V European Conference on Computational Fluid Dynamics
ECCOMAS CFD 2010
J. C. F. Pereira and A. Sequeira (Eds)
Lisbon, Portugal, 14–17 June 2010

OPTIMIZATION OF THE APPLICATION MIDDLEWARE "SPHERE" FOR BLUE GENE/L SYSTEM

Satoshi ITO* **and Kenji ONO***

*Computational Science Research Program, RIKEN
Hirosawa2-1, Wako-shi, Saitama, 351-0198 JAPAN
e-mail: {sito,keno}@riken.jp

Key words: Middleware, FEM, Optimization, IBM Blue Gene/L

Abstract. *We developed SPHERE, which is designed as an object-oriented solver framework for developing and executing simulation codes. Functions of configuration parser (XML), file I/O, dynamic memory allocation, parallelization, and flow control system are provided, thus utilization of skeleton program ease both developing new codes and porting legacy codes. Typical processes of FEM such as I/O, mathematical operations, matrix solvers, and so on are provided by this middleware so that the implementation becomes easy for application developers. Parallelization is also automatically applied by masking message passing. In addition, high performance will be achieved because mathematical operations and solvers are optimized for each architecture. In this study, we optimized SPHERE for IBM Blue Gene/L. This architecture has two floating point number processing units (DFPU) and parallel load/store system in each core. In order to utilize these systems, the data structure is converted from double floating point array into complex floating point array and intrinsic functions are called in the mathematical operation processes which are prepared by SPHERE. We ported an incompressible fluid analysis code to Blue Gene/L, which was developed on a PC. About 3.5 times speed-up and 29.5% of peak performance are achieved by using optimized SPHERE.*

1 INTRODUCTION

The Next-Generation Supercomputer[1] which is being developed at RIKEN is the new National Leadership Supercomputer in Japan. One of the purposes of our project is integrated simulation of living matter from molecular level to the whole body. It is classified into multi-physics/multi-scale simulation, therefore, coupling control of each application and message passing of data on coupling interfaces, and extremely large scale computation must be required.

By the way, the Next-Generation Supercomputer is composed of SPARC64TM VIII fx CPUs (8cores, 128GFLOPS) and high speed network so that parallelization and optimization of developed codes are inevitable due to high performance computation on the Next-Generation Supercomputer. However, these extra works are so burdensome thus the implementation becomes a barrier for application developers.

To overcome these adversities, we developed SPHERE[2], which is a kind of application middleware. SPHERE is designed as an object-oriented solver framework for developing and executing simulation codes. Functions of configuration parser (XML), file I/O, dynamic memory allocation, parallelization, and flow control system are provided, thus utilization of skeleton program ease both developing new codes and porting legacy codes.

Here, we focus on unstructured grid method such as finite element method. Typical processes of FEM such as I/O, mathematical operations, matrix solvers, and so on are provided by the middleware so that the implementation becomes easy for application developers. Parallelization is also automatically applied by masking message passing. In addition, high performance will be achieved because mathematical operations and solvers are optimized for each architecture.

In this study, we optimized SPHERE for IBM Blue Gene/L. Blue Gene series are one of the fastest supercomputers in the world, thus it is appropriate as an example architecture of optimization target. This architecture has two floating point number processing units (Double FPU: DFPU) and parallel load/store system in each core. In order to utilize these systems, the data structure is converted from double floating point array into complex floating point array and intrinsic functions are called in the mathematical operation processes which are prepared by SPHERE.

2 SOFTWARE DEVELOPMENT USING SPHERE

In this section, we briefly explain SPHERE, the concept and its main functions. Then, we develop a finite element fluid analysis code as an example of software development using SPHERE.

2.1 SPHERE

As described before, SPHERE is an application middleware, which helps researchers to both develop and operate simulation codes. Figure 1 shows the concept of SPHERE. You can easily develop your code using various classes and functions such as I/O, linear matrix solvers, mathematical operations, and etc. Skelton code are created by the utility in order to describe the algorithm of your code. Variables and parameters are belongs to the basic class so that development of your own subroutines is easy.

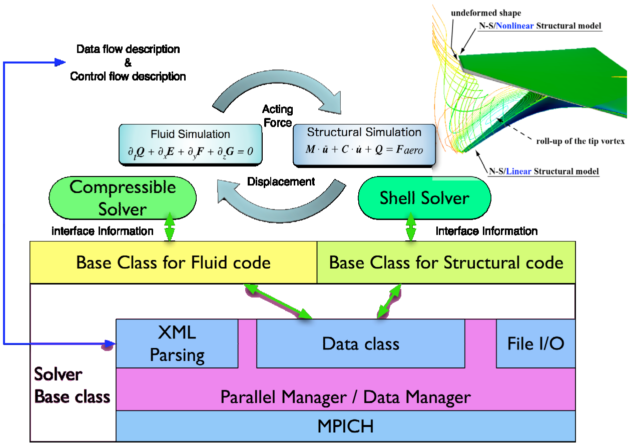


Figure 1: The concept of SPHERE.

