CFD Letters www.cfdl.issres.net Vol. 5 (4) – December 2013

Scalar Tuning of a Fluid Solver using Compact Scheme for a Supercomputer with a Distributed Memory Architecture

Hikaru Aono[1a*], Taku Nonomura[1a], Nobuyuki Iizuka[2], Takahiko Ohsako[3], Tomohide Inari[4], Yasutoshi Hashimoto[5], Ryoji Takaki[1], Kozo Fujii[1]

[1] *Institute of Space and Astronautical Science, JAXA, JAPAN*

[2] *Institute of Industrial Science, University of Tokyo, JAPAN*

[3] *Central Soft Co. Ltd. JAPAN*

[4] *Technical Computing Solutions unit, Fujitsu Limited, JAPAN*

Received:07/04/2013 – Revised 01/11/2013 – Accepted 02/11/2013

Abstract

The scalar tuning of a compressible fluid solver for a supercomputer with a distributed memory architecture is conducted. We use the K computer which is one of the peta-scale supercomputers recently developed in Japan. A computational code “LANS3D” and its high-order compact differencing option are tuned. The original version of the code achieves approximately 4.5% of full performance of CPU for the simple test case. Scalar tuning based on combining do-loops works well, and the tuned code attains about 10% of full performance for the same case. The reasons are the improvement in the use of the cache, the suppression of the data transfer, and the efficient use of the data that once transferred to the cache from the memory that results in hiding the low speed of data transfer. The tuned code becomes twice faster than the original one in the wall-clock time and enables us to perform over-160-case parametric study about airfoil flow computation by large-eddy simulations with high-order accurate and high resolution numerical scheme.

Keywords: scalar tuning; compressible fluid solver; compact scheme; large-eddy simulation; large scale computation

1. Introduction

Recently, supercomputers based on distributed memory architecture with many computer nodes, such as “the K computer” [1], have been developed. With regard to the research in fluid dynamics, it is expected that such supercomputers provide precious opportunities to address various unsolved and interesting problems including high Reynolds number flow and multi-physics problems. However, the byte-per-flops (B/F) ratios of typical new supercomputers tends to scale-down, thus fluid solvers, which generally require a large amount of data transfer from the memory to the cache and the register of a processor, become inefficient. This is very severe condition for fluid dynamics research with the supercomputers. Acceleration of computation based on scalar tuning becomes one

[a] These authors contributed equally to this work.

[*] Corresponding author: Hikaru Aono

Email: aono@flab.isas.jaxa.jp Telephone: +81-50-3362-3751

PII: S2180-1363(13)54143-X

of keys in order to improve the efficiency of the computation. Therefore, this directly shows that we need to tune computational codes to the supercomputers with lower B/F ratio. In other words, contrivance of programing is required to realize better management of the data transfer.

In the current study, an in-house compressible fluid solver, LANS3D and its option of high-order compact differencing scheme are employed. The high-order compact scheme is one of the high-resolution global schemes and requires a lot of memory transfer. Therefore, tuning it for a supercomputer with a distributed memory architecture with many nodes is difficult even though its high resolution and high formal order of accuracy are attractive. In this paper, our efforts to improve the efficiency of the computational fluid dynamics code based on the compact scheme are presented. The code is partially rewritten and its speed and time budget are measured to identify what type of the tuning should be taken generally for the tuning of the compact scheme. Lastly, the effectiveness of the scalar tuning presented in current study on one real problem is demonstrated.

2. Research significance

Research significance of this study is offering scalar tuning of a computational fluid dynamic code using a compact differencing scheme though the detailed analyses of computational time on each step. The parallel efficiency of the code is not main obstacle to the speed-up of the computational code. Therefore, the way tuning a compact-scheme-based fluid solver presented in this study can be applied for other compact-scheme-based fluid solvers.

3. Brief description of computational code

As noted in the previous section, we use an in-house compressible flow solver named "LANS3D." This code was originally developed by Fujii and Obayashi [2] and the first version of the code supported the monotonically upstream scheme for conservation law scheme [3]. Then, the compact differencing [4] option was implemented by Arasawa et al. [5] and this option was validated through several computations including the transitional boundary layer problems [5, 6]. Also, we have a different option of the weighted compact nonlinear scheme, which is for shock-containing flows [7, 8]. For temporal discretization of any options in the code, the alternate-directional-implicit symmetric-Gauss-Seidel (ADI-SGS) scheme [9, 10] is adopted to converge the Euler implicit scheme or the backward second order finite differencing scheme.

