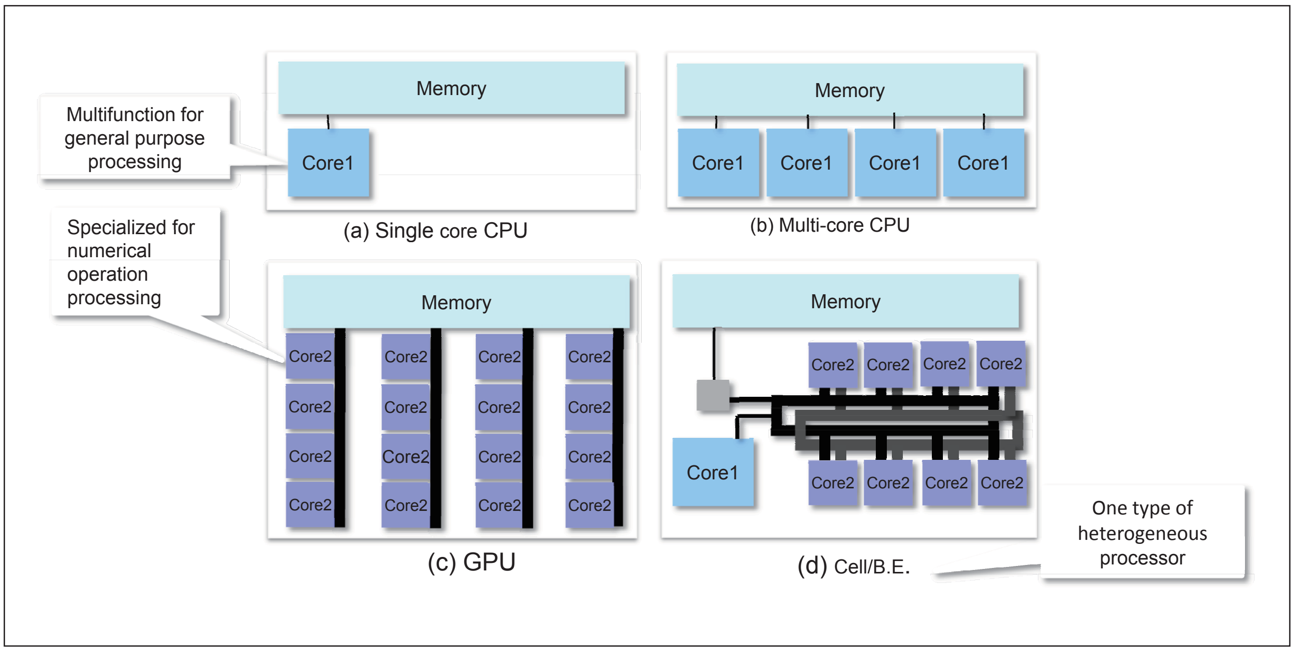
Memory
Multifunction for general purpose processing
Core1
(a) Single core CPU
Memory
Core1
Core1
Core1
Core1
(b) Multi-core CPU
Specialized for numerical operation processing
Memory
Core2
(c) GPU
Memory
Core1
Core2
(d) Cell/B.E.
One type of heterogeneous processor

Figure 1 : Architectures of Commodity Processors

Prepared by the STFC based on Reference[9,20,26]

1 shows (c) GPU and (d) Cell/B.E. (Broadband Engine)™, which are mainly composed of core2 (processing units designed for specific numerical operations). Initially, (c) GPU and (d) Cell/B.E. were developed with the goal of reducing CPU load by isolating numerical operations: dot and cross products which are widely used in 3D computer graphics, and audio and visual data processing that handles compressed data encoding and decoding. In recent years, numerical simulations using (c) GPU and (d) Cell/B.E. as cores[NOTE 1] have been popular in which their high performance is widely noticed.

Cell/B.E.[NOTE 2] in Figure 1(d), which is one type of heterogeneous processor, gives the following improvements compared to conventional CPUs.[9, 28]

● Improved data transfer efficiency in processor

From the aspect of enhancing processing speed, there are bottlenecks in the time required for reading or writing data in memory, and data transfer time among cores. In order to achieve high speed data transfer, it uses dual high speed ring networks connecting each core2 for fast numerical operations.

● Efficient processing and power savings using a general processing core and numerical operation cores

The combination with core1 and core2 improves execution efficiency and reduces power consumption by allotting operations according to their content. A core1 executes general operations and multiple core2s mainly execute numerical operations. The larger number of transistors in a CPU not only increases power consumption, but also increases the power used in chip cooling. Thus instead of increasing the number of core1 which executes various complicated processing, increasing the number of core2 which executes numerical operations, thereby reduces the total number of transistors on the chip.

Figure 2 shows changes in hardware architectures and their relationships with high performance software development. A CPU which controls operation processing as the heart of a computer has been realized with a single core for a long time. Then

[NOTE 1]
Floating point data used in GPUs were extended from single precision to double precision, and were also compatible with the IEEE754 floating point operation standard. Consequently it makes easier to use GPUs in numerical simulations which require calculation precision.
[NOTE 2]
IBM's PowerXCell™ is a product which accelerated the double precision floating point operations of Cell/B.E.

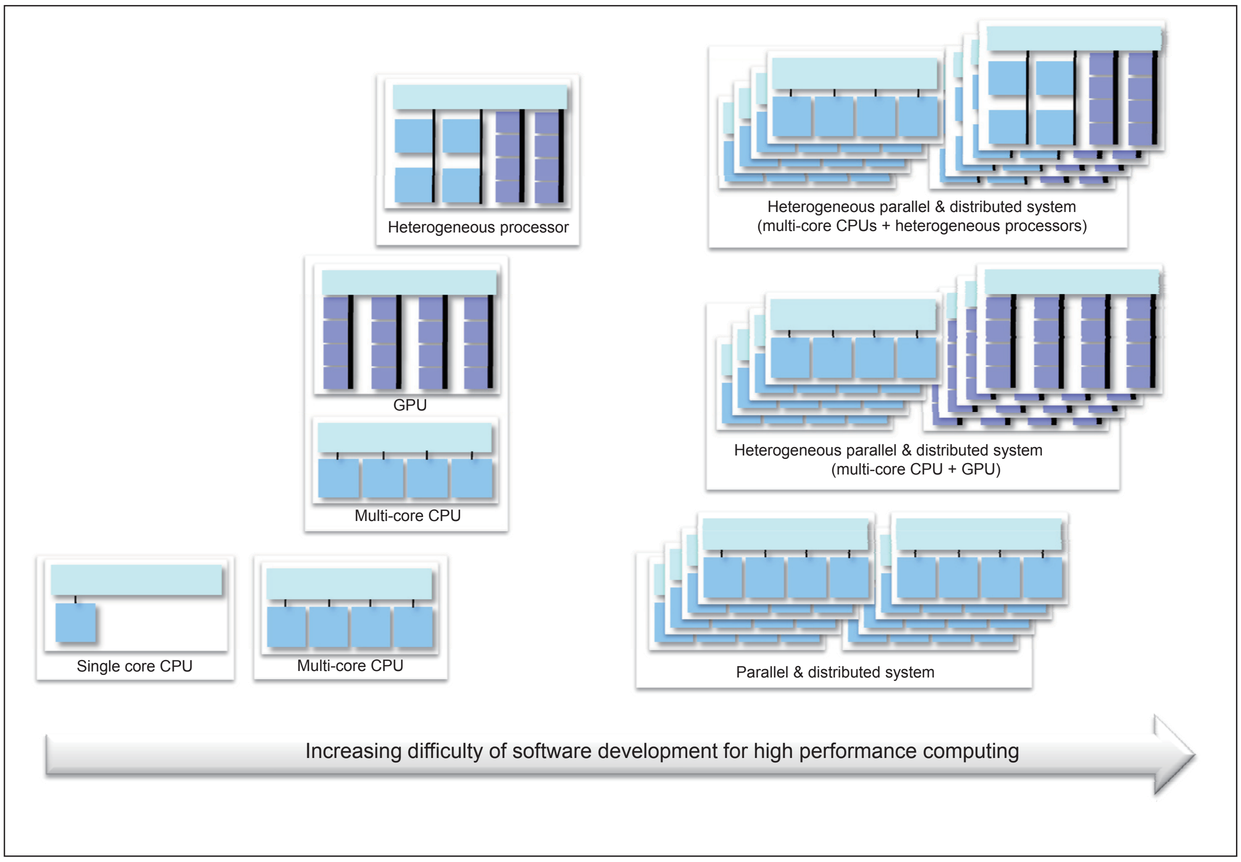
Heterogeneous processor
Heterogeneous parallel & distributed system (multi-core CPUs + heterogeneous processors)
GPU
Multi-core CPU
Heterogeneous parallel & distributed system (multi-core CPU + GPU)
Single core CPU
Multi-core CPU
Parallel & distributed system
Increasing difficulty of software development for high performance computing

Figure 2 : Hardware Architecture Changes and Increasing Difficulty of Software Development for High Performance Computing

Prepared by the STFC based on Reference[9,20,26]

