

Development of a framework for parallel simulators with various physics and its performance

Kenji Ono,^{a, b} Tsuyoshi Tamaki,^c Hiroyuki Yoshikawa^c

^a*Functionality Simulation and Information team, VCAD System Research Program, RIKEN, 2-1 Hirosawa, Wako, 351-0198, Japan*

^b*Division of Human Mechanical Systems and Design, Faculty and Graduate School of Engineering, Hokkaido University, N13, W8, Kita-ku, Sapporo, 060-8628, Japan*

^c*Fujitsu Nagano Systems Engineering, 1415 Midori-cho, Tsuruga, Nagano, 380-0813, Japan*

Keywords: Object-Oriented Framework; Parallel Computation; MPI; Data Class

1. Application Framework

An object-oriented framework with class libraries is designed to enhance the software development and to manage various physical simulators. The proposed framework provides an efficient way to construct applications using the inheritance mechanism of object-oriented technology (OOT). In addition, the inheritance plays an important role to build applications with a unified behavior. It is expected that the framework brings efficiency for software development and makes easy to operate the developed applications. The framework also delivers high-level conceptual parallel programming environment based on the parallelism of domain decomposition.

1.1. Outline of Framework Library

A proposed system is an object-oriented framework named SPHERE (Skeleton for PHysical and Engineering REsearch), which is designed to apply building unsteady

physical simulators. SPHERE would provide various benefits both developers and end-users as described by following sub-sections.

1.1.1. Basic features of SPHERE

This framework supplies both a control structure and basic functions that are essential for time evolutionary physical simulations. Since a simulation code can be conceptually divided into three parts like, pre-process, main-process, and post-process, programmers can describe their applications using a common structure. Thus, the mechanism that has the skeleton of the control structure should be established and be utilized as a solver base class depicted in Fig. 1.

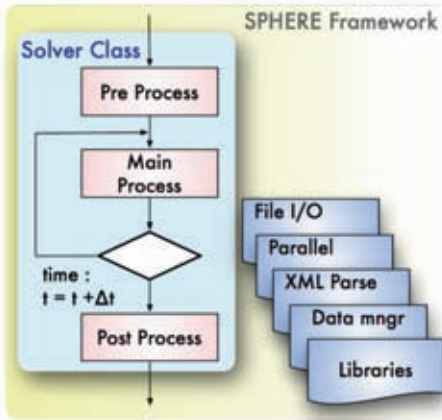


Fig. 1. Control structure of a solver class and provided libraries on a proposed framework. SPHERE framework supplies a skeleton for unsteady physical simulators that require time evolution as well as steady simulation.

A programmer makes a target application class inheriting from the base class and constructs the application by implementing user's code on the derived class. Although the framework is written by the manner of OOT, a procedural programming style is taken over because many scientific applications are described by C/Fortran. The class abstraction that corresponds to the conventional procedural process enables us to port existing software resources to the program on the framework easily. The maximum unit of a class can be assumed by the program unit working as a solver, and defines as a solver class. In addition, SPHERE provides convenient functions as libraries to construct solvers. These functions are consisting of common functions, such as data management, parsing of various parameters described by XML language, file I/O, and so on.

1.1.2. Realization of a unified user interface using a user-defined base class

Making and utilizing the base class that has more concrete and lump functions, the programmer can develop solver classes that have a unified user interface and a behavior. Both the abstraction of common functions and the versatility of the implemented

methods are required for the user-defined base class to realize this idea. To do so, the mechanism of class inheritance in OOT plays an important role.

1.1.3. Application management and execution environment

Programmers can describe their specific solver class using the proposed framework. The written solver class is equivalent to a program unit as an application in the usual sense. The framework has a structure that enables us to register user's solver classes on it. SPHERE behaves as a monolithic application that can execute the registered solver classes by invoking a solver keyword described in an XML parameter file. This mechanism brings the extension of solver coupling into view.

1.1.4. Source code portability

The current framework supports major platforms like UNIX (Linux), Windows, and Mac OSX. The interoperability of compilers turns the mixed language programming into reality. Although the main function of the framework is required to be written by C++, C and Fortran language are available as functions or subroutines. This feature helps us to port existing Fortran program to the solver class. While system call is written by C/C++, the process that requires high performance can be written by Fortran language.