The target flow in our project is subsonic, and in this study the compact differencing option is adopted for the scalar tuning to the "K computer." The K computer is a supercomputer recently developed in Japan and based on the distributed memory architecture with over 80,000 computer nodes [1]. Part of subroutines related to the compact scheme is originally written for the vector supercomputers such as Fujitsu VPP800 and NEC SX6 and SX9 that have larger B/F ratios than that of the K computer. Parallel computation has been performed using message passing interfaces (MPI) with which the sufficient parallel efficiency of weak scaling is obtained as shown in Table 1. Here, weak scaling is defined as:

$$\alpha_{\mathrm{weak}}=(\mathrm{FLOPS}_N/\mathrm{FLOPS}_M)/(N/M),$$

where N and M denotes number of computer nodes used for measurement computations and reference computations, FLOPS denotes floating-point operations per seconds and its subscript denotes number of computer nodes used for the computation. Note that the size of a problem per the cores is same as that utilized for the scalar tuning that will be discussed in the next section.

TABLE1: RESULTS OF THE WEAK SCALING OF LANS3D

Number of computer nodes for measurement (N)	Number of nodes for reference ($2M \leq N$)	Weak scalability (α_{weak})
2,000 (16,000 cores)	1,000	0.952
3,840 (30,720 cores)	1,920	0.972
12,288 (98,304 cores)	3,072	0.960
24,576 (196,608 cores)	12,288	0.967

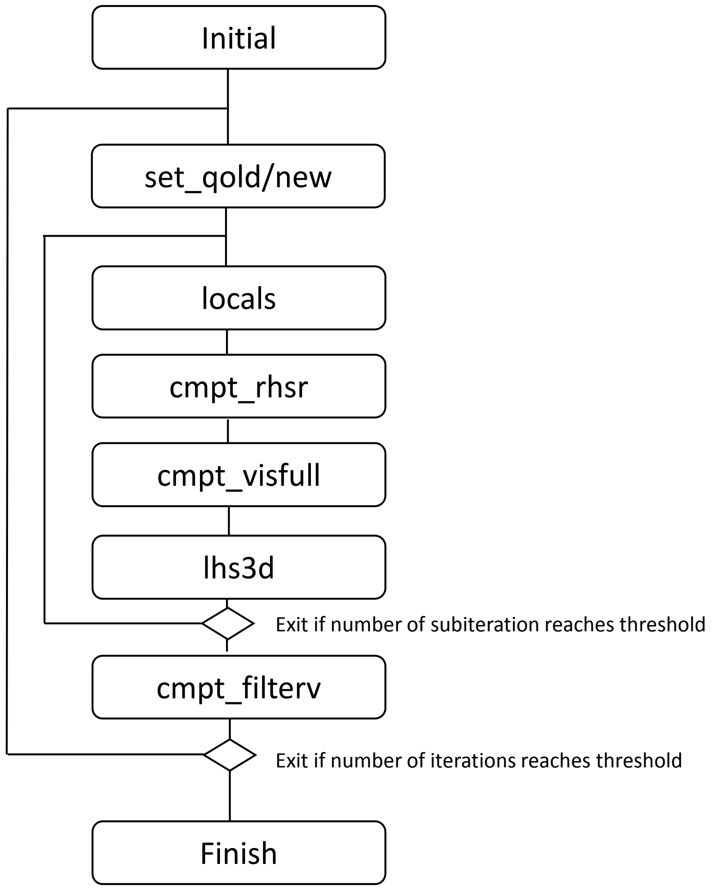
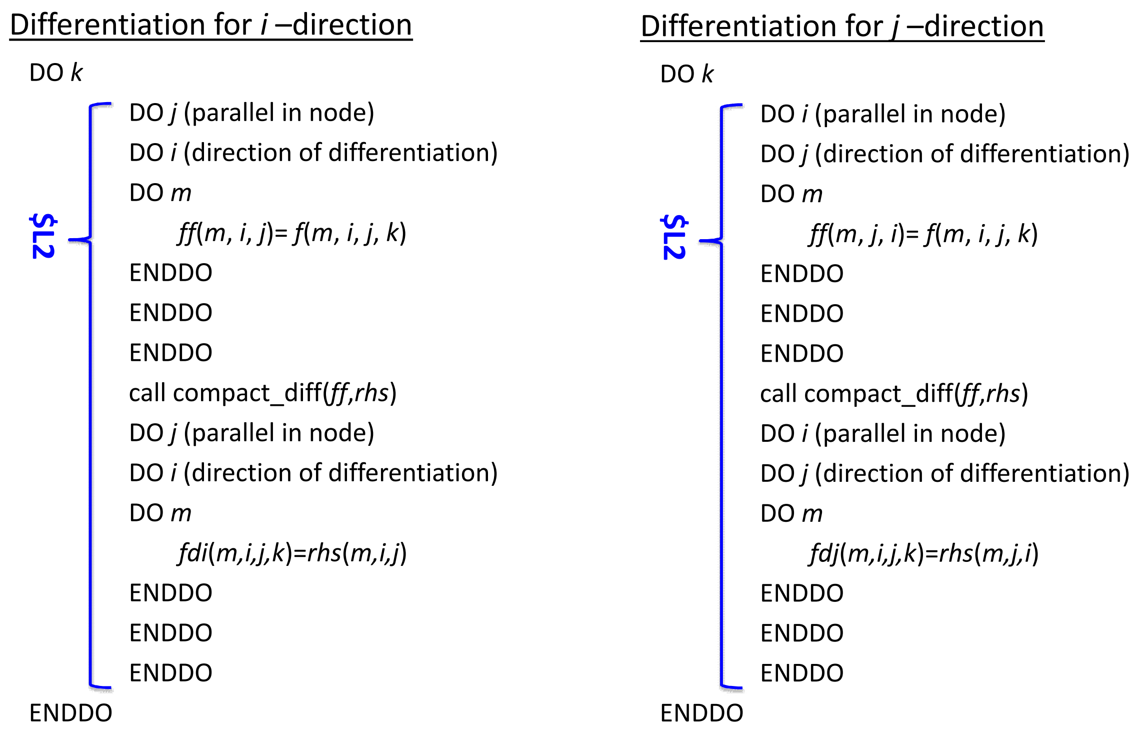

