Performance Optimization of GeoFEM on Various Computer Architecture

Kazuo Minami [1]

(1) Department of Computational Earth Sciences, Research Organization for Information Science and Technology (RIST), Tokyo, Japan (e-mail: nakajima@tokyo.rist.or.jp, phone: +81-3-3436-5271, fax: +81-3436-5274)

Abstract

We have a prospect that Geofem get good parallel performance on various computer architecture by the research which has been made in GeoFEM team. In this research, we focus a target to performance with a single processor, and common data structure / coding manner to make GeoFEM running high performance on various computer architecture was researched.

<Test coding for Solver Part of the structure and the fluid analysis code>
Data structure and coding manner was evaluated on scalar/vector/pseudo vector architecture. A new data structure and a new direct access manner was introduced in the fluid analysis solver. As a result, 21% performance for peak performance was obtained on pseudo vector architecture. And We got as good performance as the pseudo-vector execution performance in scalar execution. 25% performance for peak performance was obtained on vector architecture.
A new direct access manner was introduced in the structure code solver. As a result, 27% performance for peak performance was obtained on pseudo vector architecture. 34% performance for peak performance was obtained on vector architecture.

<Test Code for Matrix Assemble Part of the structure analysis code>
Coding of removing dependency was finished and performance was evaluated on vector/scalar machine. 736.8MFlops was obtained at matrix assembling process on SX-4. 900.7MFlops was obtained at whole test Code on SX-4, and 2.06GFlops was obtained on VPP5000 (peak : 9.6GFlops). 124MFlops was obtained at matrix assembling process on Alpha system (Alpha 21164 533MHz).

1. Introduction

The Science and Technology Agency, Japan has begun an Earth Simulator project from the fiscal year of 1997 for predicting various Earth phenomena through the simulation of virtual Earth placed in a supercomputer. The specific research topics of the project are as follows:
1) Development of a high performance massively parallel processing computer: 'Earth Simulator' (40 Tflops / Peak Performance, 10 TB Memory)
2) Modelling of atmospheric and oceanic field phenomena and high-resolution simulations
3) Modelling and simulation of solid earth field phenomena
4) Development of large-scale parallel software for the Earth Simulator.

GeoFEM deals with the topics 3) and 4) in the above, and is planned to develop the system in two phases as:
Phase I : GeoGEM/Tiger (1997-1998) : Multi-purpose parallel finite element software, which may be applied to various fields in engineering and sciences as well as becoming the basis for the solid earth simulator to be developed in Phase II.

Phase II : GeoFEM/Snake (1999-2001) : A software system optimized for 'Earth Simulator' and specialized for the simulation of solid earth phenomena such as mantle-core convection, buildup of tectonic stress, deformation of plates and seismic wave propagation.

Now, various computer architecture exist in the world. The Earth simulator is mainly target machine for GeoFEM, moreover GeoFEM is assumed to running on various computers in Multi-purpose parallel finite element software. It is the best for this assumption that "one codes" runs in high performance on various computer architecture, but generally, suitable programming is demanded from those architecture in order to get high execution performance on the architecture of each.
To change data structure and coding into every architecture boosts trouble of development and trouble of maintenance, and loses convenience of the users.

We are targeting GeoFEM which is finite element method simulation code for Unstructured grid, and we are researching common data structure / coding manner to make GeoFEM running high performance on various computer architecture. Both the parallel performance and the single processor performance have to be good, to get GeoFEM a good total performance.

In 1998, very large scale linear elastic problem was solved by GeoFEM on University Tokyo's Hitachi SR2201 using 1,000 processors[1]. In this research, GeoFEM has got at good parallel performance. Linear Solver of GeoFEM has attained good performance on SMP parallel computer in recent Nakejima's research[2]. We have a prospect that Geofem get good parallel performance on various computer architecture from these research. Therefore, in this research, we focus a target to performance with a single processor of GeoFEM, and we researched common data structure / coding manner to make GeoFEM running high performance on various computer architecture .

2. strategy

GeoFEM is constructed by the structure analysis code and the fluid analysis code. Each code can be roughly divided into two parts. First part, is calculates the coefficient of the composition equation (Matrix Assemble Part). Second part, is solves the system of linear equation (Solver Part).
Matrix having 4 area as shown in below is got (Table 1). Our research is approached to 4 areas depicting below in order to achieve a purpose.

