31^{st} International Conference on Parallel Computational Fluid Dynamics
ParCFD'2019

OPTIMIZATION OF A GAS-PARTICLE FLOW SOLVER ON VECTOR SUPERCOMPUTERS

Yoichi Shimomura[‡*], Midori Kano[†], Takashi Soga[‡*], Kenta Yamaguchi[‡*], Akihiro Musa[†*], Yusuke Mizuno[**], Shun Takahashi[**], Ryusuke Egawa[*], and Hiroyuki Takizawa[*]

[‡] NEC Solution Innovators, [†] NEC Corporation, Tokyo, 136-8608, Japan
e-mail: {tak-soga@ti, m-kano@ed.cnt, y-shimomura@wx, k-yamaguchi@vt, a-musa@bq}.jp.nec.com
[**] Tokai University, Hiratsuka, 259-1292, Japan e-mail: {7btad010@mail.u-tokai.ac, takahasi@tokai-u}.jp
[*] Tohoku University, Sendai, 980-8578, Japan
e-mail: {egawa, takizawa}@tohoku.ac.jp

Key words: Code optimization, Gas-particle flow, Vector Supercomputer

Abstract. This paper discusses code optimization of the shot peening process, which is a gas-particle flow solver, to perform large simulations on vector supercomputers. Since the simulation code is memory intensive, the simulation code is optimized to reduce memory loads and evaluate the performance on vector supercomputers: NEC SX-Aurora TSUBASA and SX-ACE. The optimized code can achieve about twice higher performance than the original code on the latest vector supercomputer, SX-Aurora TSUBASA.

1 Introduction

The shot peening process impacts a metal surface with a large number of shots (particles) and generates a compressible residual stress on the surface. Since the physical phenomena are collisions of very fine and massive particles in a short time, theoretical and experimental studies have difficulties in clarifying the behavior of particles and the process of deformation of workpieces in detail. To investigate the physical phenomena, Mizuno et al. [1] [2] have developed a simulation code that utilizes the three-dimensional incompressible Navier-Stokes equations with the immersed boundary method (IBM), and the simulation code was implemented in the vector supercomputers: NEC SX-ACE and SX-Aurora TSUBASA. Moreover, we have optimized the simulation code so as to improve the memory access efficiency, because the simulation code is memory intensive and its sustained performance strongly depends on the memory access performance.

2 Numerical Methods

The governing equations of the simulation code are the three-dimensional incompressible Navier-Stokes equations and the equation of continuity [2]. The fractional step method is applied for time marching. The equally spaced three-dimensional Cartesian grid is utilized. The convection term is evaluated by the second-order skew-symmetric scheme, and the diffusion term is discretized using the second-order central-difference scheme. Moreover, the pressure and diffusion terms are calculated by the second-order finite-difference method. The Poisson equation of the pressure is calculated using the Jacobi iterative method. The simulation code uses the IBM on the basis of the level set functions to represent particle boundaries. Figure 1 shows cells around a particle, and the cells are classified into three region categories of *fluid cells*, *ghost cells*, and object (particle) cells.

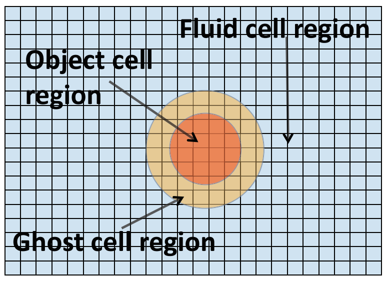


Figure 1: Diagram of cells around a particle in IBM.

3 Evaluation Systems

SX-Aurora TSUBASA is the latest-generation vector computing system [3], which consists of one or more card-type vector engines (VEs) and their hosts, called vector hosts (VHs). The VE is comprised of a vector processor, a 16 MB last-level cache, and six HBM2 memory modules. The VH is a standard x86 Linux server and executes another simple operating system for managing VEs. The VH is connected with up to eight VEs via two PCI express switches, and Fig. 2 shows the maximum configuration of one VH. In the figure, one of two Xeon processors serves as a VH, and eight VEs and two Infiniband Host Channel Adapters are connected to the VH.