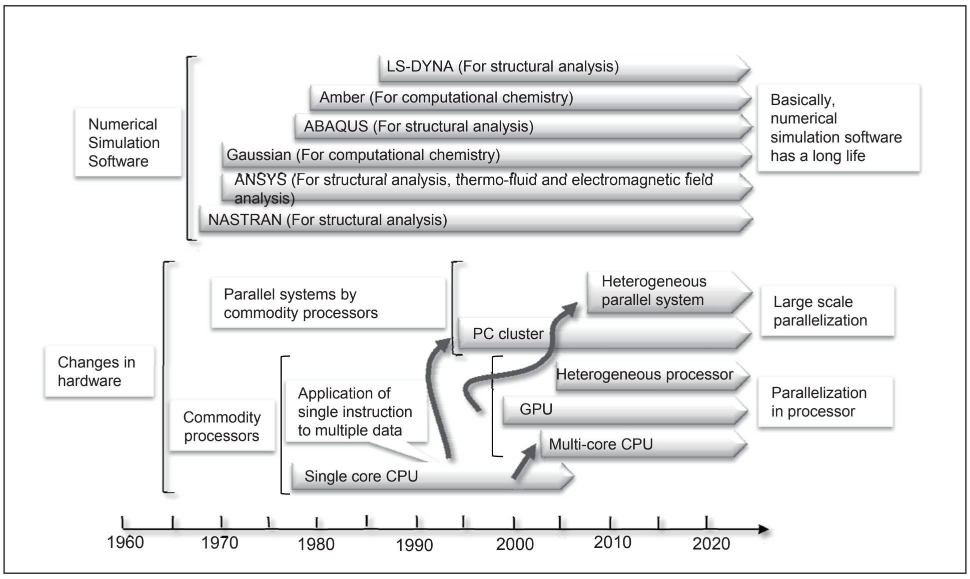
Numerical Simulation Software
LS-DYNA (For structural analysis)
Amber (For computational chemistry)
ABAQUS (For structural analysis)
Gaussian (For computational chemistry)
ANSYS (For structural analysis, thermo-fluid and electromagnetic field analysis)
NASTRAN (For structural analysis)
Basically, numerical simulation software has a long life
Changes in hardware
Parallel systems by commodity processors
Heterogeneous parallel system
PC cluster
Large scale parallelization
Commodity processors
Application of single instruction to multiple data
Heterogeneous processor
GPU
Multi-core CPU
Parallelization in processor
Single core CPU
1960
1970
1980
1990
2000
2010
2020

Figure 3 : Changes in Numerical Simulation Software and Hardware

Prepared by the STFC based on Reference[11-16]

multi-core CPUs appeared in the market. Furthemore, various architectures which use GPUs and multi-core CPUs together came to be adopted. Heterogeneous processors comprises of different cores on one chip. There is a steady move towards heterogeneous parallel and distributed systems which mix these architectures. This trend that the various hardware architectures mentioned above are combined will continue.

Heterogeneous processors would be widely used from the trend appeared in IBM's Cell/B.E.™, AMD's StreamComputing,[21] and Intel's Larrabee.[23] However, data transfer between CPU-GPU tends to be

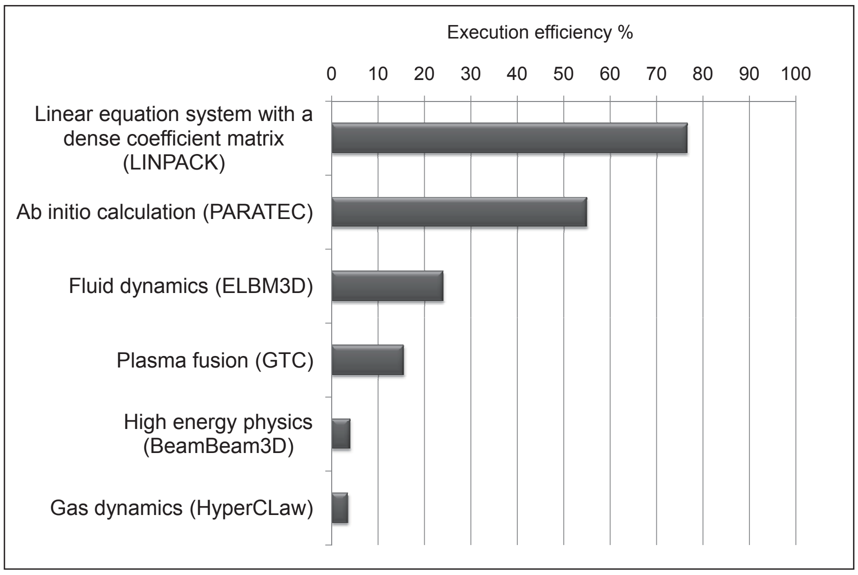

Figure 4 : Execution efficiency of Numerical Simulation Software in a Parallel and Distributed System (Results of calculation by a parallel and distributed system using 512 AMD Opteron processors)

Prepared by the STFC based on Reference[5]

a bottleneck in current architectures. In the short term, a part of numerical simulations by heterogeneous parallel and distributed systems, which combine CPUs and fast numerical calculation GPUs, or heterogeneous processors, are cost effective compared to the cases of homogenous parallel and distributed systems constructed by the identical CPUs.

Software development targetting a heterogeneous parallel and distributed system for high performance computing is extremely difficult compared with usual development targeting a single core CPU. In a parallel and distributed system, execution efficiency is more affected by processes such as data allocation and integration among processors, and data transfer, etc. Thus these factors must be considerd in the software development for high performance computing. Especially for parallel and distributed systems using heterogeneous processors which combine different functions, it makes the software development even more difficult to get sufficient performance, because each core's process must efficiently work in closer cooperation.

2-2 Lifetimes of Numerical Simulation Software

Numerical simulation software tends to have a longer lifetime than for hardware. Figure 3 shows changes in typical numerical simulation software and hardware, especially commodity processors. There is leading numerical simulation software used even today with about 40 years of history. On the other hand, commodity processor architecture has been frequently changed in a short period of time (this is referred to here as a short lifetime for hardware). In order to extend the lifetime of numerical simulation software, in other words, in order to continue using the same software even on hardware with a changed architecture, software must be rewritten each time to suit the novel hardware. For example, leading numerical simulation softwares for structural analysis and computational chemistry appeared in the late 1960s, and these have been used to the present time with function extensions.

2-3 Execution efficiency of Numerical Simulation Software in Parallel and Distributed Systems

In general, numerical simulation software results very low execution efficiency in parallel and distributed systems. Execution efficiency is the sustained performance as a ratio of theoretical performance as 100%. Figure 4 shows the relationship between numerical simulation software and its execution efficiency reported by Oliker et.al.[5] The evaluation covered the simulations shown below which were selected from various science and technology fields. The names of numerical simulation software used to investigate execution efficiency are written in parentheses.

•Linear equation system with a dense coefficient matrix (LINPACK)
•Ab initio calculation (PARATEC)

•Fluid dynamics (ELBM3D)
•Plasma fusion (GTC)
•High energy physics (BeamBeam3D)
•Gas dynamics (HyperCFlow)

The execution efficiencies shown in Figure 4 are obtained in a parallel and distributed system using 512 AMD Opteron processors. Execution efficiency was over 70% for LINPACK, a benchmark program used for comparing high performance computers. However, we recognize from Figure 4 except for PARATEC,[NOTE 3] numerical simulation software has less than 25% execution efficiency. Even if parallel processing is introduced, numerical simulation software results low execution efficiency. If the software is developed for sequential processing, source codes which can not be parallelized remain for the sake of dependency of processes. While parallelization is applicable, data transfer delay and load balance among processors make it even more difficult to improve execution efficiency in a parallel and distributed system. Therefore progress in numerical simulation software does not sufficiently catch up with hardware performance improvements.

3 Components of Numerical Simulation and Software Fundamental Technology

Numerical simulation is divided into many components, and these are arranged in the five layers shown in Figure 5: Theory, mathematical model, algorithms, software, hardware.[17]

On the left side of Figure 5 shows usual components, and on the right are new components which should be considered. Here, software fundamental technology is a set of common components classified in the software layer which are used in various numerical simulations, such as functions linking with hardware.

As shown on the left side under "Usual Components in Numerical Simulation", various components at each layer must be considered for developing numerical simulation software. In the case of software for sequential processing, there was no need to make a program enabling complicated parallel and distributed processing. However, if there is a need for numerical simulation software which runs on a parallel and distributed system, mathematical models and theories as well as algorithms should be designed in consideration of hardware architecture. In the past software development, each layer was relatively independent, but with progress in the use of parallel and distributed systems, each layer has been closely related.

On the right side of the figure, some "New Components for Software and Hardware which Should be Considered" in the hardware layer contains GPUs, heterogeneous processors, and the heterogeneous parallel distributed systems which use these processors. In the software layer, there are Software Development Kits (SDKs) for above parallel processors such as CUDA,[20] MARS,[24] and the standard OpenCL.[25][NOTE 4] Moreover, combination of MPI/OpenMP, and grid computing middleware are added to parallel and distributed processing frameworks as new components.

These components should be considered in software development for high performance numerical simulation to make full use of novel hardware. Due to this situation, software automatic tuning related technology plays more important roles as a fundamental technology in software development for high performance computing. These are discussed in detail below.

[NOTE 3]
PARATEC shows an exceptionally high execution efficiency of about 55%, because Fast Fourier Transform (FFT) accounts for the majority of the calculations which can be accelerated by parallel processing.

[NOTE 4]
Specification for an Application Program Interface (API) created by the Khronos Group, which can provide integrated handling of multi-core CPUs, GPUs and Cell/B.E. However, with OpenCL 1.0,we must write specific programs for each architecture to enahance performace.