Initial
set_qold/new
locals
cmpt_rhsr
cmpt_visfull
lhs3d
Exit if number of subiteration reaches threshold
cmpt_filterv
Exit if number of iterations reaches threshold
Finish

Figure 1. Flowchart of computation with the compact scheme option in LANS3D

Figure 1 explains the flowchart of computation with the compact scheme options in LANS3D. There are several subroutines: *set_qold/new, locals, cmpt_rhsr, cmpt_visfull, lhs3d*, and *cmpt_filterv*. Their roles in the code are summarized in Table 2. Figure 2 shows pseudo codes that describe the way to implement compact differencing *i*- and *j*-direction in the original LANS3D. Here, *f* is *m*-component vector function differenced in the subroutine, *fdi* and *fdj* are differenced vectors in *i* and *j* direction, respectively. Indices *i, j,* and *k* are used for expressing a grid point in three-dimensional space, and an array is created in the order of (*m,i,j,k*). The temporal array, namely *ff* and *rhs*, are used for simplicity and readability, and a subroutine "compact_diff" corresponds to compact differencing part.

TABLE 2: MAIN SUBROUTINES IN LANS3D

Name of subroutines	Role	Reference
initial	Read grid file, set metrics and initial flow condition	For metric evaluation, see references [11-13]
set_qold/new	Replace the previous time step solution vectors or new time step solution vectors	
locals	Calculate the primitive variables	
cmpt_rhsr	Compute convection terms by 6th order compact scheme	Reference [4]
cmpt_visfull	Compute viscose terms by 6th order compact scheme	Reference [4]
cmpt_filterv	Apply the 10th order filter to the conservative variables	Reference [14]
lhs3d	Implicit time integration with ADI-SGS scheme	Reference [8,9]



Differentiation for *i* –direction

```
DO k
    DO j (parallel in node)          ┐
    DO i (direction of differentiation)
    DO m
        ff(m, i, j)= f(m, i, j, k)
    ENDDO
    ENDDO
    ENDDO
    call compact_diff(ff,rhs)          $L2
    DO j (parallel in node)
    DO i (direction of differentiation)
    DO m
        fdi(m,i,j,k)=rhs(m,i,j)
    ENDDO
    ENDDO
    ENDDO                            ┘
ENDDO
```

(a)

Differentiation for *j* –direction

```
DO k
    DO i (parallel in node)          ┐
    DO j (direction of differentiation)
    DO m
        ff(m, j, i)= f(m, i, j, k)
    ENDDO
    ENDDO
    ENDDO
    call compact_diff(ff,rhs)          $L2
    DO i (parallel in node)
    DO j (direction of differentiation)
    DO m
        fdj(m,i,j,k)=rhs(m,j,i)
    ENDDO
    ENDDO
    ENDDO                            ┘
ENDDO
```

(b)

Figure 2. Pseudo codes which mimic the compact scheme in original LANS3D

TABLE 3: DETAILS OF THE COMPUTER NODE OF THE K COMPUTER

Devices	
Processing unit	SPARC64^{TM} VIIIfx
	8 cores
	2.0 GHz
	Shared 6MB L2 cache
	128GFLOPS/Processing unit
	Water cooling system
Memory	DDR3 SDRAM
	16GB

4. Platform of a supercomputer with a distributed memory architecture

As previously mentioned, we utilize the K computer that is one of the peta-scale supercomputers and based on the distributed memory architecture with over 80,000 computer nodes [1]. The basic information of computer node is given in Table 3. In this paper, we focus on the scalar tuning of LANS3D that has the great impact on the speed-up of the computational code.