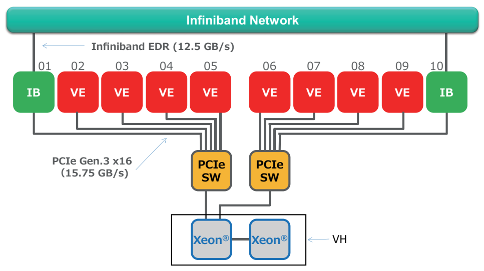


Figure 2: Block diagram of an SX-Aurora TSUBASA system.

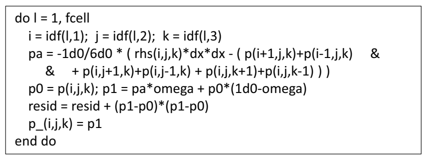
SX-ACE is a previous-generation vector computing system, which consists of one processor (four cores) with a peak performance of 256 Gflop/s, and a 64 GB memory with a 256 GB/s memory bandwidth [4]. The SX-ACE system is composed of 1024 nodes connected via a custom interconnect network (called IXS) by a 4 GB/s bandwidth. Table 1 lists the hardware configurations of the two systems, SX-ACE and SX-Aurora TSUBASA..

4 Optimization Technique

In the simulation code, the solver of a Poisson equation for pressure calculation is the most expensive kernel, and we thus preferentially optimize the solver. Specifically, the memory access pattern of the solver is modified from so-called indirect memory access to sequentially access. Figure 3 shows a part of the original solver routine, where variable *fcell* and array *idf* correspond to the number of fluid cells and the coordinate values of the fluid cells in Fig. 1, respectively. Access to array P is mostly done with an indirect memory access pattern using a list vector, *idf*. This code can calculate only the fluid cells unlike the optimized code mentioned later. However, it cannot achieve high memory access performance, because it requires a lot of memory access operations and also its indirect memory access needs a long latency. Figure 4 illustrates our optimized version of

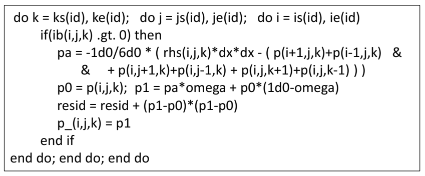
Table 1: Specifications of CPU and evaluated system on each machine.

	SX-ACE		SX-Aurora TSUBASA	
	CPU	System	CPU	System
Clock freq. (GHz)	1		1.4	
No. of cores	4	16	8	16
Perf. (Gflop/s)	256	1024	2150	4300
Mem. BW (GB/s)	256	1024	1220	2440



```
do l = 1, fcell
   i = idf(l,1);  j = idf(l,2);  k = idf(l,3)
   pa = -1d0/6d0 * ( rhs(i,j,k)*dx*dx - ( p(i+1,j,k)+p(i-1,j,k)    &
      &    + p(i,j+1,k)+p(i,j-1,k) + p(i,j,k+1)+p(i,j,k-1) ) )
   p0 = p(i,j,k); p1 = pa*omega + p0*(1d0-omega)
   resid = resid + (p1-p0)*(p1-p0)
   p_(i,j,k) = p1
end do
```

Figure 3: Source list of original code.



```
do k = ks(id), ke(id);   do j = js(id), je(id);   do i = is(id), ie(id)
      if(ib(i,j,k) .gt. 0) then
          pa = -1d0/6d0 * ( rhs(i,j,k)*dx*dx - ( p(i+1,j,k)+p(i-1,j,k)   &
             &    + p(i,j+1,k)+p(i,j-1,k) + p(i,j,k+1)+p(i,j,k-1) ) )
          p0 = p(i,j,k);  p1 = pa*omega + p0*(1d0-omega)
          resid = resid + (p1-p0)*(p1-p0)
          p_(i,j,k) = p1
      end if
end do; end do; end do
```

Figure 4: Source list of optimized code.

the solver. The arrays *ks*, *ke*, *js*, *je*, *is*, and *ie* indicate the sizes of the calculation domain in the three-dimensional Cartesian grid. The array *ib* is used to identify the region category, and a grid point at (i,j,k) represents a fluid cell if $ib(i,j,k) > 0$. The optimized code processes every grid point in the calculation domain, irrespective of whether it is a fluid cell or not. As a result, the optimized code executes more instructions than the original one. However, this code does not use an indirect memory access, and sequentially accesses continuous memory regions.