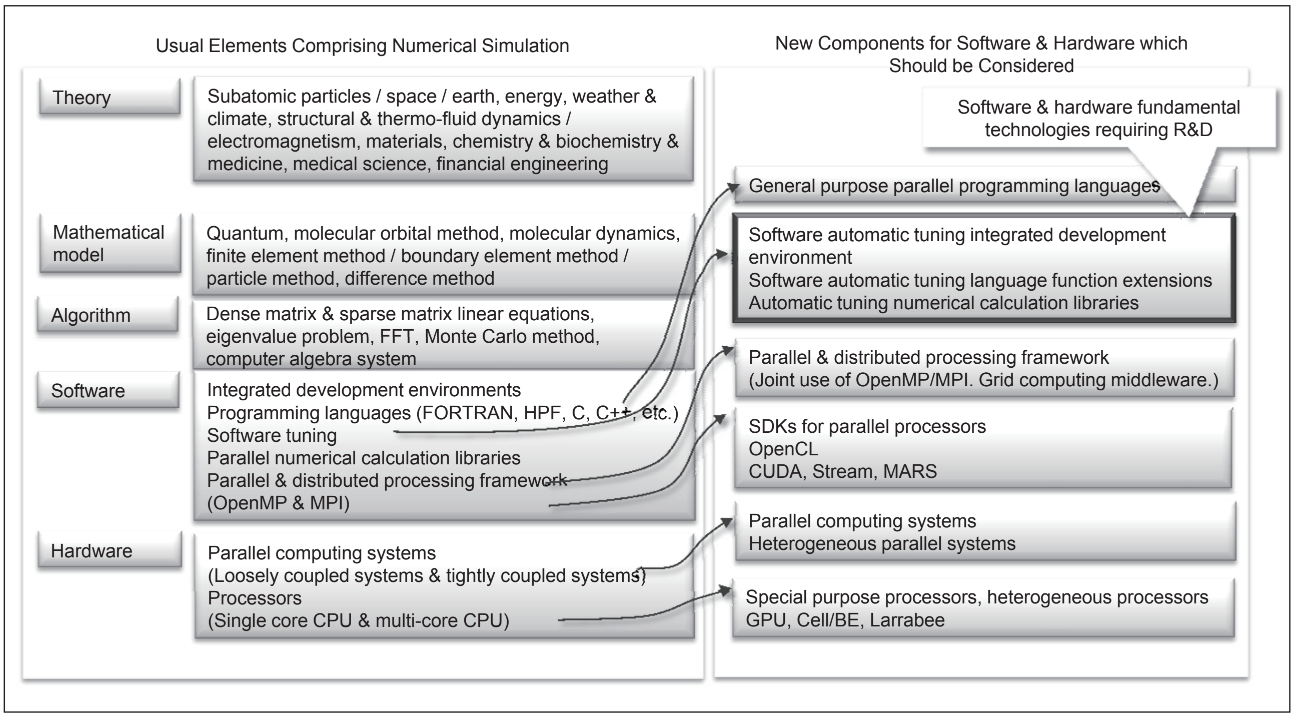
Usual Elements Comprising Numerical Simulation
Theory
Subatomic particles / space / earth, energy, weather & climate, structural & thermo-fluid dynamics / electromagnetism, materials, chemistry & biochemistry & medicine, medical science, financial engineering
Mathematical model
Quantum, molecular orbital method, molecular dynamics, finite element method / boundary element method / particle method, difference method
Algorithm
Dense matrix & sparse matrix linear equations, eigenvalue problem, FFT, Monte Carlo method, computer algebra system
Software
Integrated development environments
Programming languages (FORTRAN, HPF, C, C++, etc.)
Software tuning
Parallel numerical calculation libraries
Parallel & distributed processing framework (OpenMP & MPI)
Hardware
Parallel computing systems (Loosely coupled systems & tightly coupled systems)
Processors (Single core CPU & multi-core CPU)
New Components for Software & Hardware which Should be Considered
Software & hardware fundamental technologies requiring R&D
General purpose parallel programming languages
Software automatic tuning integrated development environment
Software automatic tuning language function extensions
Automatic tuning numerical calculation libraries
Parallel & distributed processing framework (Joint use of OpenMP/MPI. Grid computing middleware.)
SDKs for parallel processors
OpenCL
CUDA, Stream, MARS
Parallel computing systems
Heterogeneous parallel systems
Special purpose processors, heterogeneous processors
GPU, Cell/BE, Larrabee

Figure 5 : Components of Numerical Simulation

Prepared by the STFC based on Reference[9,17-25]

4 Software Automatic Tuning

4-1 Software automatic tuning in Numerical Simulation

Software tuning is a process to adjust software in order to make full use of hardware performance. Software automatic tuning in numerical simulation automatically improves execution efficiency without time-consuming manual tuning by adjusting software to suit hardware.

Figure 6 explains an outline of software automatic tuning with an example. Let's consider the case of solving the fundamental equation described by a partial differential equation using the finite element method. In this numerical simulation, the linear equation system is solved. Here, the items shown in the bottom of Figure 6 represent factors affecting calculation speed and calculation accuracy, which are called tuning conditions. Tuning conditions are divided into simulation models, numerical properties of linear equations, solution algorithms, quality of calculation program, and the computer system. These factors are subdivided into several factors which further affect each condition. Arrowed lines shown in the figure denote the subdivided factors.

A basic software automatic tuning process consists of the following 5 processes.

(1) Experiment: Set tuning conditions and execute the software

(2) Measurement: Obtain measurement items from experimental results, and calculate the evaluation function

(3) Analysis: Estimate the performance model from the measurement items and evaluation function

(4) Learnning: Automatically update the tuning conditions

(5) Decision: Determine optimal tuning conditions

This process improves performance while repeating steps (1) to (4) with changing the tuning conditions, then finally obtains the optimal tuning conditions in step (5). Among these tuning conditions, some items can be automatically determined by usual compiler optimization technology. But the compiler optimization technology can not handle algorithm selection, parameter adjustment and so on. Consequently, we define an evaluation function reflecting calculation speed of numerical simulation, then solving optimization problems, automatic tuning is achived which selects optimal algorithms and adjusts parameters.

Next, we explain the effects of software tuning and the necessity of automatic tuning, taking the example of a size 400x400 matrix multiplication computation. The histogram in Figure 7 shows the distribution of calculation speed obtained from all 16,129 tuning conditions in a 400×400 matrix multiplication, in which the number of results with the same calculation speed is counted. In this case, the highest calculation speed is 1459 MFLOPS, and there are peaks at 1300, 1175 and 1100 MFLOPS in the histogram whose height indicates the number of tuning conditions with the same speed. This suggests that the optimal solution is not easily obtained, even if we define the evaluation function using calculation speed. Tuning conditions greatly vary the calculation speed, therefore there is a small peak near 600 MFLOPS, which is under half of the maximum calculation speed. We can obtain the optimal solution if we investigate the whole search space, but this results in large tuning calculation cost. Consequently, we need an automatic tuning method which efficiently finds conditions maximizing performance from a huge number of tuning condition candidates.

Software automatic tuning can be divided into the functions of static automatic tuning, dynamic automatic tuning, and advanced automatic tuning. These functions are used to create numerical calculation libraries and applications with automatic tuning functions. In developing automatic tuning functions, different development environments such as an integrated automatic tuning development environment, language extension etc. are required. Figure 8 shows these relationships.

According to the history of automatic tuning research and development, software automatic tuning was first applied to the parts of numerical calculation libraries depending on hardware. This is now called static automatic tuning. Next, optimization considering property of input data was applied to numerical calculation. This is now called dynamic automatic tuning. Dynamic automatic tuning investigates the matrix size and distribution of nonzero elements, then automatically determines suitable tuning conditions in addition to the static tuning functions. In order to implement these dynamic automatic tuning functions, it was necessary to extend the specifications of the programming language (programming language extension) to a development environment. In order to archieve advanced automatic tuning, numerical optimization techniques and databases has been used, and the research is now in progress on constructing integrated development environments for automatic tuning software which includes verification of performance improvement. However, the scope of application is still limited to numerical calculation