1.1.5. Achievement of high performance

On scientific applications with OOT, it should be taken care to achieve high performance. One may describe the code with the operator overload technique that permits to carry out operations between the classes in order to increase flexibility of the code description. Giving high readability of source code to us, this operator overload technique brings a serious defect on performance due to the internal generation of temporal class objects [1]. To remedy this problem, expression templates technique [2] is proposed and employed in POOMA [3]. Instead, the current framework keeps the performance high by simple implementation, which suppresses the operator overload and passes the address of array variable to subroutines and functions directly.

1.1.6. Easy parallelization from serial code based on domain decomposition

SPHERE provides a high-level parallel programming environment based on the domain decomposition method. An extension to a parallel code can be realized by inserting methods into the serial source code. Several useful patterns for parallelization are prepared and are delivered by the framework. This simple procedure is so powerful to build up the parallel code.

1.2. Organization of framework

The framework takes on the functions between an operating system and applications as shown in Fig. 2. The MPI library is employed to help the parallelization and is used to construct data/parallel manager classes. Several libraries are incorporated in the framework such as file I/O library, libxml2 library that parses the parameter described in XML language and so on. Other libraries, of course, can be added to the framework.

SolverBase class, which is a base class provided by the framework, has all functions of libraries incorporated and the procedure of the solver control. The classes Ω and β just above on SolverBase class in Fig. 2 indicate the user-defined base classes; and four different kind of solver classes are build on them.

As mentioned above, although this framework is designed with OOT, the framework SPHERE has advantages of high ability of porting from and high affinity with the existing C/Fortran codes, introducing the abstraction based on conventional procedure and mixed language programming. These merits are unique feature that is not found in other OOT frameworks like SAMRAI [4], Overture [5], and Cactus [6].

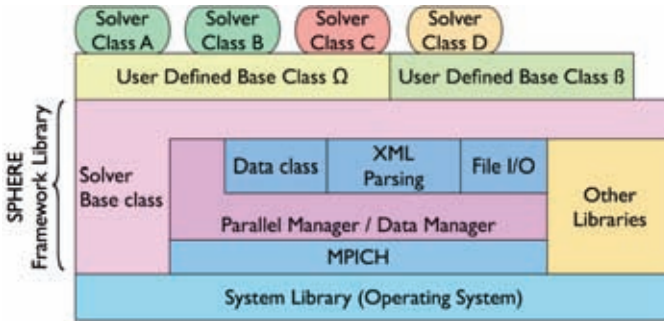


Fig. 2. Diagram of SPHERE architecture. User-defined base classes are useful class that includes common functions for specific applications such as flow simulators, structural simulators, and electro-magnetic solvers. Each solver class A~D indicates a real application in the usual sense.

2. Application packaging by inheritance of a user-defined base class

As shown in Fig. 2, SPHERE framework consists of libraries and provides its functions as a base class to the programmers who will write an application program using the inheritance mechanism. SPHERE is made up of highly abstracted generic program components that support the development of time evolutionary simulator. More concrete and functionally lumped processes should be described with those basic functions to make a specific application. Consequently, the authors employed the approach that the programmer could design the user-defined base class that has versatility in a way for limited applicable scope; programmer can write the specific application inheriting the user-defined base class.

In Fig. 3, SklSolverBase class is a super class, which is able to use all the function of the incorporated libraries. On the other hand, FlowBass class is the user-defined base class that is derived from SklSolverBase class and is designed for flow simulators. This FlowBase class has functions such as the solver control, boundary control and treatment, preprocess of voxel data, the retention and reading parameters in XML files. The user-defined base class includes basic user interface of applications.

This approach offers both a programmer and end-users several merits. First, the development of application can be efficient. The base classes enable us to write source

code with high-level description. The programmers can concentrate their effort on algorithm and maintain their codes without great effort. Second, the unified user interface will deliver for the end-users. In this context, since the behavior of the application on the framework is almost same regardless of the different types of physics or methods utilized, the barrier and the learning curve at the introducing the applications will be greatly reduced. As a demerit, meanwhile, the current way has no small effect on the developers, who are enforced in coding manner and programming style as well as the concept of OOT.