5. Discussion of results

5.1 Case description for the test of scalar tuning

A regular structure grid is adopted for a test case. The number of computational grid points is 64×64×64. The freestream condition is imposed on all the grid points as an initial condition.

5.2 *Scalar tuning of LANS3D*

Table 4 and Figure 3 present results of scalar tuning of the test case with the original and tuned LANS3D. It is noted that time cost and floating-point operations per seconds (FLOPS) are measured by a timer function and a hardware counter information (PA) provided by the K computer, respectively, and measurements are performed for subroutine by subroutine.

TABLE4: RESULTS OF SCALAR TUNING

	Time cost		Speedup	The ratio of actual FLOPS to peak FLOPS	
	Before tuning [sec.]	After tuning [sec.]		Before tuning [%]	After tuning [%]
Program					
LANS3D	119.89	55.14	2.17	4.47	9.65
Subroutines					
cmpt_rhsr	39.36	7.95	4.95	2.94	9.31
cmpt_visfull	25.10	18.31	1.37	6.93	9.66
cmpt_filterv	10.70	8.34	1.28	8.40	10.91
lhs3d	33.05	9.27	3.57	1.32	3.41
set_qold/qnew	3.77	1.42	2.65	0	0
locals	4.11	3.37	1.22	9.34	12.08

Following two points are identified from the results of the original LANS3D that need the scalar tuning: i) waiting time for synchronization (see the purple region in bars shown in Figure 3) and ii) waiting time for floating-point load memory and cache access (see the red and dark-pink regions in bars shown in Figure 3). We conduct several scalar tunings for main subroutines in LAN3D as shown in Tables 1 and 3 in order to improve a performance of LANS3D in the K computer in terms of a ratio of actual FLOPS to peak FLOPS and computational time. Detailed discussion regarding the results of the scalar tuning is given later.

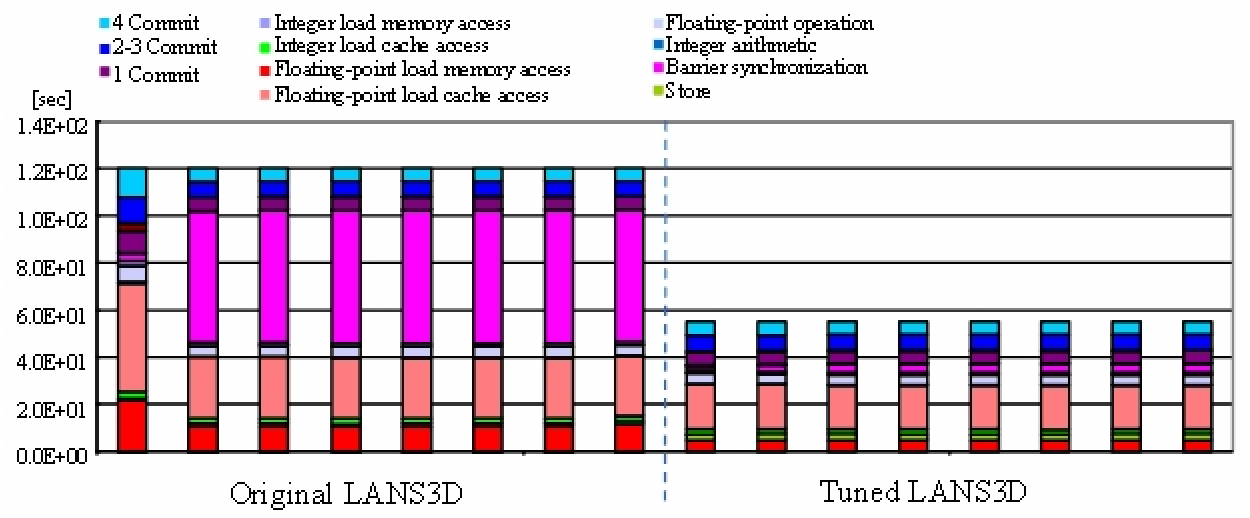


Figure 3. PA plot of original and tuned LANS3D

As seen in Table 4 and Figure 3, it is clearly shown that the scalar tuning performed in current study improves the performance of LANS3D. Specifically, total computational time significantly decreases (from 119.89 seconds down to 55.14 seconds) and the ratio of actual FLOPS to peak FLOPS remarkably improves (from 4.47% up to 9.65%). Additionally, we have theoretical assessments for a subroutine named *cmpt_triv4* that is frequently called in compact-scheme-related subroutines in LANS3D to estimate maximum ratio of actual FLOPS to peak FLOPS. The estimated maximum ratio of actual FLOPS to peak FLOPS is 9.6 % for this subroutine and this value is close to the result of tuned LANS3D.