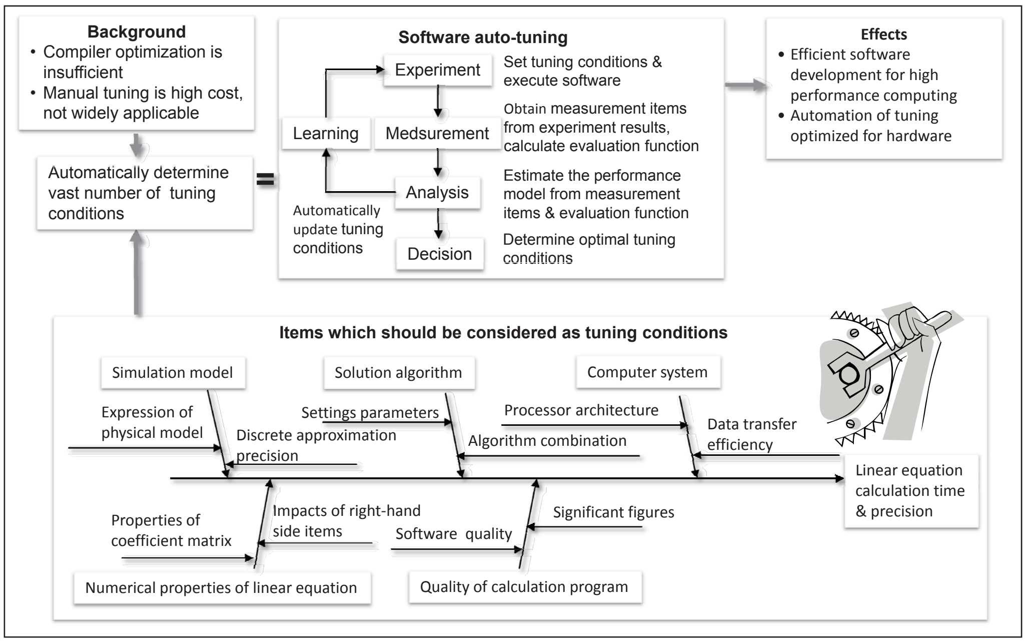
Background
• Compiler optimization is insufficient
• Manual tuning is high cost, not widely applicable
Automatically determine vast number of tuning conditions
=
Software auto-tuning
Experiment
Set tuning conditions & execute software
Learning
Medsurement
Obtain measurement items from experiment results, calculate evaluation function
Analysis
Estimate the performance model from measurement items & evaluation function
Automatically update tuning conditions
Decision
Determine optimal tuning conditions
Effects
• Efficient software development for high performance computing
• Automation of tuning optimized for hardware
Items which should be considered as tuning conditions
Simulation model
Solution algorithm
Computer system
Expression of physical model
Settings parameters
Discrete approximation precision
Processor architecture
Algorithm combination
Data transfer efficiency
Linear equation calculation time & precision
Properties of coefficient matrix
Impacts of right-hand side items
Software quality
Significant figures
Numerical properties of linear equation
Quality of calculation program

Figure 6 : Outline of Software Automatic Tuning

Prepared by the STFC based on References[30-33]

[NOTE 5]
In linear equation systems and eigenvalue problems, sometimes it does not give numerical solution in which iteration depends on the initial parameters and algorithms. Even in these cases, a numerical calculation library with automatic tuning functions is useful, because initial values and parameters are automatically adjusted.

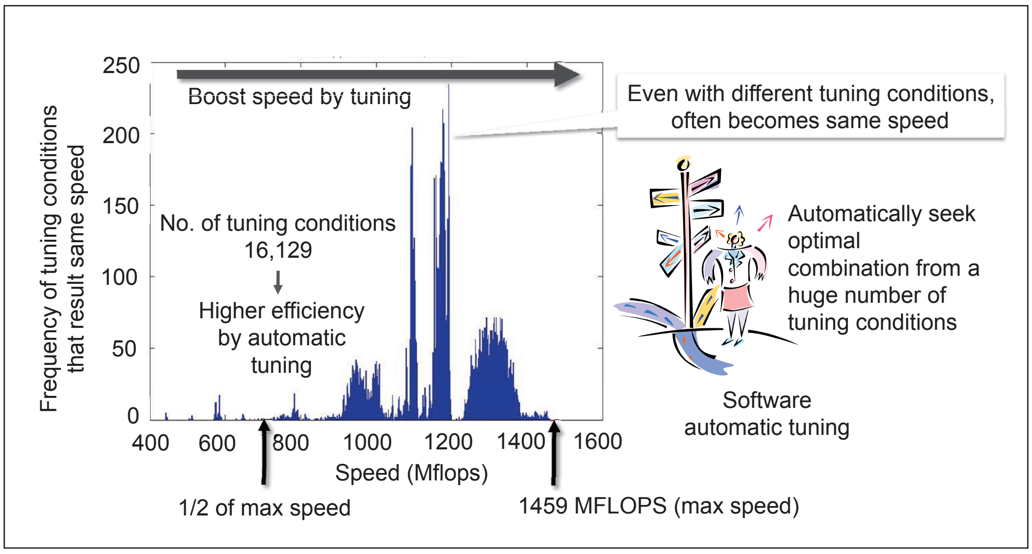

Figure 7 : Relation between Automatic Tuning Conditions and Matrix Calculation Speeds

Prepared by the STFC based on References[40]

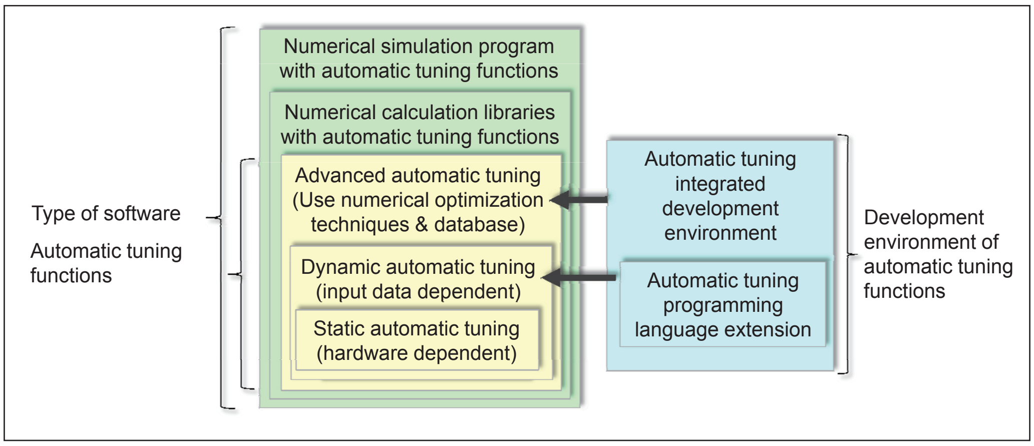
Numerical simulation program with automatic tuning functions
Numerical calculation libraries with automatic tuning functions
Advanced automatic tuning (Use numerical optimization techniques & database)
Dynamic automatic tuning (input data dependent)
Static automatic tuning (hardware dependent)
Type of software
Automatic tuning functions
Automatic tuning integrated development environment
Automatic tuning programming language extension
Development environment of automatic tuning functions

Figure 8 : Software Types and Software Automatic Tuning Functions

Prepared by the STFC based on References[30-33, 40, 42]

libraries, and there are expectations for progress in automatic tuning research which is also applicable to numerical simulations in addition to matrix calculations and signal processing.

4-2 Numerical Calculation Libraries with Automatic Tuning Functions

Numerical simulations frequently use numerical calculation libraries for matrix calculations and signal processing. Operations executed by these numerical calculation libraries often cause bottlenecks decreasing performance. By incorporating automatic tuning functions into numerical calculation libraries for linear equation systems, eigenvalue problems, FFT, etc., the performance of numerical simulations can be improved.

Numerical calculation libraries with automatic tuning functions are also classified into numerical calculation libraries with static automatic tuning functions depending on hardware, and numerical calculation libraries with dynamic automatic tuning functions depending on input data. Characteristics of tuning techniques used in these numerical calculation libraries with automatic tuning functions are shown below.[NOTE 5]

(a) Numerical calculation library with static automatic tuning functions

- During installation, it evaluates the hardware configuration and performance such as the numbers of processor cores and data transfer rate, then adjust the parameters used in libraries to maximize performance.

(b) Numerical calculation library with dynamic automatic tuning functions

- According to the matrix size and distribution of nonzero elements in a sparse matrix, it selects the algorithms and calculation parameters for linear equation systems and eigenvalue problems.

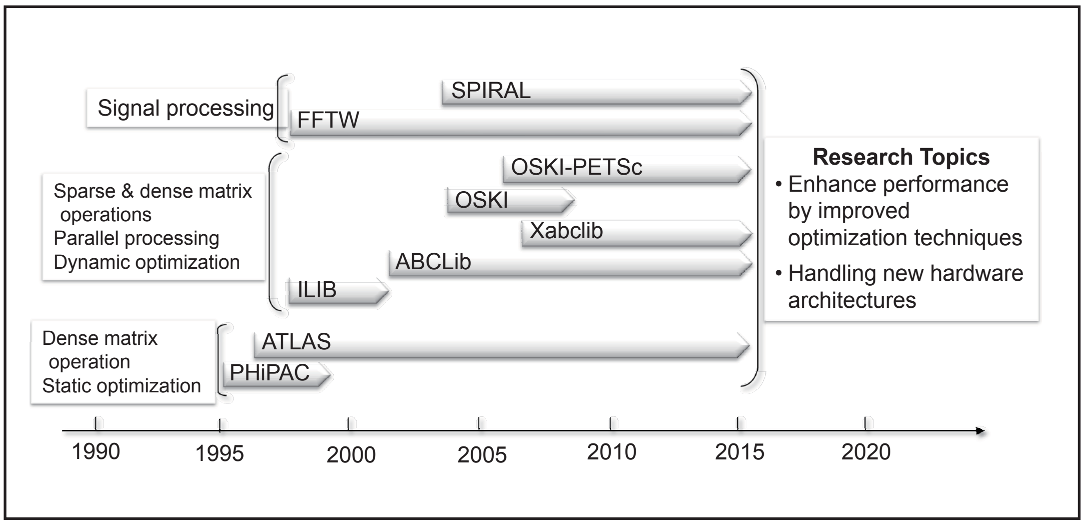

Figure 9 : Trends in Development of Numerical Calculation Libraries with Automatic Tuning Functions

Prepared by the STFC based on References[32, 33, 38, 39, 43]

Table 1 : Numerical Calculation Libraries with Automatic Tuning Functions for which R&D is in progress