	Matrix Assemble part	Solver part
structure analysis code	1	2
fluid analysis code	3	4

Table 1 Target Area in This Research

GeoFEM Test Code which has similar GeoFEM Matrix Assemble process and GeoFEM Solver (about area 1 and area 2) have been made. Real GeoFEM code have been used about area 3 and area 4. In the structure analysis code, 85% in all operation count is occupied by operation count of solver part. The remaining 15% are occupied by operation count of Matrix Assemble Part. In the fluid analysis code, 91% in all CPUTIME is occupied by solver part. The remaining 9% is occupied by Matrix Assemble Part.

We decided advancing the research by the following priority sequence because of Solver Part is important in both the structure analysis code and the fluid analysis code.
-Solver Part of the structure analysis code and the fluid analysis code
-Matrix Assemble Part of the structure analysis code

-Matrix Assemble Part of the fluid analysis code

The progress of the each researching items is shown below.

-Solver Part of the structure analysis code and the fluid analysis code

Data structure and coding manner was evaluated on the scalar/vector/pseudo vector architecture, and this result will be shown in this paper.

-Matrix Assemble Part of the structure analysis code

A coding of removing dependency was finished and performance was evaluated on vector/scalar machine. This result is shown in this paper. But evaluation of data structure and coding manner wasn't finished. It is next problem.

-Matrix Assemble Part of the fluid analysis code

We do not start it yet and it is next problem.

3. Evaluation Result of GeoFEM Linear Solver

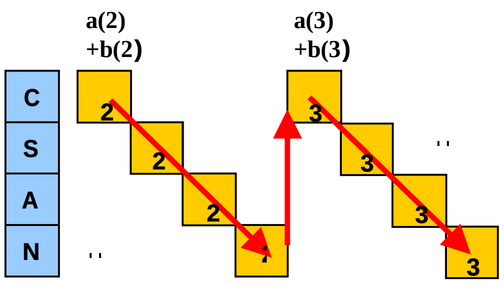
3.1 Dependency of Data

Data dependency gives big influence when we think about performance on a single processor. Now, almost processor is pipelined, therefore efficiently using pipeline is important things for both architecture of the scalar machine and the vector machine.
I exemplify good use of efficiency of a pipeline. An addition pipeline consists of four following stage as example.

C: Compare of exponent part
S: Shift of Mantissa part
A: Addition
N: Normalization

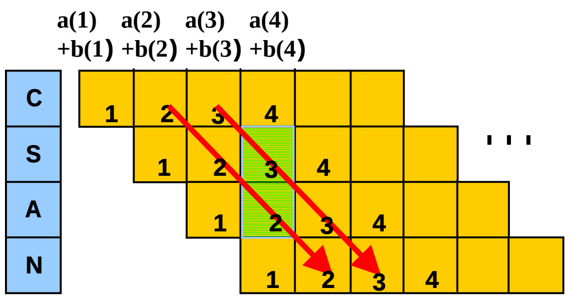
Movement of a pipeline is shown in a Figure below when there are dependency between array elements. Pipelines can not process concurrently among array element, when there are dependency between array elements.

```
do i = 2,n
  a(i)= a(i-1) + b(i)
enddo
```



Movement of a pipeline is shown in a Figure below when there are no dependency between array elements.

```
do i = 1,n
  a(i)= a(i) + b(i)
enddo
```



The pipelines can processes concurrently among array element, when there are no dependency between array elements. Removing dependency between array elements is important to use of efficiency of a pipeline.

3.2 Outline of The GeoFEM Linear Solver

The GeoFEM Linear Solver is parallel iterative solver having characteristic as below[2].
-Localized Preconditioning[2]
-Distributed data structures by node-based partitioning with overlapping elements at partition interfaces[2]
-Cyclic Multicolor on Hyperplane/RCM[2]
-DJDS(Descending-order Jagged Diadonal Storage) re-ordering[2] [3]
In case of Solver of the finite element method code using Unstructured Grid, usually dependency of loop data is generated in main calculating loop. Cyclic Multicolor Hyperplane/RCM Manner and DJDS Re-Ordering Manner were introduced in GeoFEM to remove dependency of loop data and to run with high performance on the vector machine.

3.3 Cost Analysis

Result of the cost analysis for solver part / Matrix Assemble Part of both the structure analysis and fluid analysis is shown in Table 2.
Solver Part is important in both the structure analysis code and the fluid analysis code.