5 Performance Evaluation

We evaluate the performances of the original and optimized codes, which are parallelized with the MPI library and executed using 16 cores on both SX-Aurora TSUBASA and SX-ACE. The model size is $128 \times 128 \times 512$ and the number of particles is 100. Figure 5 shows the effect of optimization on SX-Aurora TSUBASA. *V. Load Exec* indicates the stall time of the processor due to data loads from the memory. The V. Load Exec time of the original code accounts for more than half of the execution time. Thanks to the optimization, the V. Load Exec time of the optimized code is reduced from 1094.3 to 386.6 seconds, which means 2.8 times higher performance in terms of memory access efficiency. We apply the continuous memory access instead of the indirect memory access, and the total memory access latency time decreases. Moreover, the number of the vector load elements is reduced from 4.17×10^{12} to 3.42×10^{12}.

Figure 6 shows the execution times of the optimized code on SX-ACE and SX-Aurora TSUBASA. The performance of SX-Aurora TSUBASA is 2.2 times higher than that of SX-ACE. Since the code is memory intensive, the performance of the code is limited by the memory bandwidth of each system. Specifically, the aggregated memory bandwidth of the SX-Aurora TSUBASA system is 2,440 GB/s, and 2.38 times larger than that of SX-ACE, 1,024 GB/s. Therefore, the results indicate that the sustained performance of the code is mostly determined by the memory bandwidth, and clearly show the importance of memory bandwidth for achieving high sustained performance on this kind of computational fluid

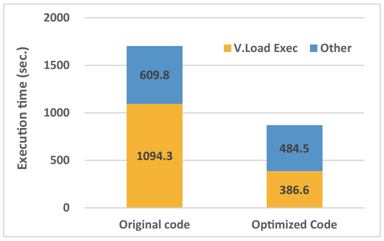


Figure 5: Execution times of original and optimized codes on SX-Aurora TSUBASA.

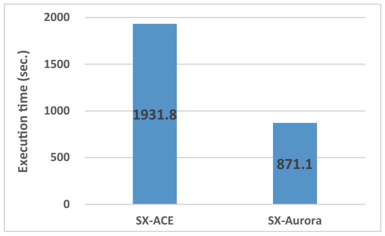


Figure 6: Execution times of optimized code on SX-ACE and SX-Aurora TSUBASA.

dynamics simulations.

6 Conclusions

We optimize the simulation code of the shot peening process and evaluate its performance on SX-Aurora TSUBASA and SX-ACE. The performance of the optimized code is about twice higher than that of the original code on SX-Aurora TSUBASA. Moreover, SX-Aurora TSUBASA can achieve 2.2 times higher performance than SX-ACE. We have executed the large-model size of $1,024 \times 1,024 \times 16,384$ with 100,000 particles on 4,096 cores SX-ACE within 20 days. On the other hand, this work clearly shows that SX-Aurora TSUBASA is expected to drastically reduce the execution time even if such a large model is used. Therefore, we will evaluate the performance of the larger-model size on SX-Aurora TSUBASA when the large system of SX-Aurora TSUBASA, named A500-64, is released.

Acknowledgment

This work is supported by "Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures" in Japan (Project ID: jh180051-NAJ).

REFERENCES

[1] Mizuno, Y., Takahashi, S., Nonomura, T., Nagata, T., Fukuda, K., "A Simple Immersed Boundary Method for Compressible Flow Simulation around a Stationary and Moving Sphere," In: Mathematical Problems in Engineering Volume 2015, Article ID 438086, 17 pages (2015).

[2] Mizuno, Y., Takahashi, S., Fukuda, K., Obayashi, S., "Direct Numerical Simulation of Gas-Particle Flows with Particle-Wall Collisions Using the Immersed Boundary Method," In: applied science Volume 8, Issue 12 (2018).

[3] Yamada, Y., Momose, S.,"Vector Engine Processor of NEC's Brand-New Supercomputer SX-Aurora TSUBASA," In: International symposium on High Performance Chips (Hot Chips), Cupertino, USA (August 2018).

[4] Momose, S., "SX-ACE, Brand-New Vector Supercomputer for Higher Sustained Performance I," In: Resch, M., et al. (eds.) Sustained Simulation Performance 2014, pp. 57-67, Springer-Verlag (2014).