The possible explanations for above improvement of the performance of LANS3D due to scalar tuning are listed as below:

1) Adjustment of load balancing by changing the do-loop that thread parallelization applies.
2) Reduction of overhead due to creating new threads between two do-loops, improvement in use of cache, and reduction in a number of sliding access by the fact that each thread solves *i* and *j* direction together (combining do-loops) and then *k* direction.

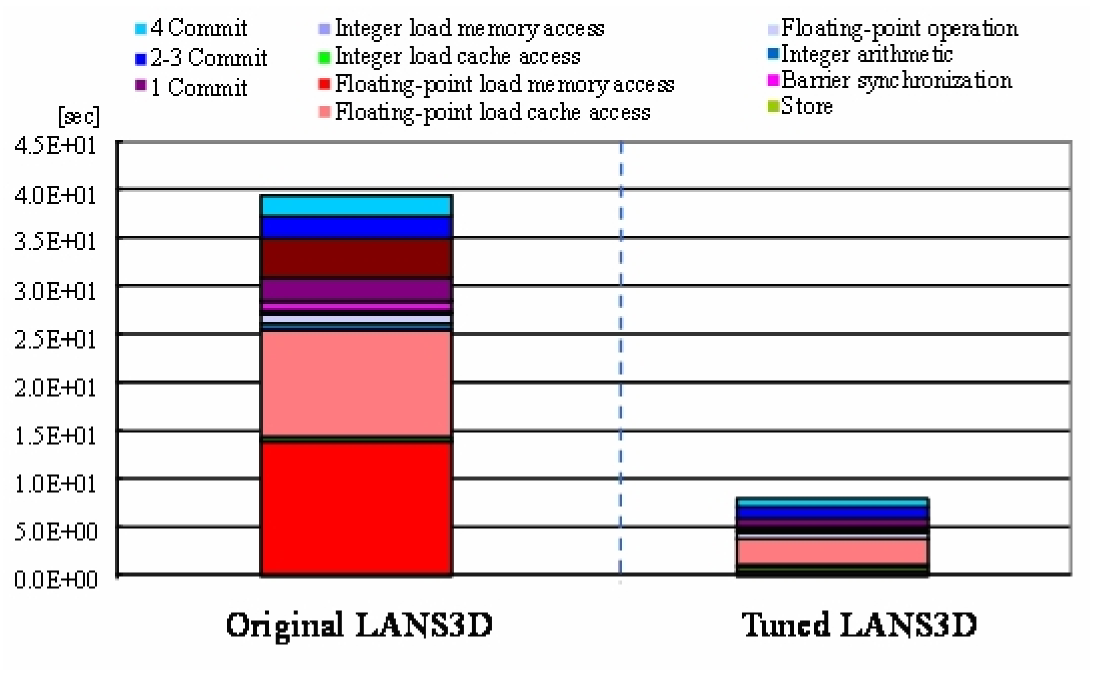


Figure 4. PA plot of first thread for *cmpt_rhsr* in the original and tuned LANS3D

Detail description of the scalar tuning and source of the issues for subroutines related to compact scheme (i.e. *cmpt_rhsr*, *cmpt_visfull*, and *cmpt_filterv*) is given as follows:

cmpt_rhsr:

As seen in Figure 4, waiting time for floating-point load cache (the dark-pink region in a bar in Figure 4) and memory access (the red region in a bar in Figure 4) dominates the computational time of this subroutine in the original LANS3D. Since the original LANS3D has been optimized and tuned for vector supercomputers, the implementation of the original LANS3D causes frequent discontinuous-array accesses in supercomputers with the distributed memory architecture. In order to decrease the cost of floating-point load cache and memory accesses, we combine couple of do-loops in this subroutine. The modified structure of this subroutine is similar to that shown in Figure 5 in which the tuned pseudo code for the same computation shown in Figure 2. In Figure 5, *ffi*, *ffj*, *rhsi*, and *rhsj* are temporal array introduced for simplicity and readability. This tuning results in significant reduction of a number of \$L1 and \$L2 miss (see a bar of right side in Figure 4). Especially, waiting time for floating-point memory access (the red region of the bar in left side in Figure 4) becomes invisible range. Hence almost five times speed-up is attained.