2.2 Development of an incompressible fluid analysis code

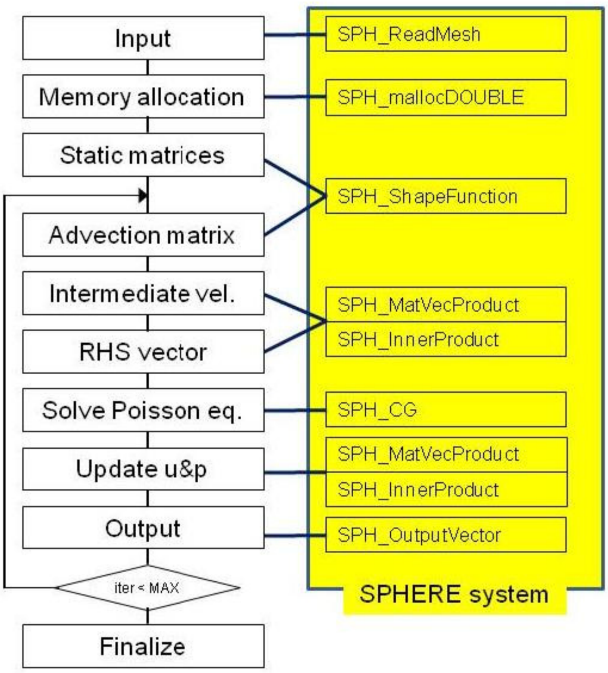
A finite element fluid analysis code was developed due to evaluate the efficiency of SPHERE. The governing equations for the incompressible fluid are following:

$$\frac{\partial u_i^*}{\partial t^*} + u_j^* u_{i,j}^* - \sigma_{ij,j}^* = 0 \quad \text{in } \Omega \tag{1}$$

$$\sigma_{ij}^* = -p^* \delta_{ij} + \frac{1}{Re}\left(u_{i,j}^* + u_{j,i}^*\right) \tag{2}$$

$$\frac{\partial u_i^*}{\partial x_i^*} = 0 \quad \text{in } \Omega \tag{3}$$

Here, u, p, Re, Ω denote velocity, pressure, Reynolds number, and the calculation domain, respectively. After applying finite element discretization and Fractional Step method, the algorithm shown in Figure 2 was obtained.




```
#include"MakeDiffMatrix.h"

void MakeDiffMatrix(Control *ctrl, Mesh   *mesh, Matrix9 *diffMAT)
{

  for ( ie = 0; ie < mesh->nInner+mesh->nOverlapped; ie++ ){

  SPH_ShapeFunction(ie, ctrl, mesh, &detJ, N, NX, NY, NZ);
  detJ *= ctrl->Reinv;

  for  ( in = 0; in < 8; in++ ){
   for ( im = 0; im < 8; im++ ){

     for ( ig = 0; ig < 8; ig++ ){

       index = 64*ie + 8*in + im;
       K = detJ * ( NX[in][ig]*NX[im][ig]
                +NY[in][ig]*NY[im][ig] + NZ[in][ig]*NZ[im][ig]);
#define SDIVERGENCE
#ifdef SDIVERGENCE
       diffMAT->xx->mat[index] += detJ * NX[in][ig]*NX[im][ig] +K;
       diffMAT->xy->mat[index] += detJ * NY[in][ig]*NX[im][ig];
       diffMAT->xz->mat[index] += detJ * NZ[in][ig]*NX[im][ig];

       diffMAT->yx->mat[index] += detJ * NX[in][ig]*NY[im][ig];
       diffMAT->yy->mat[index] += detJ * NY[in][ig]*NY[im][ig] +K;
       diffMAT->yz->mat[index] += detJ * NZ[in][ig]*NY[im][ig];

       diffMAT->zx->mat[index] += detJ * NX[in][ig]*NZ[im][ig];
       diffMAT->zy->mat[index] += detJ * NY[in][ig]*NZ[im][ig];
       diffMAT->zz->mat[index] += detJ * NZ[in][ig]*NZ[im][ig] +K;
#else
       diffMAT->xx->mat[index] += K;
       diffMAT->yy->mat[index] += K;
       diffMAT->zz->mat[index] += K;
#endif

     }

    }
   }
  }
}
```

Figure 2: The algorithm and source code of the developed code.