Name	Type	Organization	Functions
PHiPAC (Portable High Performance ANSI C)	Dense matrix calculation	University of California, Berkeley, United States	Library automatically accelerating a matrix multiplication loop by adjusting to the hardware architecture. The code generator outputs multiple source codes which implement different tuning conditions, then the fastest codes are automatically selected. Specifically, it improves memory access efficiency by using cache that introduces local variables. Moreover it improves execution efficiency by parallelization for independent codes using loop unlooling and elimination of conditional branches.
ATLAS (Automatically Tuned Linear Algebra Software)	Dense matrix calculation	University of Tennessee, United States	Matrix calculation library including automatic tuning functions that supports parts of BLAS (Basic Linear Algebra Subprograms) and LAPACK (Linear Algebra PACKage). It generates multiple programs with different block sizes which affects matrix calculation performance to adjust hardware. During library installation, it uses a timer to measure execution time and select an optimized program.
FFTW (the Fastest Fourier Transform in the West)	Signal processing	Massachusetts Institute of Technology, United States	High speed Fourier transform library including automatic tuning functions which reduces memory access frequency and amount of calculations. In addition to the automatic tuning during installation, it executes run-time optimization based on the input data size. The parallelized library using MPI and Cell B.E. implementation have been developed so far.
SPIRAL (Software/Hardware Generation for DSP Algorithms)	Signal processing	Carnegie Mellon University, United States	Library containing automatic tuning functions for signal processing, such as FFT, DCT and Wavelet transforms.
OSKI (Optimized Sparse Kernel Interface)	Sparse matrix calculation	University of California, Berkeley Lawrence Livermore National Laboratory, United States	Automatic tuning library for sparse matrices developed by the BeBOP (Berkeley Benchmarking and Optimization Group). This can also handle parallel processing in combination with the PETSc numerical calculation library using MPI and BLAS.
ILIB (Intelligent LIBrary)	Dense and sparse matrix calculation	University of Tokyo	Library for linear equation systems and eigenvalue problems using parallel algorithms. It selects a suitable algorithm based on the distribution of nonzero elements. It can be applied for both sparse and dense coefficient matrices.
ABCLib (Automatically Blocking and Communication-adjustment Library)	Dense and sparse matrix calculation	University of Electro-Communications	Numerical calculation library for parallel distributed systems. It implements automatic blocking which corresponds to cache size parameter tuning, and dynamic selection for optimal communication reflecting the input data.
Xabclib (eXtended ABCLib)	Dense and sparse matrix calculation	University of Tokyo	Parallel automatic tuning library that extends ABCLib using OpenATLib. It supports eigenvalue problems using the LANCZOS method, and linear equation systems using the GMRES method.

Prepared by the STFC based on References[32, 33, 38, 39, 43]

• Applies tuning techniques corresponding to dynamically allocated processors and number of cores.

Figure 9 shows the history of developments of these numerical calculation libraries with automatic tuning functions, and predicted future developments. PHiPAC and ATLAS are libraries which implement static optimization for dense matrix calculations. The following are numerical calculation libraries with dynamic automatic tuning functions for parallel and distributed systems: FFTW and SPIRAL for signal processing, and OSKI, ILIB, ABCLib and Xabclib for matrix calculations. ILIB had been extended to ABCLib, and moreover its numerical algorithm selection during run-time and communications methods tuning were improved in Xabclib.[33] Table 1 summarizes trends in development of numerical calculation libraries which focus on each automatic tuning function. As shown in Table 1, many research institutions tackle researches on numerical calculation libraries with these automatic tuning functions.

In general, future research topics will be performance enhancement by improved optimization methods, and automatic tuning techniques for novel hardware architectures. Regarding heterogeneous parallel and distributed system which is considered as one of typical novel hardware architectures, research is still at the stage of manual tuning.[35,36] Thus progress from manual tuning into automatic tuning is expeceted.

4-3 Programming Language Extension and Integrated Development Environments for Automatic tuning

As described above, in order to achieve dynamic automatic tuning functions, programming language extension is introduced for automatic tuning in existing programming languages. Moreover, for advanced automatic tuning, support tools including performance evaluation and analysis functions are used from integrated development environments. For once we write source code containing automatic tuning functions, we can easily improve performance corresponding to updated hardware such as a large scale parallel and distributed system. Consequently, it results benefits of reduced hardware dependency and improved software portability.

Figure 10 shows the history of development of programming language extensions and integrated development environments for software automatic tuning, and forecasted future development trends.[31,33,40-43] Here, items with a * denote integrated development environments, and the others are programming language extensions. ROSE and Active Harmony are programming language extension tools which add performance measurement functions for automatic tuning into source code written in various programming languages. POET in combination with ROSE, and CHiLL in combination with Active Harmony, are integrated development environments which optimize various parameters of software automatic tuning. ABCLibScript provides a programming language extension with multiple functions for automatic tuning, but

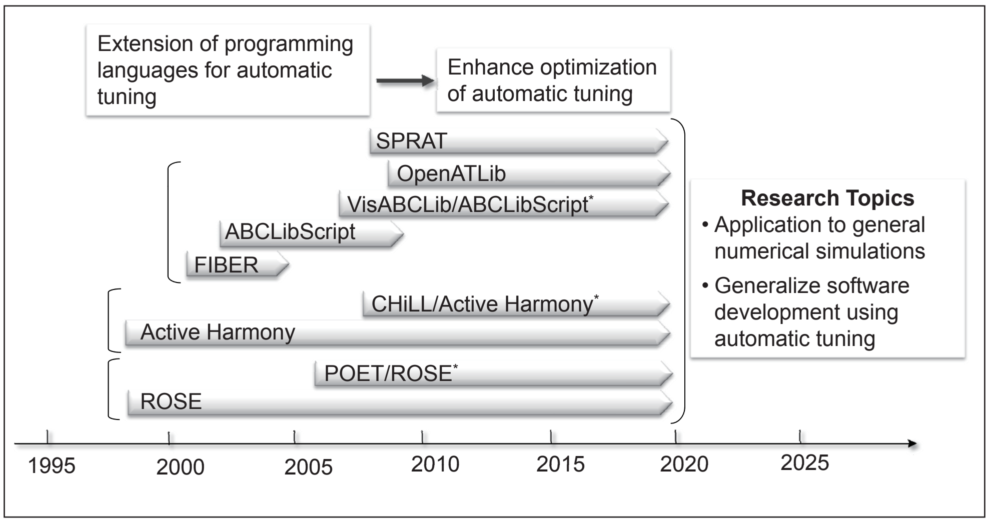
Extension of programming languages for automatic tuning
Enhance optimization of automatic tuning
SPRAT
OpenATLib
VisABCLib/ABCLibScript*
ABCLibScript
FIBER
CHiLL/Active Harmony*
Active Harmony
POET/ROSE*
ROSE
Research Topics
• Application to general numerical simulations
• Generalize software development using automatic tuning
1995
2000
2005
2010
2015
2020
2025

Figure 10 : Trends in Development of Programming Language Extension and Integrated Development Environments for Automatic Tuning

Prepared by the STFC based on References[31,34,41,42,44] applicable programming language is currently limited to FORTRAN, C language extension is now planned. VisABCLib is a software automatic tuning integrated development environment which handles ABCLibScript, with the special feature of advanced functions for visualizing software performance. SPRAT is a programming language extension that generates CUDA source code for GPUs and C++ source codes for multi-core CPUs, to achieve higher performance by automatically switching calculations between CPUs and GPUs corresponding to hardware performance.[34]

High performance numerical calculation libraries, such as matrix calculation and signal processing, have been developed using programming language extensions and integrated development environments for automatic tuning, but different issues remain in automatic tuning for general numerical simulations. For example, in simulations which describe interaction between fluids and rigid (or elastic) bodies and interaction in molecular dynamics, etc., conditional branches often appear in loop iteration, which cannot be handled sufficiently by automatic tuning techniques for high performance numerical calculation libraries. Consequently, software development based on automatic tuning for these general simulations will be a research topic.

As shown in Table 2, many research institutions tackle research on both programming language extensions and integrated development environments.

5 New Moves towards Software Automatic Tuning Applications

The history of software automatic tuning began from research on performance enhancement in numerical calculation libraries. Therefore it has focused on performance enhancement for numerical calculation libraries such as linear equation systems and eigenvalue problems, and applications to general numerical simulation software seem to be inactive so far. A problem is insufficient cooperation between

Table 2 : Characteristics of Automatic Tuning Programming Language Extensions and Integrated Development Environments on R&D in progress

Name	Organization	Functions
ROSE	Lawrence Livermore National Laboratory,United States	Programming language extension in order to convert source code written in FOTRAN, C, C++, OpenMP, and UPC. By using ROSE, it allows to implement automatic tuning for source codes written in various programming languages.
POET (Parameterized Optimizations for Empirical Tuning)	University of Texas at Austin, United States	Integrated development environment which applies optimization techniques such as full search, simplex method, simulated annealing, genetic algorithms, etc. to parameter adjustment by automatic tuning. Used in combination with ROSE.
Active Harmony	University of Maryland, United States	Programming language extension for automatic tuning for run-time software performance measurement and feedback.
CHiLL	University of Southern California, United States	Integrated development environment for automatic tuning with optimization techniques which adjust parameters by changing region correspoinding to simplex in search space. Used in combination with Active Harmony.
FIBER (Framework of Install-time, Before Execute-time, and Run-time Auto-tuning)	University of Tokyo	Development framework for numerical calculation libraries with automatic tuning during installation, before execution and run-time. It supports the following automatic tuning techniques. •During installation: Optimization of library to match target hardware. •Before execution: Optimization depending on problems such as matrix size •Run-time: Optimization considering distribution of nonzero elements in a sparse matrix, optimization of communication methods
ABCLibScript	University of Electro-Communications	Programming language extension for automatic tuning specialized for numerical simulation. It automatically executes 3 tuning techniques: block width adjustment, algorithm selection, loop unrolling adjustment. ABCLibCodeGen generates automatic tuning programs from source codes written in FORTRAN with additional ABCLibScript description. Then it repeats performance sampling, thereby an automatically tuned program can be obtained.
VizABCLib	University of Electro-Communications	Programming support tool using ABCLibScript that has the following functions. •Interactive display for automatic tuning code •Generate a log in automatic tuning process •Compare predicted performance and measured performance •Systematic performance evaluation •Database of information required in automatic tuning: calculation scheme, algorithms, etc.
SPRAT	Tohoku University	Compiler generating both C++ program for CPU and CUDA program for GPU from source code written in a special programming language which does not depend on the CPU and GPU.