Differentiation for combined *i&j* –direction

```
DO k
  $L2
      DO j
      DO i
      DO m
          ffi(m, j, i)= f(m, i, j, k)
          ffj(m, j, i)= f(m, i, j, k)
      ENDDO
      ENDDO
      ENDDO
      call compact_diff(ffi,rhsi)
      call compact_diff(ffj,rhsj)
      DO j
      DO i
      DO m
          fdi(m,i,j,k)=rhsi(m,i,j)
          fdj(m,i,j,k)=rhsj(m,j,i)
      ENDDO
      ENDDO
      ENDDO
ENDDO
```

Figure 5. Pseudo code which mimics the compact scheme in tuned LANS3D

cmpt_visfull:

Waiting time for floating-point load cache access dominates a computational time of this subroutine in the original one (see the dark-pink region in a bar in Figure 6). Six do-loops (i.e. similar to Figure 5) are combined so as to reduce amount of memory transfer. Although the combining of do-loops decreases a number of $L1 and $L2 misses, its speedup is less than that of *cmpt_rshr*. This suggests that combining of do-loops works well in the case that waiting time for floating-point memory access dominates a computational time.

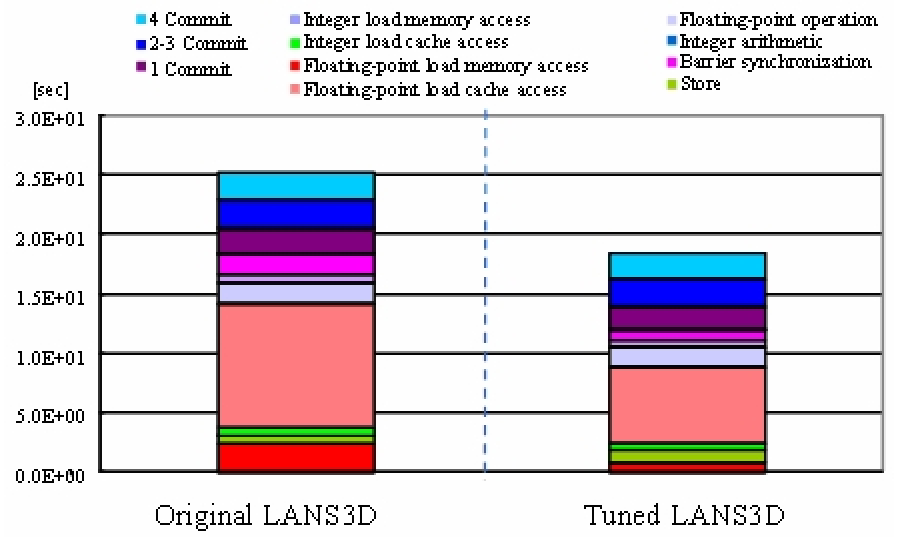


Figure 6. PA plot of first thread for *cmpt_visfull* in the original and tuned LANS3D

cmpt_filterv:
As similar to *cmpt_visfull*, waiting time for floating-point load cache access dominates a computational time of this subroutine in the original one. Couple do-loops are unified for designing a reduction of amount of memory access and transfer. This also results in reduction of a number of $L1 and $L2 misses.

Furthermore, detail description of scalar tuning and source of the issues for subroutine that is irrelevant to the compact differencing scheme is presented as below:

set_qnew/qold:
As explained in Table 2, *set_qnew/qold* is designed to replace the previous time step solution vectors or the new time step solution vectors. It means that no computations are required. For this subroutine, waiting time for floating-point load cache access dominates a computational time. After we modify the way to replace the solution vector from using an assignment statement expression to using pointers, this leads to decrease of a number of $L1 and $L2 misses.

locals:
As similar to *cmpt_visfull*, *cmpt_flterv*, and *set_qnew/qold*, waiting time for floating-point load cache access dominates a computational time of this subroutine in the original one. We optimize memory transfer and then it reduces a number of $L1 and $L2 misses.

lhs3d:
As explained in Table 2, *lhs3d* is designed to perform implicit time integration with ADI-SGS scheme. Waiting time for floating-point load cache access dominates a computational time in the original one. Moreover, there is a problem in load balancing among threads due to overhead of thread parallelization. We apply thread parallelization to outer do-loop so as to improve load balancing and combine couple do-loops to localize memory access. Those tunings adjust load balancing as well as reduce a number of $L1 and $L2 misses.

5.3 Example: a large-eddy simulation around NACA0015 wing at a Reynolds number 63,000

The effectiveness of scalar tuning presented in current study on one real problem is demonstrated. A large-eddy simulation (LES) around a NACA0015 wing at a stall angle (i.e. 16 deg.) and a Reynolds number *Re* of 63,000 is performed using 380 computer nodes (3,040 cores) of the K computer.