	Fluid Analysis (CPUTIME)	Structure Analysis (operation count)
Solver Part	91.2%	84.7%
Matrix Assemble Part	8.8%	15.3%

Table 2 Result of Cost Analysis

In Solver Part of the fluid analysis code, cost of the CPUTIME (96.8%) is concentrated to 4 processes depicted in Table 3. Trend of this result concentrating to 4 processes is applied to the Structure Analysis Code too.

process(1)	forward substitution	14.6	
process(2)	backward substitution	14.5	
process(3)	Matrix-Vector Product(lower)	12.3	
process(4)	Matrix-Vector Product(upper)	12.3	
	others		1.8
		53.7	1.8

Table 3 Cost Analysis of the Fluid Analysis Solver

3.4 Model Coding

4 processes being depicted in chapter 3.3 have almost same coding. Example of coding for the Fluid Analysis Code is shown in Fig.1.
Example of coding for the Structure Analysis Code is shown in Fig.2.
In Fig.1 and Fig.2, array:**W** having index:kk in right hand side of equation in innermost loop is accessed by indirect access manner.

In Fig.1, arrays except W having index:kk is accessed by direct access manner.This coding is Matrix-Vector product among Matrix having 1*1 elements and Vector having 1 elements.
In Fig.2, arrays except W having index:kk is accessed by constant intervals access manner.This coding is Matrix-Vector product among Matrix having 3*3 elements and Vector having 3 elements.
Model coding of Fig.3 was made by modeling from coding of Fig.1.
Model coding of Fig.4 was made by modeling from coding of Fig.2.
Access manner of each based coding is modeled as both coding Fig.3 and Fig.4.

3.5 Pseudo Vector Architecture

In this study, We used Hitachi SR8000 system that is the 128-node system in the Computing Center, University of Tokyo. SR8000 system is a machine of pseudo vector architecture, and SR8000 system has function of scalar machine as is Software-Pipelining. Software-Pipelining make object code to be modeled at Fig.5(b). A object code of Fig.5(a) depict not being Software-Pipelining. Effective pipelining process for element of array will be possible by Software-Pipelining after removing dependency between array elements. SR8000 system has function of pseudo-vector. The function of pseudo-vector has function of pre-load and pre-fetch(Fig.6).

In SR8000 system, each processor has 128 floating-registers to be possible to use as Vector-Resister. The function of pre-load loads directly data from memory to floating-registers without using cache. Load latency can be hidden by loading the data before using loaded data. The function of pre-load will usually be applied for indirect accessed array. The function of pre-fetch fetches data from memory to cache before using data and Load latency can be hidden. The function of pre-fetch will usually be applied for direct accessed array.

```
  do iv= 1, NVECT
   iv0= IVECT(iv-1)
   do  j= 1, NLhyp(iv)
     iS= INL(NL*(iv-1)+j-1)
     iE= INL(NL*(iv-1)+j  )
     do i= iv0+1, iv0+iE-iS
        k= i+iS - iv0
       kk= IAL(k)
       W(i,Z)= W(i,Z) - AL(k) * W(kk,Z)
     enddo
   enddo
   ........
enddo
```

Fig.1 Coding of forward substitution for the Fluid Analysis Code

```
   do iv= 1, NVECT
   iv0= IVECT(iv-1)
   do  j= 1, NLhyp(iv)
     iS= INL(NL*(iv-1)+j-1)
     iE= INL(NL*(iv-1)+j  )
     do i= iv0+1, iv0+iE-iS
        k= i+iS - iv0
       kk= IAL(k)
       Zm2= W(3*kk-2,Z)
       Zm1= W(3*kk-1,Z)
       Zm0= W(3*kk  ,Z)
       W(3*i-2,Z)= W(3*i-2,Z) - AL(9*k-8)*Zm2 - AL(9*k-7)*Zm1     &
&                                             - AL(9*k-6)*Zm0
       W(3*i-1,Z)= W(3*i-1,Z) - AL(9*k-5)*Zm2 - AL(9*k-4)*Zm1     &
&                                             - AL(9*k-3)*Zm0
       W(3*i  ,Z)= W(3*i  ,Z) - AL(9*k-2)*Zm2 - AL(9*k-1)*Zm1     &
&                                             - AL(9*k  )*Zm0
     enddo
   enddo
   ...........
 enddo
```

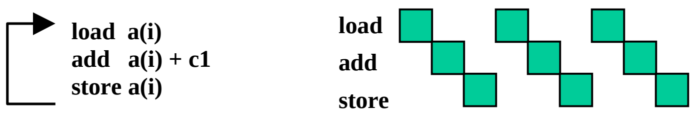
Fig.2 Coding of forward substitution for the Structure Analysis Code

```
 do j=1,1000
 do i=1,1000
   a(i) = a(i) + b(1000*(j-1)+i)*a(1000+N(i))
 enddo
 enddo
```