In Fig. 3, five solver classes are derived from FlowBase and registered on SPHERE. Each keyword of CBS, CVC, RSC, and PBC means the shape approximation by binary voxel for an object on the Cartesian mesh with staggered variable arrangement, by volume fraction on the Cartesian with collocated, by Signed Distance Function (SDF) on non-uniformly Cartesian with collocated, and by binary voxel on Octree with collocated, respectively. In addition, _H, _CP, _IC keywords indicate heat solver, compressible flow solver, and incompressible flow solver, respectively. Thus, the proposed framework has two aspects; a development environment and a run-time environment.

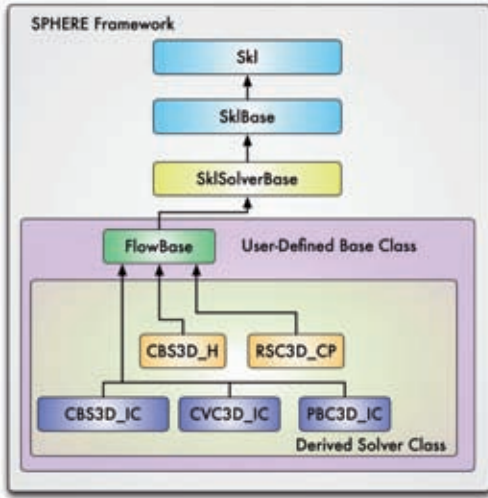


Fig. 3. A user-defined base class and derived solver classes. While framework provides Skl, SklBase, and SklSolverBase classes, other classes are written by application developers.

3. Data Class

Parallel process, which the framework takes on, is one of the key functions in SPHERE. In the case of distributed parallel computation, generally, programming and the implementation of boundary conditions tend to be complex. SPHERE provides an abstracted programming environment where the description of the low-level process is hidden by introducing a concept of data class. While parallel process will be discussed

for a domain decomposition method on a structured grid system in this paper, the proposed framework has no limit to incorporate any data structure, calculation algorithm, and parallel algorithm.

Data class is a collection of class to provide useful functions, which are designed for accessing to one-dimensional array data inside the data class as if the data class is multi-dimensional array. Data class library consists of two layers, i.e., data class itself and data manager class which keeps it up, to provide flexible programming environment. An abstraction of data class is designed so that the provided operations and the functions are confined to the management, the operation, and the parallelization for the array. The selection of implemented functions are paid attention to following issues; the portability of legacy Fortran programs, the affinity on the program development with mixed language, the suppression of inherent overhead for the object-oriented language, and the flexibility of programming.

3.1. Role and function of data class

Data class, which is one of a class libraries inside SPHERE, is in charge of management of arrays such as creating, deleting, allocating arrays on memory space, and so on. A base class of data class is implemented as C++ template class. N-dimensional class inherits from the base class that has only one-dimensional array and provides interface to access array data with n-dimensional indices. In the concrete, as shown in Fig. 4, a set of data class has SklArrayidxN (N stands for 1, 2, 3, 4) classes, which are derived from SklArray (Base) class. These classes are also distinguished by the variable type of scalar and vector.

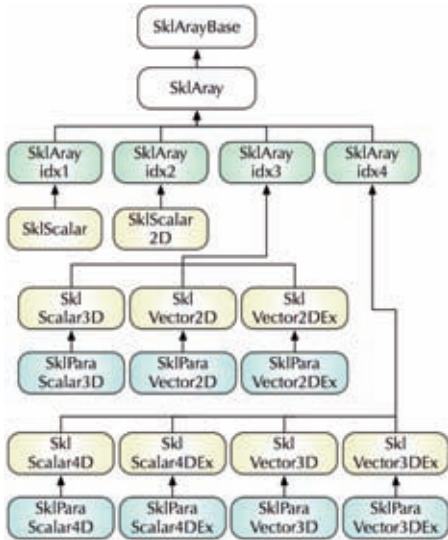


Fig. 4. Class hierarchy for data class. SklArrayBase is implemented as an abstract class with pure virtual functions.

In the case of parallel solver, this data class has functions of data communication between each sub-domain. The basic idea is borrowed from Ohta [7], but the concept is extended by the combination of data class and manager class so that SPHERE can give more flexibility to manage various types of data array. Derived parallel version has additional information of guide cell, which acts as buffer region to synchronize data between adjacent domains.

Data class provides the functions such as the acquisition of top address of array data inside data class, the operator to access n-dimensional array, synchronization according to the domain decomposition manner.