Three-dimensional compressible Navier-Stokes equations are employed as the governing equations. These equations are solved in the generalized curvilinear coordinates. The spatial derivatives of the convective and viscous terms, metrics, and Jacobian are evaluated by a sixth-order compact scheme [3]. Near the boundary, second-order explicit difference schemes are used. Tenth-order filtering [14] is used with a filtering coefficient of 0.495. For time integration, ADI-SGS methods [9, 10] are used. To ensure time accuracy, a backward second-order difference formula is used for time integration, and five sub-iterations are adopted. The non-dimensional computational time step is 0.0002 that corresponds to maximum Courant-Friedrichs-Lewy number of approximately 2.0. In a standard LES approach, additional stress and heat flux terms are appended, but in an implicit LES (ILES) approach [17] they are not appended. In this research, ILES is employed, and a high-order, low-pass filter selectively damps only poorly resolved high-frequency waves. This filtering regularization procedure provides an alternative method to the use of standard sub-grid-scale models. At the outflow boundary, all variables are extrapolated from one point in front of the outflow boundary. For the airfoil surface, no-slip and adiabatic-wall conditions are adopted. A periodic boundary condition is applied to the boundaries in the spanwise direction.

Total number of grid points is approximately 20,000,000. Further information can be found in the references [15, 16].

Table 4 proves that the scalar tuning works well for a real problem and remarkably saves the computational time in comparison with that computed by the original LANS3D. This makes one simulation twice more efficient, and leads to over-160-case parametric study of ILES [16] as shown in Figure 7.

TABLE4: COMPARISON OF A COMPUTATIONAL TIME BETWEEN THE TUNED AND THE ORIGINAL LANS3D FOR AN ILES AROUND NACA015 AIRFOIL

Airfoil	Number of computer nodes used	Time cost spent to compute 1 flow through		Speedup
		Before tuning [sec.]	After tuning [sec.]	
NACA0015	380	8,520	3,180	2.68

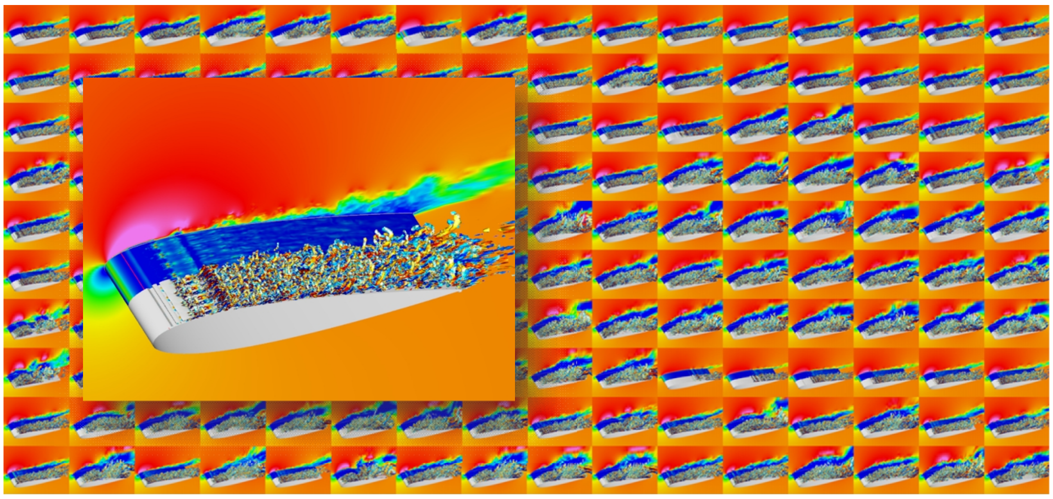


Figure 7. Overview of over-160-case parametric study of ILES [16]. Iso surface indicates second invariant of velocity gradient tensor and it is colored by chord-direction vorticity. Contours near the wing surface in the backside and the plane denote chord-direction velocity.

6. Summary

In this study, the scalar tuning of a compressible fluid dynamics solver using the compact scheme for the supercomputer with the distributed memory architecture with many computer nodes was presented. The K computer was used. Acceleration of computation and improvement maximum ratio of actual FLOPS to peak FLOPS were achieved by: i) improvement of load balancing by changing the do-loop that thread parallelization applies; and ii) reduction of overhead due to creating new threads between two do-loops, improvement in use of cache, and reduction in a number of sliding access by the fact that each thread solves *i* and *j* direction together and then *k* direction. The tuned code was twice faster than the original code and its efficiency is close to 10% that is theoretical limit. Using the tuned code, over-160-case parametric study was successfully conducted with approximately half of computational cost in comparison with that of the original code. We believe that the way tuning compact-scheme-based fluid solver in this study can be also applied for other compact-scheme-based fluid solvers.