As demonstrated in Figure 2, SPHERE provides applicable classes and functions for each step of the algorithm.

3 OPTIMIZATION FOR BLUE GENE/L

3.1 IBM BLUE GENE/L

BM Blue Gene series is one of the fastest supercomputers in the world. Here, we optimize SPHERE for Blue Gene/L. This machine employs three-dimensional torus network system that is determined to built in the Next-Generation supercomputer. Therefore, it is a good example of optimization for the Next-Generation supercomputer.

Blue Gene/L is composed of PowerPC 440 based dual core processor and DDR SDRAM, and has 1024 nodes in a rack. Each processor core can load/store two sequential floating point numbers at once. Thus, fully utilization of these features is necessary to achieve good performance on it.

3.2 Optimization

Most of calculation time is spent on mathematical operations in unstructured grid methods. Actually, the result of which IBM XL compiler analyzed the developed code indicated that more than 90% of elapse time was used for the matrix-vector multiplication procedure in some numerical tests. Here, we mainly focus on this procedure and optimize it for Blue Gene/L.

Matrix-vector multiplication proceeds with respect to each element. In this element loop, matrix component and vector data are loaded from the memory, then multiplication is executed, and finally the result data array is stored to the memory. First, we employed complex data type that is recognized by Blue Gene XL compilers in order to apply parallel load/store instructions. This data type can utilize DFPU by using intrinsic functions simultaneously. Second, we used temporary buffer array for the right-hand side vector. Matrix components are aligned continuously on the memory, but the processor cannot load vector data sequentially because it is referred indirectly. Cache tuning is also significantly efficient because the CPU type is scalar. Temporary buffer also has advantage on this point. In addition, we divided the element loop into some element parts for enhancing cache hit ratio.

3.3 Test results

We ported the code to Blue Gene/L without any optimization. Figure 3 shows six cases of results of the original code and the result of the developed code. Here, the original code means that was developed without SPHERE, and optimized only by compiler options. The developed code uses SPHERE that is optimized for Blue Gene/L.

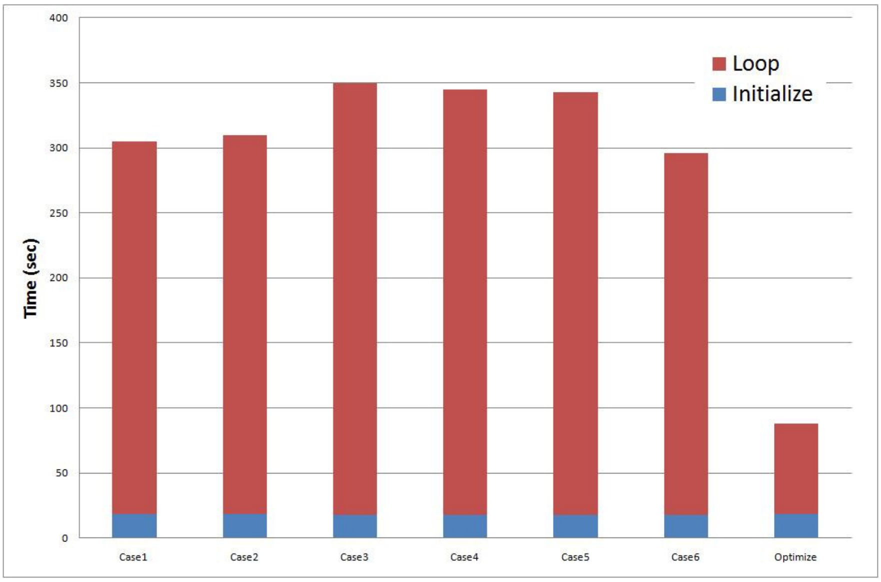


Figure 3: Measured flops of the developed code.

As is shown in the FIGURE 3, the original code spent about 300 seconds for all cases, but the developed code spent only 86 seconds. The performance of the developed code was 29.5% of the theoretical performance.

4 CONCLUSIONS

We developed an application middleware SPHERE and optimized it for IBM Blue Gene/L. An incompressible fluid analysis code was developed using SPHERE and ported to Blue Gene/L. About 3.5 times speed-up and 29.5% of peak performance are achieved by using optimized SPHERE.

AKNOWLEDGEMENTS

This research was supported by Research and Development of the Next-Generation Integrated Simulation of Living Matter, a part of the Development and Use of the Next-Generation Supercomputer Project of the Ministry of Education, Culture, Sports, Science and Technology (MEXT).

REFERENCES

[1] http://www.nsc.riken.jp/index-eng.html

[2] K. Ono and T. Tamaki, "SPHERE - Development of a Framework and Execution Environment for Physical Simulation." , *JSCES*, No. 20060031, 2006.