Fig.3 Model coding of solver for the Fluid Analysis Code

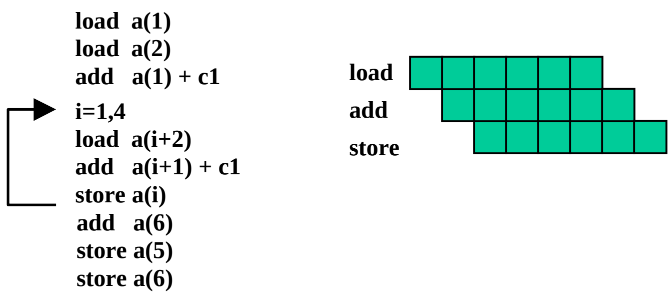
```
 do j=1,1000
 do i=1,1000
       k = i
      kk = N0(k)
     Zm2 = a(3*kk-2)
     Zm1 = a(3*kk-1)
     Zm0 = a(3*kk  )
     a(3*i-2)= a(3*i-2) - b(9*k-8)*Zm2 - b(9*k-7)*Zm1     &
&                                      - b(9*k-6)*Zm0
     a(3*i-1)= a(3*i-1) - b(9*k-5)*Zm2 - b(9*k-4)*Zm1     &
&                                      - b(9*k-3)*Zm0
     a(3*i  )= a(3*i  ) - b(9*k-2)*Zm2 - b(9*k-1)*Zm1     &
&                                      - b(9*k  )*Zm0
 enddo
 enddo
```

Fig.4 Model coding of solver for the Structure Analysis Code



```
load  a(i)
add   a(i) + c1
store a(i)
```

Fig.5(a) Case of not Software Pipelining



```
load  a(1)
load  a(2)
add   a(1) + c1
i=1,4
load  a(i+2)
add   a(i+1) + c1
store a(i)
add   a(6)
store a(5)
store a(6)
```

Fig.5(b) Case of   Software Pipelining

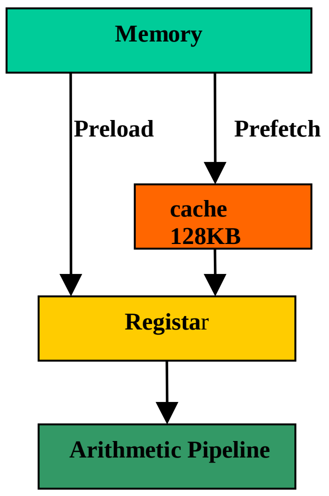


Fig.6 Preload and Prefetch

3.6 Performance Factor

Performance of Model-Coding being depicted in Fig.2 is 68MFlops and this performance is corresponding to performance of real coding almost. Next, coding of Fig.7 was made from coding of Fig2 to change a accessing manner of array:**a** from indirect access manner to direct access manner. Performance improved to 520MFlops in Fig.7 by changing a accessing manner.

```
do j=1,1000
do i=1,1000
  a(i) = a(i) + b(1000*(j-1)+i)*a(1000+i)
enddo
enddo
```

Fig.7 Model Coding of Dilect Access Manner

Performance factor improving from 68MFlops to 520MFlops was shown by compiler message etc. as below.
-outer loop expansion (8 times)
-To change a accessing manner of array:**a** to direct access manner
We will be got a more performance improvement by reducing load/store latency about outer loop expansion. Original coding in example is shown as below.

```
do i=1,1000
do j=1,1000
  y(i) = y(i) + a(i,j)*x(j)
enddo
enddo
```

Improved coding in example is shown as below.

```
do is=1,1000,10
do j=1,1000
  y(is  ) = y(is  ) + a(is  ,j)*x(j)
  y(is+1) = y(is+1) + a(is+1,j)*x(j)
  .......
  y(is+9) = y(is+9) + a(is+9,j)*x(j)
enddo
enddo
```

Original coding has 2 load operation and 2 floating operation, and a ratio of load/store operation and floating operation is 2/2. Improved coding has 11 load operation and 20 floating operation, and a ratio of load/store operation and floating operation is 11/20. We will be able to get good performance in case of loop having small load/store operation in comparison with floating operation.

Next, coding of Fig.8 was made from coding of Fig.7 by exchanging array:**a** and array:**b** in coding of Fig.7. Performance changed for the worse to 140MFlops from 520MFlops in Fig.8. A reason changing for the worse was shown by compiler message etc. It was a failure of software pipelining although the pseudo vector process was success.

We understood that the performance factor is 4 items discussing in this chapter, in addition to being pseudo vectorizetion explaining in chapter 3.5. The performance factors are shown in below.

(1) The loop is to be pseudo vectorization.
(2) The loop is to be software pipelining.
(3) The loop is to be outer loop expansion.
(4) The loop has small load/store latency.
(5) The array in the loop is accessed at direct access manner.