3.2. Role and function of manager class

Manager class has two types; one is for serial code and the other is for parallel. Serial manager class performs the creation, the deletion, and the registration of a data class. Parallel manager class has a function of management of parallel environment in addition to one of serial manager class.

3.3. Functions for parallel process

In the case of parallel code, the parallel manager and related classes are derived from SklBase class as shown in both Fig. 3 and Fig. 5. SklParaManager class has SklParaNodeInfo class, SklVoxelInfo class, and SklParaIF class in its inside (see Fig. 6). SklParaIF is an abstract class and real communication procedures are implemented on SklParaMPI class. Of course, other communication libraries like PVM and LAM are another option. SklParaNodeInfo and SklVoxelInfo classes have the information such as the number of cells for a whole computational domain, the number of nodes, the number of domains, and rank number of each domain.



Fig. 5. Class hierarchy for manager class. Both SklBase and SklParaIF are abstract class. SklParaManager class includes SklParaNodeInfo, SklVoxelInfo, and SklParaIF class.

Fig. 6 shows the behaviour of both a solver class and SklParaManager classes. Node numbers indicate each sub-domain. A solver class in SPHERE has the class objects of SklParaManager class as one of the member, and requests parallel instructions to SklParaManager objects. SklParaManager class takes in charge of all communications as far as parallel process in the solver class. When synchronizing, solver class object issues the synchronization instruction to SklParaManager object, and then SklParaManager objects communicate each other.

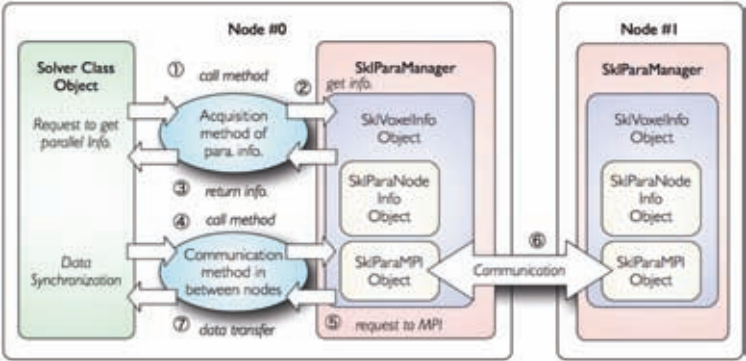


Fig. 6. Role of parallel manager class. SkiParaManager class is in charge of all communications between sub-domains and includes low-level implementation for communication.

4. Performance Evaluation

To confirm the performance of the developed application on the framework, a benchmark program was used to evaluate on several parallel machines.

4.1. Test Environment and Preliminarily Benchmark Code

Benchmark is performed in several environments as shown in Table 1. RSCC [8], which is the acronym for “Riken Super Combined Cluster,” is Linux cluster system and consists of 1,024 nodes (2048 CPUs) in total. SCore is used as a clustering middleware.

Table 1. Specification for evaluated environments.

	RSCC	SGI Altix 4700	Xeon Cluster
CPU	Xeon (Prestonia)	Itanium2 9000 (Montecito)	Xeon 5160 (Woodcrest)
Clock [GHz]	3.06	1.6	3.0
CPU (Core) / Node	2 (2)	2 (4)	2 (4)
Node	1,024	32	32
Memory/node	1GB	4GB	8GB
Middleware	Score 5.6.1	SGI ProPack 5	SGI ProPack 5
Interconnect	InfiniBand (8Gbps)	NUMA link (8.5GB/s)	InfiniBand (8Gbps)
Compiler	Fujitsu Linux Parallel Package	Intel C++/Fortran Compiler 9.1	Intel C++/Fortran Compiler 9.1
MPI Library	Included in SCore	SGI MPT 1.14	Voltaire MPI

A program used for the benchmark is a code to solve an elliptic partial differential equation by Jacobi relaxation method, which is commonly found in an incompressible flow solver or an electro-magnetic solver. This code has the characteristics of memory-bound, that is, the memory bandwidth has much impact on the performance in addition to CPU ability because the number of arithmetic operation is comparable with the number of load/store in the source code. This original code is parallelized by MPI library based on the domain decomposition method. The benchmark code on the framework [8] is ported from the original code which is parallelized by the similar MPI coding.

4.2. Parallel Performance

Table 2 shows the comparison results of the measured timing on RSCC for both the original code without the framework and SPHERE code. It was found that the speed of operation on SPHERE code goes down around five percent in comparison with the original code. This point is thought as reasonable because the framework has the inherent overhead for its parallel process in exchange for convenience in writing code.