Prepared by the STFC based on References[31-34, 41-44] computer scientists who pursue software automatic tuning and researchers in various fields using numerical simulation. As an example of promoting such cooperation, in the Scientific Discovery through Advanced Computing (SciDAC-2) program by the Office of Science of the U.S. Department of Energy, there is the Performance Engineering Research Institute (PERI)[45] project which focuses on software performance engineering including automatic tuning. Their activities are described below.

The background to launch PERI is as follows. In the SciDAC-1 program started in 2001, PERI's predecessor the Performance Evaluation Research Center (PERC) project achieved research works on benchmarking, analysis, performance modeling and optimization of numerical simulation programs for high performance computing, and their application to climate prediction models, plasma turbulence and accelerator simulations. From these research works and their applications, it makes the issues described in Section 2 clear. There are obstacles to smooth progress in research in which software must be rewritten for novel hardware because hardware lifetime is shorter than software lifetime. Problems between both types of researchers were also pointed out: researchers using numerical simulation do not provide information on portability of source code, on the other hand, computer scientist are not interested in tools to port developed software. While considering these problems, the PERI project was begun as a successor project to PERC.

Currently, both the SciDAC program and INCITE (Innovative and Novel Computational Impact on Theory and Experiment) program are included in the Advanced Scientific Computing Research (ASCR) program by the Office of Science, Department of Energy in the U.S. SciDAC-2 is a program which focuses on software fundamental technology in high performance computing. On the other hand, INCITE program mainly provides high performance hardware and computing resources for numerical simulations.

5-1 SciDAC-2 Research on Fundamental Software for Numerical Simulation

As shown in Figure 11, SciDAC-2 is broadly grouped into 3 organizations in charge of Research on fundamental technologies, Application of fundamental technologies, and Scientific applications. Table 3 shows each project and research topics of SciDAC-2 in 2009. It is a notable aspect that Outreach Center in SciDAC-2 which acts as a support organization

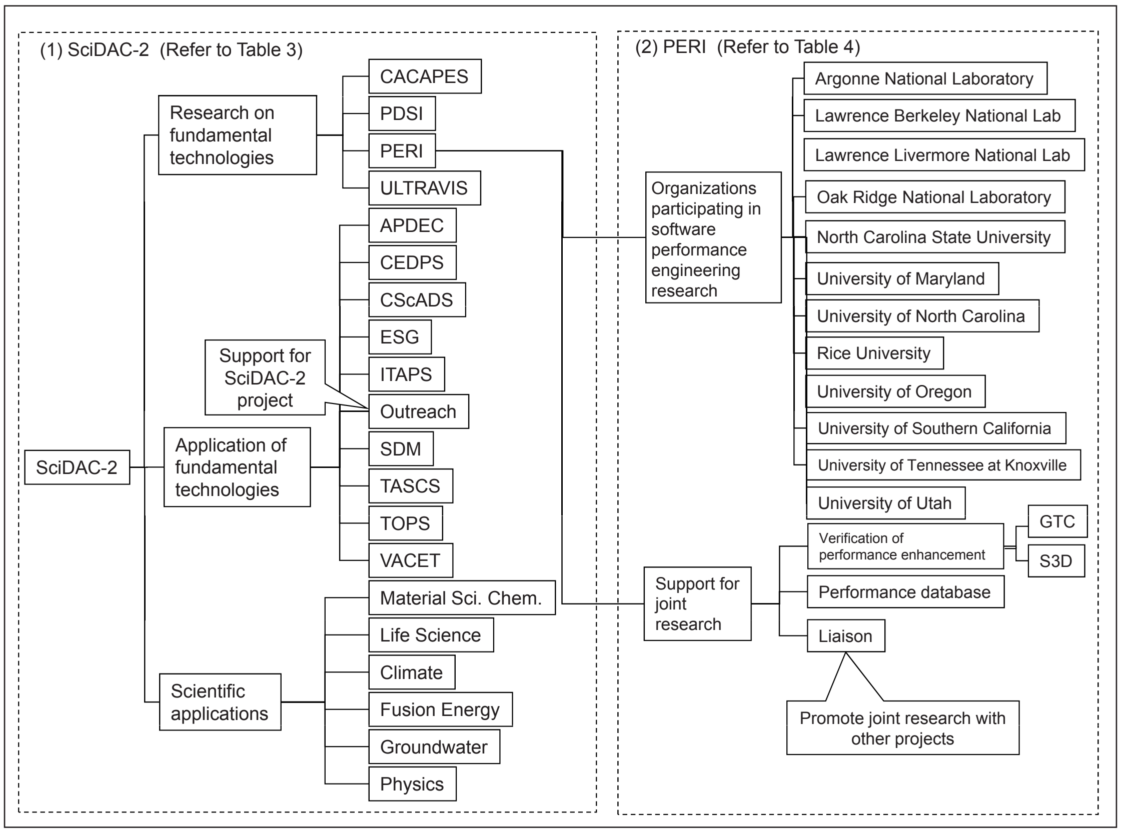
(1) SciDAC-2 (Refer to Table 3)
SciDAC-2
Research on fundamental technologies
CACAPES
PDSI
PERI
ULTRAVIS
Application of fundamental technologies
APDEC
CEDPS
CScADS
ESG
ITAPS
Outreach
SDM
TASCS
TOPS
VACET
Support for SciDAC-2 project
Scientific applications
Material Sci. Chem.
Life Science
Climate
Fusion Energy
Groundwater
Physics
(2) PERI (Refer to Table 4)
Organizations participating in software performance engineering research
Argonne National Laboratory
Lawrence Berkeley National Lab
Lawrence Livermore National Lab
Oak Ridge National Laboratory
North Carolina State University
University of Maryland
University of North Carolina
Rice University
University of Oregon
University of Southern California
University of Tennessee at Knoxville
University of Utah
Support for joint research
Verification of performance enhancement
GTC
S3D
Performance database
Liaison
Promote joint research with other projects

Figure 11 : Components of the SciDAC-2 Program including PERI Project
Prepared by the STFC based on References[45]

for disclosure of research results, etc. In addition to publishing project information, this Outreach Center plays important roles, such as user support and training, and active promotion for building up closer connections between projects.

5-2 Research on Software Performance Engineering in PERI Program

(1) Roles of PERI

As part of the SciDAC-2 program, a goal of PERI is to provide software development technology for high performance computing to numerical simulation research in other projects. PERI is in charge of the following R&D.

•Performance modeling for numerical simulations
•Accurately predict the execution speed which can be obtained from developed software.
•Software automatic tuning R&D
•Set highly difficult long term research targets for reducing the researchers' programming burdens.
•Application R&D
•Apply research works in PERI to numerical simulations in other R&D projects in SciDAC-2.

(2) PERI's Organization and Operation

Looking at PERI's organization and operation, it

Table 3 : Organizations and Topics in SciDAC-2 Program

	Abbreviation of organization	Organization name	Topics
Research on fundamental technologies	CACAPES	Combinatorial Scientific Computing and Petascale Simulations Institute	Load balancing for parallel computing, automatic differentiation, sparse matrix calculation
	PDSI	Petascale Data Storage Institute	Specifications, standards, algorithms, and performance measurement tool development focused on data storage
	PERI	Performance Engineering Research Institute	Software performance engineering: software performance modeling, performance prediction, software automatic tuning, applications
	ULTRAVIS	Institute for Ultra-Scale Visualization	Development of visualization tools for extracting potential information from huge data sets
Application of fundamental technology	APDEC	Applied Partial Differential Equations Center	Algorithms and software framework for partial differential equations
	CEDPS	Center for Enabling Distributed Petascale Science	High reliable and high performance data transfer mechanism and resource allocation and virtualized environment on grid
	CScADS	Center for Scalable Application Development Software	Petascale computing platform, communication library, mathematical library, open source compiler
	ESG	Earth System Grid Center for Enabled Technologies	Data creation for next generation simulation integrating the atmosphere, sea and land for climate and weather forecasts
	ITAPS	Interoperable Technologies for Advanced Petascale Simulations Center	Mutual use of SciDAC applications, and compatible data manipulation tools for mesh, geometry, etc.
	Outreach	Outreach Center	Share information among projects, support services, training, transfer of SciDAC research results to new organizations
	SDM	Scientific Data Management Center	Science and technology computing workflow automation, data mining, data analysis, efficient access to storage
	TASCS	Center for Technology for Advanced Scientific Component Software	Develop component software for parallel simulations, hardware and software to improve quality, robustness, dynamic adaptability, and usability.
	TOPS	Towards Optimal Petascale Simulations	R&D to solve bottlenecks of scalable solvers and applications
	VACET	Visualization and Analytics Center for Enabling Technologies	R&D on data visualization and analysis software
Scientific applications	Material science, Chemistry		Petascale computational chemistry and material modeling, quantum simulation of nanostructures, crack analysis under stress, chemical reactions and interactions in fluids
	Life science	Impacts of microbes and bacteria on environment. Energy generation.	Hydrogen generation, bioethanol and energy generation from microbes
	Climate	Climate and weather	Successive, dynamic and adaptive grid computing for physical and chemical models of earth climate, cloud modeling and its validation, improvement of global climate model, and atmosphere model.
	Fusion energy	Alternative clean energy	Turbulence analysis for plasma fusion
	Groundwater		Model for contaminant dispersion by groundwater and geometric, biological and chemical model of underground
	Physics		Subatomic particles, nuclear energy, astrophysics, turbulence analysis of shock waves, open science grid