3.7 Evaluation of Performance for Model loops of the Fluid Analysis Code

Both (1) and (2) of performance factor, explained in chapter 3.6, was implemented to the model coding of Fig.3 . A new model coding implementing both (3) and (4) of performance factor was made. This coding is implemented outer loop expansion. And, We exchanged data structure of array:**b** to get good effect of prefetch. A new data structure is shown in Fig.9.

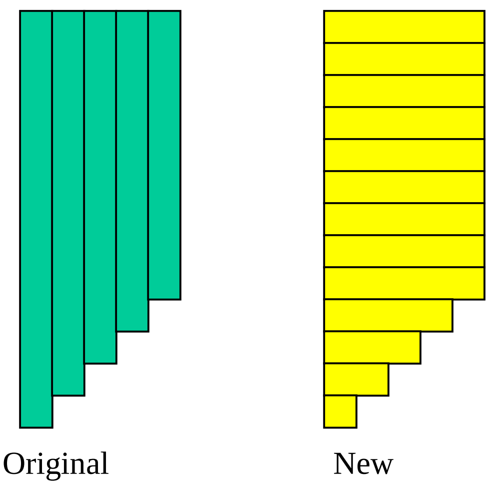


Fig.9 Data Structure of Original PDJDS/CM-RCM and New Data Structure[2]

A new model coding is shown in Fig.10.

```
      do j=1,1000,10
      do i=1,1000
        a(i) = a(i) + b(10*(i-1)+1)*a(N0(10*(i-1)+1)) &
     &              + b(10*(i-1)+2)*a(N0(10*(i-1)+2)) &
     &              + b(10*(i-1)+3)*a(N0(10*(i-1)+3)) &
     &              + b(10*(i-1)+4)*a(N0(10*(i-1)+4)) &
     &              + b(10*(i-1)+5)*a(N0(10*(i-1)+5)) &
     &              + b(10*(i-1)+6)*a(N0(10*(i-1)+6)) &
     &              + b(10*(i-1)+7)*a(N0(10*(i-1)+7)) &
     &              + b(10*(i-1)+8)*a(N0(10*(i-1)+8)) &
     &              + b(10*(i-1)+9)*a(N0(10*(i-1)+9)) &
     &              + b(10*(i-1)+10)*a(N0(10*(i-1)+10))
      enddo
      enddo
```

Fig.10 Model Coding of Outer Loop Expansion and Reducing Load/Store Latency

Model coding of Fig.3 has 4 load/store operation and 2 floating operation, and a ratio of load/store operation and floating operation is 4/2. Model coding Fig.10 has 22 load operation and 20 floating operation, and a ratio of load/store operation and floating operation is 22/20. Moreover, load/store operation is more smaller by using prefetch fetching many data in a one fetch operation. These fetch operation is possible, because element of array:**b** is referred at continuity order in model coding Fig.10. The performance improved to 174MFlops from 68MFlops in Fig.10. That result is depicted in Table 4[B].

A new model coding implementing (5) of performance factor explained in chapter 3.6 was made. Direct access manner for array:**a** was used in this new coding in addition to used performance factors in coding of Fig.10. As shown in chapter 3.3, cost of the solver part is concentrated to 4 processes depicted in below.

(a) forward substitution
(b) backward substitution
(c) Matrix-Vector Product(lower)
(d) Matrix-Vector Product(upper)

Same list array is used for accessing array:**a** in indirect access manner in both (a) and (c) processes. Same situation occurs on (b)(d) processes. That is to perform 4 processes by using 2 list array. Therefore, these 4 processes are performed at high performance by reordering array:**a** to 2 temporary array at using 2 list array. This new coding is shown in Fig.11.
Coding of Fig.11(a) is the reordering process for array:**a**, and coding of Fig.11(b) is the main calculating process. The main calculating processes are applied direct access manner.