Table 2. Measured performance in GFLOPS for both the original and SPHERE code with different size (Original / SPHERE code).

Number of CPU core	256x128x128 (M)	512x256x256 (L)	1024x512x512(XL)
32	11.13 / 10.21	11.34 / 11.50	11.22 / 12.57
64	22.42 / 20.99	21.19 / 22.85	21.48 / 23.98
128	38.53 / 35.91	44.44 / 44.68	40.59 / 48.94
256	65.36 / 62.56	83.12 / 83.40	80.18 / 90.93
512	109.00 / 107.14	158.09 / 160.28	155.10 / 180.95
1024	150.57 / 147.25	264.93 / 251.17	316.11 / 297.90

Next, Table 3 shows the performance on different kind of machines for SPHERE code. SGI Altix 4700 system shows the best performance among them due to the fastest interconnect speed and also, achieves high speed up ratio; this performance is over 30 percent of its theoretical performance. The Xeon (Woodcrest) cluster demonstrated second best performance. In this result, no gain is observed from one core to two cores because this system shares a system bus with two cores and the benchmark code is characterized by memory-bound.

5. Concluding Remarks

An object-oriented framework for physical simulators is developed to enhance efficiency of the software development and to manage various applications, and then, a preliminarily benchmark is performed on several parallel environments. Finally, it was found that the newly designed framework worked well and exhibited reasonably good

performance in comparison to the original benchmark code without the framework. This framework will be deliver from our website [9].

Acknowledgement

This study was partially supported by Industrial Technology Research Grant Program in '04 from New Energy and Industrial Technology Development Organization (NEDO) of Japan and, by computational resources of the RIKEN Super Combined Cluster (RSCC). We would like to thanks to Mr. Takeda, SGI Japan, Ltd. for the cooperation of the benchmark.

Table 3. Measured performance in GFLOPS and speed up ratio of SPHERE code for 512x256x256 (L size) grid points. Values with * mark are for reference. Those are results for 256x128x128 (M size) due to the restriction of amount of main memory. The figures shown in parenthesis indicate the speed up ratio of parallelization.

Number of CPU core	RSCC	SGI Altix 4700	Xeon Cluster
1	0.55* (1.00)	2.57 (1.00)	1.33 (1.00)
2	0.97* (1.76)	3.45 (1.34)	1.35 (1.02)
4	1.58 (2.87)	6.84 (2.67)	2.35 (1.71)
8	3.08 (5.60)	13.67 (5.33)	5.08 (3.81)
16	5.65 (10.27)	26.67 (10.39)	10.32 (7.74)
32	11.73 (21.33)	54.29 (21.16)	20.82 (15.61)
64	23.30 (42.36)	111.73 (43.55)	44.54 (33.39)
128	45.55 (82.82)	268.51 (104.65)	88.80 (66.57)

Reference

1. Bulka, D., and Mayhew, D., Efficient C++ Performance Programming Techniques, Addison-Wesley, (1999).
2. Veldhuizen, T., "Expression Templates," C++ Report, Vol. 7 No. 5 June, pp. 26-31 (1995).
3. <http://www.nongnu.org/freepooma>
4. Hornung, R.D., and Kohn, S.R., "Managing Application Complexity in the SAMRAI Object-Oriented Framework," in Concurrency and Computation: Practice and Experience (Special Issue), 14, pp. 347-368 (2002).
5. Henshaw, W.D., "Overture: An Object-Oriented Framework for Overlapping Grid Applications," AIAA conference on Applied Aerodynamics (2002), also UCRL-JC-147889.
6. <http://www.cactuscode.org/>
7. Ohta, T., and Shirayama, S., "Building an Integrated Environment for CFD with an Object-Oriented Framework," Transactions of JSCES, No. 19990001, in Japanese (1999).
8. <http://accc.riken.go.jp/>
9. <http://vcad-hpsv.riken.jp/>

Parallel Computational Fluid Dynamics 2007
Implementations and Experiences on Large Scale and
Grid Computing

Tuncer, I.H.; Gülcat, Ü.; Emerson, D.R.; Matsuno, K.
(Eds.)

2009, XII, 488 p. 333 illus., 197 illus. in color., Softcover
ISBN: 978-3-540-92743-3