Prepared by the STFC based on References[45]

Table 4 : Participating Organizations in PERI and Topics

Organization	Topics
Argonne National Laboratory	Service infrastructure software quality enhancement for numerical simulations (in cooperation with TASCS). Software performance databases: interface extension and addition of simple interface for application developers. Definition and component implementation of interface to automate learning in automatic tuning. Infrastructure to share hardware performance database among applications. Performance analysis interface extension using PerfExplorer. Making more robust analysis component prototype based on machine learning (in cooperation with University of Oregon). FLASH application performance evaluations on Argonne National Laboratory's Blue Gene/P and Oak Ridge National Laboratory's Cray XT3.
Lawrence Berkeley National Laboratory	PERI project management. Check progress of PERI and fundamental technology research organizations. Analysis processing of plasma turbulence analysis team. Quantum calculation software tuning of material simulations for new solar cells. Development and testing of automatic tuning functions for applications on multi-core processors.
Lawrence Livermore National Laboratory	Empirical tuning using POET, which is a tool for automatic tuning. Continue survey of software performance evaluation, especially cross-platform models. Coordinate with PERI researchers of other organizations and integrate various performance prediction tools. Application of software automatic tuning and performance prediction tools to SciDAC applications. Generate models showing activity of MPI applications. Extension of MPI tracing mechanism which measures communication patterns. Implementation of functions which measure performance distribution of MPI events. Survey of advanced techniques for ideal tracing including timestamps. Promote activities of performance enhancement verification teams.
Oak Ridge National Laboratory	Continue development of interconnect simulator for network topology and routing settings, which is required by the teams verifying performance enhancement. Comparison of models and simulation results for performance measurement in large scale systems. Improve accuracy of models which identify scaling bottlenecks. Support joint research for applications in climate and weather, fusion, material science, and groundwater. Promote application of PERI's research results. Promote cooperation with projects outside SciDAC.
North Carolina State University	Continue support for joint research of application teams, analyze performance and optimize on Cray XT4 and BlueGene/P. Continue to support communication of application teams with PETSc developers and users, and improve I/O and user routines.
Rice University	Continue joint work with Cray and IBM to solve problems of performance sampling using hardware counter function of OS. Introduce HPCToolkit at the stage after OS problems are fixed. Continue to work on extension of path profiling function of optimization code of HPCToolkit. Continue to work on extension of performance analysis techniques for OpenMP and MPI+OpenMP programs. Coordinate with SciDAC and INCITE application teams.
University of California, San Diego	α testing of network simulator. β release of network simulator. Basic R&D for memory tracing estimation in large scale data and processor systems.
University of Maryland	Integration of automatic tuning framework including an empirical search function. Complete integration of Active Harmony with the ChiLL framework, and start evaluation. Development of PERI-DB search API. Support for performance enhancement verification teams.
University of Oregon	Continue support for performance measurement and analysis of petascale applications. Continue performance measurement and fluid analysis and plasma turbulence analysis applications of performance enhancement verification teams. Continue integration of performance database with PERI-DB group. Use PerfExplorer in data analysis of performance enhancement verification teams.
University of Southern California	Management of entire PERI project. Continue API development for automatic tuning users. Research to determine specifications related to automatic tuning and ChiLL. Coordinate with SciDAC and INCITE application teams. R&D on data copying libraries (cooperate with University of Utah). Continue integrating Active Harmony with ChiLL (joint research with University of Maryland and University of Utah).
University of Tennessee at Knoxville	Continue development of cross-platform performance counter library which supports PERI performance modeling and automatic tuning. Research on empirical search techniques for automatic tuning, and integration with PERI automatic tuning framework. Research on optimization techniques for multi-core architectures, and integration with PERI automatic tuning framework. Cooperate in building database of performance enhancement verification teams.
University of Utah	Drive the PERI automatic tuning groups, make reports with external joint researchers, and coordinate poster presentations and paper publications. Introduce thread mechanisms in automatic tuning compiler technology. Develop of data compiler library (joint research with University of Southern California) Continue to integrate Active Harmony with Chill (cooperate with University of Southern California and University of Maryland). Work to build a stable release of CHiLL (cooperate with University of Southern California).

Prepared by the STFC based on References[46 is noteworthy that they form and operate a highly productive organization based on comprehensive understanding of the components in numerical simulations as shown in Figure 5, focused on R&D in software performance engineering for software fundamental technologies. Specifically, it works to share R&D goals, and to build up a connection between each fundamental research and applied R&D. As a result, PERI seems to be excellent at quickly removing obstacles on the way to practical use.

As shown in Figure 11, PERI contains groups being in charge of software performance engineering, and groups supporting joint research on numerical simulation applications etc. Table 4 shows the topics assigned to 4 national laboratories and 8 universities. In addition to each project's R&D, it is noteworthy that they work on coordination with other projects in PERI, SciDAC-2, INCITE, etc. In supporting organizations for joint research, there are plasma fusion simulation applications (GTC), fluid simulations (S3D), performance database building, and liaison for joint research with other projects. Note that the liaison group members are also members of groups being in charge of software performance engineering.

A meeting of all PERI groups is held each year, and biweekly telephone conferences are held for close coordination. It is also publicly decided that unscheduled meetings are held Monday mornings. Moreover, limited resources are focused on important SciDAC-2 projects, and they take care not to change a research organization for general computer science and mathematics which are unrelated to software performance engineering. In this way, efficient research management is performed, and in order to smoothly apply research results to numerical simulations, Outreach Center supporting SciDAC-2 projects and liaison within PERI play important roles.

6 Issues for Research Promotion in Japan

As described above, there is a steady increase in numerical simulations on parallel distributed systems using commodity processors, but it is extremely difficult to develop software for high performance computing which makes full use of hardware performance. This is why software fundamental technology with automatic tuning technology as the core is playing an important role in software development for high performance computing. Especially for heterogeneous parallel distributed systems, tuning itself is at a research stage before automation, and there is a need to advance research which aims at practical use of automatic tuning.

In Japan, researchers in numerical computing who belong to universities and companies have launched the Automatic Tuning Research Group. They reported a survey of automatic tuning technology[48] and created a specification of Application Program Interface (API) in the OpenATLib automatic tuning library and programming language extension. Also, the Automatic Tuning Research Group has held the International Workshop on Automatic Performance Tuning (iWAPT) since 2006, which is also attracting the attention of overseas researchers.

The research works in Japan at a level similar to in the U.S., but issues remain its promotion. A roadmap from research to practical use in numerical simulations, resource allocation and sharing among researchers are insufficient in Japan, because the promotion organization consists of the researchers themselves. Especially for R&D on software fundamental technologies at universities and research institutes, it may be most efficient to pursue research and applications in parallel with overlook of the related projects, like SciDAC-2 and its PERI in the U.S. We hope to reconsider of research management in research organizations in Japan, in order to efficiently drive fundamental technology research and applied research.

Acknowledgements

We are very grateful for the valuable comments from Professor Kengo Nakajima of the University of Tokyo Information Technology Center, Project Associate Professor Takahiro Katagiri, Shoji Itoh of the Advanced Center for Computing and Communication at RIKEN, Associate Professor Toshio Endo of the Graduate School of Information Science and Engineering at Tokyo Institute of Technology, Program Director Kunihiko Watanabe of the Japan Agency for Marine-Earth Science and Technology, and CEO Satoshi Miki and CTO Yosuke Tamura of Fixstars Corp.