The performance of 82Moperation/sec was got for reordering process, the performance of 547MFlops was got for main calculating process, and the total performance of 211Mflops was got. That result is depicted in Table 4[C].

```
      do j=1,1000,5
      do m=1,10
      do i=1,100
        kk = 500*(m-1)+5*i-5
        x1(kk+1) = a(N0(kk+1))
        x1(kk+2) = a(N0(kk+2))
        x1(kk+3) = a(N0(kk+3))
        x1(kk+4) = a(N0(kk+4))
        x1(kk+5) = a(N0(kk+5))
      enddo
      enddo
      enddo
```

Fig.11(a) Model Coding of Direct Access (Reordering Process)

```
      do j=1,1000,5
      do i=1,1000
        kk = 5*(i-1)
        kk1=kk+1
        kk2=kk+2
        kk3=kk+3
        kk4=kk+4
        kk5=kk+5
        a(i) = a(i) + b(kk1)*x1(kk1) &
     &              + b(kk2)*x1(kk2) &
     &              + b(kk3)*x1(kk3) &
     &              + b(kk4)*x1(kk4) &
     &              + b(kk5)*x1(kk5)
      enddo
      enddo
```

Fig.11(b) Model Coding of Direct Access (Main Calculating Process)

Each processor (not each node) has 1 Gflops peak performance in the SR8000 system. In previous discussion, 21% performance was got for peak performance on pseudo vector architecture.

Next, 3 cases depicted in below was performed to measuring performance on the scalar architecture. That result is depicted in Table 4.
-scalar execution with compiler option Oss on SR8000 system (Table 4[D])
-scalar execution with compiler option opt3 on SR8000 system (Table 4[E])
-scalar execution on Alpha system (Alpha 21164 533MHz) (Table 4[F])

Software pipelining was probably used in case[D], and was not probably used in both case[E] and case[F].   We could get as good performance as the pseudo-vector execution performance in scalar execution with software pipelining.

	[A]	[B]	[C]	[D]	[E]	[F]
	Fig.3	Fig.10	Fig.11	Fig.11	Fig.11	Fig.11
(1) pseudo vectorization	o	o	o	x	x	x
(2) software pipelining	o	o	o	o	x	x
(3) outer loop expansion	x	o	o	o	o	o
(4) load/store latency	x	o	o	o	o	o
(5) direct access	x	x	o	o	o	o
performance (MFlops)	68	174	211	225	63	109

[A] base model coding on SR8000
[B] pseudo vector with software pipelining on SR8000
[C] pseudo vector with software pipelining on SR8000
[D] scalar with software pipelining on SR8000
[E] scalar without software pipelining on SR8000
[F] scalar without software pipelining on Alpha system

Table 4 Performance for Model loops of the fluid analysis code on Pseudo Vector and Scalar Archi.

Next, model coding of both Fig.3 and Fig.11 was performed on Fujitsu VPP5000 being vector machine that each processor has 9.6 Gflops peak performance.
That result is depicted in Table 5.
We could get 25% performance for peak performance on single processor of VPP5000.

	[A] Fig.3	[C] Fig.11
(3) outer loop expansion	x	o
(4) load/store latency	x	o
(5) direct access	x	o
performance (MFlops)	1405	2381

Table 5 Performance for Model loops of the fluid analysis code on Vector Archi.

I summarize a result of discussion in this chapter.
- A new data structure depicted in Fig.9 was introduced to get good effect of prefetch.
- A new direct access manner using reordered array was introduced.
-21% performance for peak performance was got on pseudo vector architecture after applying both new data structure and new direct access manner.
- We got as good performance as the pseudo-vector execution performance in scalar execution with software pipelining.
-25% performance for peak performance was got on vector architecture after applying both new data structure and new direct access manner.

3.8 Evaluation of Performance for Model loops of Solid analysis code

Both (1) and (2) of performance factor, explained in chapter 3.6, was implemented to the model coding of Fig.4. A new model coding implementing both (3) and (5) of performance factor was made as in the fluid analysis code. The performance factor of (4) wasn’t implemented because of being able to get good performance in case[C] without implementing performance factor (4). Implementing (4) is a next problem. This new coding is shown in Fig.12.

```
      do j=1,1000
      do i=1,1000
        k = i
        kk = N0(k)
        x2(i) = a(3*kk-2)
        x1(i) = a(3*kk-1)
        x0(i) = a(3*kk  )
      enddo
      enddo
```

Fig.12(a) Model Coding of Direct Access (Reordering Process)

```
      do j=1,1000
      do i=1,1000
      k=i
          a(3*i-2)= a(3*i-2) - b(9*k-8)*x2(i) - b(9*k-7)*x1(i)      &
     &                                       - b(9*k-6)*x0(i)
          a(3*i-1)= a(3*i-1) - b(9*k-5)*x2(i) - b(9*k-4)*x1(i)      &
     &                                       - b(9*k-3)*x0(i)
          a(3*i  )= a(3*i  ) - b(9*k-2)*x2(i) - b(9*k-1)*x1(i)      &
     &                                       - b(9*k  )*x0(i)
      enddo
      enddo
```

Fig.12(b) Model Coding of Direct Access (Main Calculating Process)

Coding of Fig.12(a) is the reordering process for array:**a**, and coding of Fig.12(b) is the main calculating process. The performance of 122M operation/sec for reordering process was got, the performance of 335MFlops was got for main calculating process, and the total performance of 273Mflops was got. That result is depicted in Table 6[C]. In previous discussion, 27% performance for peak performance was got on pseudo vector architecture.