Acknowledgements

The computations are performed using the K computer at the RIKEN Advanced Institute for Computation Science, Japan (Proposal number: hp120296 and hp130001). This research is supported by Strategic Programs for Innovative Research (SPIRE) of High Performance Computing

Initiative (HPCI). M. Sato, A. Yakeno, K. Okada, and K. Asada conducted the part of computations in the parametric study. The authors are grateful to their help.

References

[1] Miyazaki, H., Kusano, Y., Okano, H., Nakada, T., Seki, K., Shimizu, T., Shinjo, N., Shoji, F., Uno, A., and Kurokawa, M., *K computer: 8.162 PetaFLOPS Massively Parallel Scalar Supercomputer Built with over 548,000 Cores*. in 2012 *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, p. 192-194. 2012. San Francisco, CA.

[2] Fujii, K. and Obayashi, S. *High-resolution Upwind Scheme for Vortical-flow Simulations*. Journal of Aircraft, 1989. **26**: p. 1123-1130.

[3] van Leer, B. *Towards the Ultimate Conservation Difference Scheme V. A Second-order Sequel to Godunov's Methods*. Journal of Computational Physics, 1979. **32**: p. 101-136.

[4] Lele, S. K. *Compact Finite Difference Schemes with Spectral-like Resolution*. Journal of Computational Physics, 1992. **103**: p. 16-42.

[5] Arasawa, T., Fujii, K., and Miyaji, K. *High-order Compact Difference Scheme Applied to Double-delta Wing Vortical Flows*. Journal of Aircraft, 2004. **41**: p. 953-957.

[6] Kawai, S. and Fujii, K. *Compact Scheme with Filtering for Large-eddy Simulation of Transitional Boundary Layer*. AIAA Journal, 2008. **46**: p. 690-700.

[7] Nonomura, T., Iizuka, N., and Fujii, K. *Effects of Difference Scheme Type in High-order Weighted Compact Nonlinear Schemes*. Journal of Computational Physics, 2009. **228**: p. 3533-3539.

[8] Nonomura, T., Iizuka, N., and Fujii, K. *Freestream and Vortex Preservation Properties of High-order WENO and WCNS on Curvilinear Grids*. Computers and Fluids, 2010. **39**: p. 197-214.

[9] Iizuka, N., Ph.D thesis at the University of Tokyo, March, 2006.

[10] Nihsida, H. and Nonomura, T. *ADI-SGS Scheme on Ideal Magnetohydrodynamics*. Journal of Computational Physics, 2011. **228**: p. 3182-3188.

[11] Deng, X. Mao, M., Tu, G., and Zhang, H. *Geometric Conservation Law and Applications to High-order Finite Difference Schemes with Stationary Grids*. Journal of Computational Physics, 2011. **230**: p.1100-1115.

[12] Abe, Y., Iizuka, N., Nonomura, T., and Fujii, K. *Conservative Metric Evaluation for High-order Finite Difference Schemes with the GCL Identities on moving and deforming grids*. Journal of Computational Physics, 2013. **232**: p. 14-21.

[13] Abe, Y., Iizuka, N., Nonomura, T., and Fujii, K. *Geometric Interpretation and Spatial Symmetry Property of Metrics in the Conservative Form for High-order Finite-difference Schemes on Moving and Deforming Grids*, submitted to Journal of Computational Physics.

[14] Gaitonde, D. V. and Visbal, M. R. *Pade-type Higher-order Boundary Filters for the Navier-Stokes Equations*. AIAA Journal, 2000. **38**: p. 2103-2112.

[15] Nonomura, T., Aono, H., Sato, M., Yakeno, A., Okada, K., Abe, Y., and Fujii, K. *Control Mechanism of Plasma Actuator for Separated Flow around NACA0015 at Reynolds Number 63,000 -Separation Bubble Related Mechanisms*. in 51st AIAA Aerospace Science Meeting including the New Horizons Forum and Aerospace Exposition, AIAA paper 2013-0853. 2013. Grapevine, TX.

[16] Sato, M., Okada, K., Nonomura, T., Aono, H., Yakeno, A., Asada, K., Abe, Y., and Fujii, K. *Massive Parametric Study by LES on Separated-Flow Control around Airfoil using DBD Plasma Actuator at Reynolds Number 63,000*. in 43rd Fluid Dynamic Conference, AIAA paper 2013-2750. 2013. San Diego, CA.

[17] Grinstein, F.F., Margolin, L.G., and Rider, W. Implicit Large Eddy Simulation. Cambridge University Press. NY, USA. 2007.