Abbreviations

ABCLib: Automatically Blocking and Communication-adjustment Library
APDEC: Applied Partial Differential Equations Center
ASCR: Advanced Scientific Computing Research
ATLAS: Automatically Tuned Linear Algebra Software
CACAPES: Combinatorial Scientific Computing and Petascale Simulations Institute
CEDPS: Center for Enabling Distributed Petascale Science
CScADS: Center for Scalable Application Development Software
ESG: Earth System Grid Center for Enabled Technologies
FFT: Fast Fourier Transform
FFTW: the Fastest Fourier Transform in the West
FIBER: Framework of Install-time Before Execute-time, and Run-time auto-tuning
GPU: Graphics Processing Unit
GTC: Gyrokinetic Turbulence Code
HPC: High Performance Computing
ILIB: Intelligent Library
INCITE: Innovative and Novel Computational Impact on Theory and Experiment
ITAPS: Interoperable Technologies for Advanced Petascale Simulations Center
iWapt: International Workshop on Automatic Performance Tuning
OSKI: Optimized Sparse Kernel Interface
PARATEC: Parallel Total Energy Code
PDSI: Petascale Data Storage Institute
PHiPAC: Portable High Performance ANSI C
PERC: Performance Evaluation Research Center
PERI: Performance Engineering Research Institute
SciDAC: Scientific Discovery through Advanced Computing
SDK: Software Development Kit
SDM: Scientific Data Management Center
SPIRAL: Software/Hardware Generation for DSP algorithms
TASCS: Center for Technology for Advanced Scientific Component Software
TOPS: Towards Optimal Petascale Simulations
ULTRAVIS: Institute for Ultra-Scale Visualization
VACET: Visualization and Analytics Center for Enabling Technologies
Xabclib: eXtended ABCLib

References

[1] World Technology Evaluation Center, Inc., WTEC Report on International Assessment of Research and Development in Simulation-Based Engineering and Science, (Apr. 2009)

[2] Minoru Nomura, Trends in High-End Computing in United States Government, Science & Technology Trends : Quarterly Review No.16 (2005.07)

[3] Yoshitaka Tateyama, Dissemination of Nanosimulation Techniques to Promote the Development of Nanotechnology, Science & Technology Trends : Quarterly Review No.20 (2006.07)

[4] Minoru Nomura, Petascale Computing Trends in Europe, Science & Technology Trends : Quarterly Review No.27 (2008.04)

[5] Leonid Oliker, Andrew Canning, Jonathan Carter, Costin Iancu, Michael Lijewski, Shoaib Kamil, John Shalf, Hangzhang Shan, Eric Strohmaier, Stephan Ethier, Tom Godate, Scientific Application Performance on Candidate PetaScale Platforms, Proc. IPDSP, (2007)

[6] Fumitoshi Sato, Toshiyuki Hirano, Toshihiko Abe, Noriko Uemura, Naoki Tsunegawa, Yasuyuki Nishimura, Tomomi Yamaguchi, Hidenori Yukawa, Kentaro Ishikawa, Koji Chiba, All-Electron Simulations of Proteins by Density Functional Method, Proceedings of the 28th Japan Society for Simulation Technology Annual Conference, pp.163-166, (2009.06) [Japanese language]

[7] Japan Agency for Marine-Earth Science and Technology, Annual Report of the Earth Simulator Strategic Industrial Use Program, (2008.09) [Japanese language]

[8] Increase in Number of Processors Installed in High Performance Computers, Science & Technology Trends, No.90, (2008.09) [Japanese language]

[9] IBM, Cell Broadband Engine Technology
http://www-03.ibm.com/technology/resources/technology_cell_pdf_CellBrBandEngineWhitePaper.pdf

[10] Richard Walsh, Steve Conway, Earl C. Joseph, Jie Wu, With Its New PowerXCell 8i Product Line, IBM Intends Take Accelerated Processing into the HPC Mainstream, August 2008
http://www-03.ibm.com/technology/resources/technology_cell_IDC_report_on_PowerXCell.pdf

[11] MSC NASTRAN, http://www.mscsoftware.co.jp/products/nastran/

[12] ANSYS Inc., http://www.ansys.com/

[13] The Official Gaussian Website, http://www.gaussian.com/

[14] SIMULIA, http://www.simulia.com/

[15] Amber and NPACI: A Strategic Application Collaboration for Molecular Dynamics
http://www.sdsc.edu/pub/envision/v14.4/sac_amber.html

[16] Livermore Software Technology Group, http://www.lstc.com

[17] Kengo Nakajima, Education Program for "Interdisciplinary Computational Science and Engineering" at the University of Tokyo
http://hss.iic.hokudai.ac.jp/WS07/pdf/Nakajima.pdf

[18] OpenMP, http://openmp.org/wp/

[19] MPICH2, http://www.mcs.anl.gov/research/projects/mpich2/

[20] CUDA Zone, http://www.nvidia.com/object/cuda_home.html#

[21] ATI Stream Software Development Kit
http://developer.amd.com/gpu/ATIStreamSDK/Pages/default.aspx

[22] General-Purpose Computation on Graphics Hardware, http://gpgpu.org/

[23] Aleksey Bader et al., Game Physics Performance on the Larrabee Architecture
http://download.intel.com/technology/architecture-silicon/GamePhysicsOnLarrabee_paper.pdf

[24] MARS: Multicore Application Runtime System
http://ftp.uk.linux.org/pub/linux/Sony-PS3/mars/1.1.4/mars-docs-1.1.4/html/

[25] The Khronos OpenCL Working Group, The OpenCL Specification, Version 1.0, Document Revision 29
http://www.khronos.org/news/press/releases/the_khronos_group_releases_opencl_1.0_specification/

[26] Toshio Endo, Tokyo Institute of Technology, Accelerator Utilization Example in TSUBAME, Journal of Information Processing, Vol.50, No.2, pp.100-106, (2009.02) [Japanese language]

[27] Takayuki Aoki, CFD Applications Fully Accelerated by GPU, Journal of Information Processing, Vol.50, No.2, pp.107-115, (2009.02) [Japanese language]

[28] Akira Tsukamoto, Kinuko Yasuda, Yosuke Tamura, Hiroyuki Machida, Characteristics of Cell/B.E. Programming and Introduction of Utilization Example, Journal of Information Processing, Vol.50, No.2, pp.116-128, (2009.02) [Japanese language]

[29] Tetsu Narumi, Tsuyoshi Hamada, Fumikazu Konishi, Acceleration of Particle Method Simulation by Accelerator, Journal of Information Processing, Vol.50, No.2, pp.129-139, (2009.02) [Japanese language]

[30] Reiji Suda, Mathematics of Software Automatic Tuning, Journal of Information Processing, Vol.50, No.6, pp.487-479, (2009.06) [Japanese language]

[31] Shoji Itoh, Support Tools for Software Automatic Tuning, Journal of Information Processing, Vol.50, No.6, pp.499-504, (2009.06) [Japanese language]

[32] Hisayasu Kuroda, Ken Naono, Takeshi Iwashita, Numerical Libraries with Automatic tuning Function, Journal of Information Processing, Vol.50, No.6, pp.505-511, (2009.06) [Japanese language]

[33] Takahiro Katagiri, Progamming Language to Describe Software Automatic Tuning, Journal of Information Processing, Vol.50, No.6, pp.494-498, (2009.06) [Japanese language]

[34] Hiroyuki Takizawa, Software Automatic Tuning in GPU Computing, Journal of Information Processing,

Vol.50, No.6, pp.527-531, (2009.06) [Japanese language]

[35] Zendo Shimoda, PowerXCell and Linear Calculation, Forum on Advanced Scientific Computing 2008 – Focused on Linear Calculation -, (2008.09) [Japanese language]

[36] Toshio Endo, Akira Nukada, Satoshi Matsuoka, Naoya Maruyama, Hideyuki Jitsumoto, Linpack Tuning on a Heterogeneous Supercomputer with Four Types of Processors, 16th Hokkaido "High Performance Computing and Architecture Evaluation" Workshop (HOKKE-2009), (2009.02) [Japanese language]

[37] Takahiro Katagiri, Takao Sakurai, Hisayasu Kuroda, Ken Naono, Kengo Nakajima, OpenATLib: Design and Implementation of General Automatic Tuning Interface, SwoPP2009, (2009.08) [Japanese language]

[38] Richard Vuduc, James W. Demmel, and Katherine A Yelick, OSKI: A library of automatically tuned sparse matrix kernel, Journal of Physics: Conference Series, Vol.16, No.1. 512 (2005)

[39] M. Puschel el al., SPIRAL: A Generator for Platform-Adapted Libraries of Signal Processing Algorithm, International Journal of High Performance Computing Applications, Vol.18, No.1, 21-45 (2004)

[40] Keith Seymour, Haihang You, Jack Dongarra, A Comparison of Search Heuristics for Empirical Code Optimization, Proc.2008 IEEE International Conference on Cluster Computing, pp.421-429, (Oct. 2008)

[41] ROSE, http://www.rosecompiler.org/index.html

[42] Qu Yi, Keith Seymour, Haihang You, Richard Vuduc, Dan Quinlan, POET: Parameterized Optimization for Empirical Tuning, Proc. IPDPS 2007, pp.1-8, (Mar. 2007)

[43] PETSc, http://www.mcs.anl.gov/petsc/petsc-as/

[44] Ananta Tiwari, Chun Chen, Jacqueline Chame, Mary Hall, Jefferey K. Hollingsworth, A Scalable Autotuning Framework for Compiler Optimization, Proc. IPFPS 2009, (May. 2009)

[45] SciDAC, http://www.scidac.gov/

[46] Performance Engineering Research Institute, http://www.peri-scidac.org/perci/

[47] The Office of Advanced Scientific Computing Research http://www.er.doe.gov/ascr/About/about.html

[48] Automatic Tuning Research Group, Survey of Topics in Automatic Tuning Technology, (2008.11) [Japanese language]

Profile

Takao FURUKAWA
Senior Research Fellow
Promoted Fields Unit
Science and Technology Foresight Center
http://www.nistep.go.jp/index-j.html

In an IT venture company, performed R&D on design support systems using computer graphics, and applications applying real-time video processing. At his present position since 2009.

Minoru NOMURA
Affiliated Fellow
Information & Communications Unit
Science and Technology Foresight Center
http://www.nistep.go.jp/index-j.html

At a company, performed R&D on CAD for computer design, and business development in the high performance computing market and ubiquitous market. Now works at STFC. Interested in science and technology trends in information and communications fields: supercomputing, LSI design technologies, etc.

(Original Japanese version: published in November 2009)