Next, 2 cases depicted in below were performed to measuring performance on the scaler architecture. That result is depicted in Table 6.

-scalar execution with compiler option Oss on SR8000 system (Table 6[D])
-scalar execution on Alpha system (Alpha 21164 533MHz) (Table 6[F])

Software pipelining was probably used in case[D], and was not probably used in case[F]. We could not get as good performance as the pseudo-vector execution performance in scaler execution because of no implementing performance factor (4).

	[A]	[C]	[D]	[F]
	Fig.4	Fig.12	Fig.12	Fig.12
(1) pseudo vectorization	o	o	x	x
(2) software pipelining	o	o	o	x
(3) outer loop expansion	x	o	o	o
(4) load/store latency	x	x	x	x
(5) direct access	x	o	o	o
performance (MFlops)	177	273	163	140

[A] base model coding on SR8000
[C] pseudo vector with software pipelining on SR8000
[D] scalar with software pipelining on SR8000
[F] scalar without software pipelining on Alpha system

Table 6 Performance for Model loops of the Structure analysis code on Pseudo Vector and Scalar Archi.

Next, model coding of both Fig.4 and Fig.12 were performed on Fujitsu VPP5000. That result is depicted in Table 5.
34% performance for peak performance was got on single processor of VPP5000.

	[A]	[C]
	Fig.4	Fig.12
(3) outer loop expansion	x	o
(4) load/store latency	x	x
(5) direct access	x	o
performance (MFlops)	2658	3276

Table 7 Performance for Model loops of the Structure analysis code on Vector Archi.

I summarize a result of discussion in this chapter.
- A new direct access manner using reordered array was introduced.
- 27% performance for peak performance was got on pseudo vector architecture after applying new direct access manner.
- We could not get as good performance as the performance of the fluid analysis solver in scalar execution.
- Getting a good performance is expected by implementing performance factor (4) in scalar execution.
- 34% performance for peak performance was got on vector architecture after applying new direct access manner.

4. Matrix Assemble Part of the Structure Analysis Code.

4.1 Outline of Original code of Matrix assemble part

Loop structure of matrix assemble part for original code is shown in Fig.13.

```
do kk1= 1, NZ
  do jj1= 1, NY
    do ii1= 1, NX
      |-jacobian calculate
      do ie= 1, 8
        do je= 1, 8
          do kpn= 1, 2
            do jpn= 1, 2
              do ipn= 1, 2
                |-stiffness matrix calculate
              enddo
            enddo
          enddo
        enddo
      enddo
    enddo
  enddo
enddo
```

Fig.13 Loop structure of matrix assemble part for original code

```
do loer = 1,50
  do lpn = 1,numlp+1
    do kp=1,2
      do jp=1,2
        do ip=1,2
          do ijk = 1,llnum
            |-jacobian calculate
          enddo
        enddo
      enddo
    enddo
    do ie= 1, 8
      do je= 1, 8
        do kpn= 1, 2
          do ijk = 1,llnum
            |-stiffness matrix calculate
          enddo
          do ijk = 1,llnum
            |-stiffness matrix calculate
          enddo
        enddo
      enddo
    enddo
  enddo
enddo
```

Fig.14 Loop structure of matrix assemble part for tuned code

The outermost 3 loops are 'typewriter scanned' elements loops. 'typewriter scan' is a rectangular sweep for the nested loops consecutively. Jacobian calculating process and element stiffness matrix calculating process are inside the outermost 3 loops. The element stiffness matrix calculating process assembles whole stiffness matrix. This process is constructed from 2 loops which have 8 elements for each loop. This 8 means number of node for hexahedral element. Innermost 3 loops corresponds to integral calculation for each local Coordinates of element.

4.2 Optimization Manner

Strategy for Optimization Manner is loop exchange putting a element loop into innermost loop for getting long vector length. If a elements loop is single loop, a processes of adding up numerical value to same node is produced. This process causes dependency of data for a elements loop. The elements loop was divided into some small loops of groups having no recursive reference. Moreover, each groups having no recursive reference were divided into some loops for saving the memory storage of jacobian calculation results. Elements loop was divided finally into 3 loops ,to do dividing for loops of above 2 times. Loop Structure is described in Fig.14.

Outermost loop is a first element loop. Next second loop is a second element loop.
Both the jacobian calculating process and the element stiffness matrix calculating process were put into a second loop. In the jacobian calculating process, A third element loop was put into 3 loops for integral calculation. In the element stiffness matrix calculating process, outer loop is 2

loops (8X8) assembling whole stiffness matrix. A loop of integral calculating for ζ coordinate was put into a outer loop and a third element loop was put into a loop of integral calculating for ζ . In a third loop of elements, 2 loops having each 2 elements for integral calculating for $\xi - \eta$ co-ordinates were replaced 4 (=2*2) lines operations.

4.3 Problem definision

Test problem definition is shown in below.
-Elastic Stracture Analysis for Cube Shape
-50000 Elements
-164000 DOF

4.4 Performance

Performance and operation counts of each process is depicted in Fig.15. CPUtime/FLOPS of matrix assembling process was 28.8sec/53.3MFlops (SX-4 /peak : 2GFlops) before tuning, and 2.09sec/736.8MFlops(same machine) after tuning.
CPUtime/FLOPS of whole test Code was 11.18sec/900.7MFlops(same machine).
4.88sec/2.06GFlops was attained on VPP5000 (peak : 9.6GFlops).

Process	Oparation Count(Flop)	Dec		SX-4		VPP5000	
		Cputime	Mflops	Cputime	Mflops	Cputime	Mflops
Bounbary condition Matrix Assemble Solver Pre Process	1.54G	0.54 11.85 0.07	123.6	0.19 1.73 0.17	736.8	0.04 0.76 0.11	1692.3
Solver	8.53G	197.10	43.3	9.09	938.4	3.97	2148.6
Total	10.07G	209.56	48.1	11.18	900.7	4.88	2063.5

Table8 Performance and Operation Counts of Each Process

5. Conclusion and Further Study

We have a prospect that Geofem get good parallel performance on various computer architecture by the research which has been made in GeoFEM team. In this research, we focus a target to performance with a single processor, and common data structure / coding manner to make GeoFEM running high performance on various computer architecture was researched.
We decided advancing the research by the following priority sequence.

(1)Solver Part of the structure analysis code and the fluid analysis code
(2)Matrix Assemble Part of the structure analysis code
(3)Matrix Assemble Part of the fluid analysis code

In this paper, I reported as below for each item on the priority sequence.

(1)Solver Part of the structure analysis code and the fluid analysis code
Data structure and coding manner was evaluated on scalar/vector/pseudo vector architecture. A new data structure and a new direct access manner was introduced in the fluid analysis solver. As a result, 21% performance for peak performance was obtained on pseudo vector architecture. And We got as good performance as the pseudo-vector execution performance in scalar execution. 25% performance for peak performance was obtained on vector architecture.

A new direct access manner was introduced in the structure code solver. As a result, 27% performance for peak performance was obtained on pseudo vector architecture. We could not get as good performance as the performance of the fluid analysis solver in scalar execution. But We expect getting a good performance by implementing a new data structure in scalar execution. 34% performance for peak performance was obtained on vector architecture.

My next work is to implement a new data structure for the structure analysis solver and to implement a new data structure and a new direct access manner for real GeoFEM code.

(2)Matrix Assemble Part of the structure analysis code
Coding of removing dependency was finished and performance was evaluated on vector/scaler machine. 736.8MFlops was obtained at matrix assembling process on SX-4. 900.7MFlops was obtained at whole test Code on SX-4, and 2.06GFlops was obtained on VPP5000 (peak : 9.6GFlops). 124MFlops was obtained at matrix assembling process on Alpha system(Alpha 21164 533MHz).

Evaluation of data structure and coding manner wasn't finished, and it is next my works.

(3)Matrix Assemble Part of the fluid analysis code
I do not start it yet and it is next my works.

Acknowledgments

This work is a part of the "Solid Earth Platform for Large Scale Computation" project funded by "Special Promoting Funds of Science & Technology" of STA (Science and Technology Agency of Japan). Author also thank to important suggestions by members of GeoFEM team.

Furthermore the authors would like to thank Professor Yasumasa Kaneda (Computing Center,The University of Tokyo) for discussion on high performance computing.

References

[1] K.Garatani,H.Nakamura,H.Okuda,G.Yagawa,GeoFEM:High Performance Parallel FEM for Solid Earth,Proceedings of 7th High-Performance Computing and Networking(HPCN Europe'99),LNCS-1593,133-140,1999.
[2]K.Nakajima,H.Okuda,Parallel Iterative Solver for Unstructured Grid using Directive/MPI Hybrid Programming Model for GeoFEM Platform on SMP Cluster Architectures.