DOI: 10.14529/jsfi210203

First Experience of Accelerating a Field-Induced Chiral Transition Simulation Using the SX-Aurora TSUBASA

Shinji Yoshida[1], *Arata Endo*[2], *Hirono Kaneyasu*[3], *Susumu Date*[2]

An analysis method based on the Ginzburg-Landau equation for the superconductivity is applied to the field-induced chiral transition simulation (FICT). However, the FICT is time consuming because it takes approximately 10 hours on a single SX-ACE vector processor. Moreover, the FICT must be repeatedly performed with parameters changed to understand the mechanism of the phenomenon. The newly emerged SX-Aurora TSUBASA, the successor of the SX-ACE processor, is expected to provide much higher performance to the programs executed on the SX-ACE as is. However, the SX-Aurora TSUBASA processor has changed its architecture of compute nodes and gives users three different execution models, which leads to users' concerns and questions in terms of how three execution models should be selectively used. In this paper, we report the first experience of using the SX-Aurora TSUBASA processor for the FICT. Specifically, we have developed three implementations of the FICT corresponding to the three execution models suggested by the SX-Aurora TSUBASA. For acceleration of the FICT, improvement of the vectorization ratio in the program execution and the efficient transfer of data to the general purpose processor as the vector host from the vector processor as the vector engine is explored. The evaluation in this paper shows how acceleration of the FICT is achieved as well as how much effort of users is required.

Keywords: SX-Aurora TSUBASA, OS Offload, VH Call, VEO, vectorization ratio.

Introduction

The simulation based on the Ginzburg-Landau equation [4] is the standard way to analyze the superconductivity in an external magnetic field. The analysis method is also applied to study a field-induced chiral phenomenon in the superconductivity [6, 7, 12], that is, the field-induced chiral transition simulation (FICT). In the chiral superconductivity, the superconducting electron pairs have the orbital magnetization [14], which makes the spontaneous intrinsic fields.

The chiral orbital magnetization paramagnetically couples with the external magnetic field, and it induces a field-induced chiral transition with generating the paramagnetic supercurrent [6, 7]. The field-induced chiral phenomena do not occur in the non-chiral state which generates the screening supercurrents diamagnetic to the external field with breaking the superconducting electron pairs. Thus, the field-induced chiral phenomena appear by the paramagnetic coupling of the orbital magnetization with the external field, as the chiral nature.

Such field-induced chiral phenomena are expected in candidates of chiral superconductors [5, 9, 16], such as the ruthenium oxide Sr_2RuO_4 (SRO) [11]. The field-induced chiral phenomena are theoretically clarified in the inhomogeneous superconductivity arising in SRO near interfaces of the micro-meter size Ru-metal inclusions embedded in the eutectic SRO-Ru [10], by using the field-induced chiral transition simulation (FICT) [6, 7].

However, the FICT is time consuming and furthermore has to be repeatedly performed with its parameters changed. Moreover, reducing the differential width in the Ginzburg-Landau equation is needed to analyze the dependence of the superconducting order parameter and the vector potential on distances in more detail, and then more computation is required. In addition

[1]Graduate School of Information Science and Technology, Osaka University, Osaka, Japan
[2]Cybermedia Center, Osaka University, Osaka, Japan
[3]Department of Science, University of Hyogo, Hyogo, Japan

to this, the lowering of temperature makes the superconductivity spread far away from the interface in the FICT. It needs the increase of mesh-number for the distance to analyze the chiral stability in the lower temperatures. At this point, the reduction of real time by the acceleration is effective for the analysis extended to the lower temperature. Thus, the acceleration makes it possible to clarify the field dependence of chiral phenomena in the lower temperatures in detail.

Until now, the SX-ACE system [3] has been used for performing the FICT, because the SX-ACE system gives a higher performance in comparison with general purpose processors. However, the newly-emerged SX-Aurora TSUBASA [2, 8], the successor of the SX-ACE, has changed the architecture of the compute node. The system architecture of the SX-Aurora TSUBASA consists of vector engines (VEs), which accelerate computation using vectorization and a vector host (VH), which is a standard x86 server and executes the operating system's tasks. This system architecture provides researchers with three types of execution models which differ in terms of how a program is executed on VE and VH. The FICT is accelerated on the SX-Aurora TSUBASA compute node by modifying the FICT that conforms to the best model that makes the FICT the fastest. However, the best execution model for the FICT is not clarified. Also, what operations are needed to modify the FICT that conforms to each execution model has not been clarified. From the perspective above, we report the first experience of accelerating the FICT using the SX-Aurora TSUBASA. We believe that the contribution of this paper is to present a case study of comparing the FICT based on the three execution models.

The rest of this paper is organized as follows. Section 1 shows our approach to accelerate the FICT after briefly explaining the FICT and the SX-Aurora TSUBASA. Section 2 presents our methods for accelerating the FICT. In Section 3, we evaluate the FICT accelerated with our acceleration approach on the SX-Aurora TSUBASA. In Conclusion, this paper is concluded.

1. Approach to Accelerate the FICT

This section first describes an overview of the FICT and the SX-Aurora TSUBASA. Secondly, we present our approaches to accelerate the FICT on the SX-Aurora TSUBASA.

1.1. Overview of the FICT

The chiral superconducting state is denoted with two components of the superconducting order parameter, while the non-chiral state is represented with one component [14]. The chiral transition corresponds to the transition to the two components state by yielding the second component in the addition to the one component. Thus, the field-induced chiral transition occurs by inducing the second component in the application of the external field [6, 7].

The FICT computes both the two order parameter components and the vector potential. Moreover, paramagnetic and screening supercurrents are calculated from the order parameter and the vector potential which are obtained as results of the equation. In the FICT, the strength of an external magnetic field is increased to simulate how the strength of the external magnetic field affects the superconducting state at the fixed temperatures respectively. The superconducting is computed at each strength of the external magnetic field at the fixed temperatures. That is, the dependence of two order parameter components and the vector potential on the distance is calculated iteratively under varying parameters for an external magnetic field and a temperature.

Figure 1 shows a flow chart of the FICT. When the FICT starts, the initial parameters regarding the superconducting order parameter are given. Next, the Ginzburg-Landau equation is computed, and then the result of the equation is written to a file, which repeats until the strength of the external magnetic field reaches a set value. After that, the parameters regarding the superconducting order parameter are initialized to the initial parameters and the strength of the temperature increases. When the strength of the external magnetic field and that of the temperature reach their values which we set, the FICT finishes.

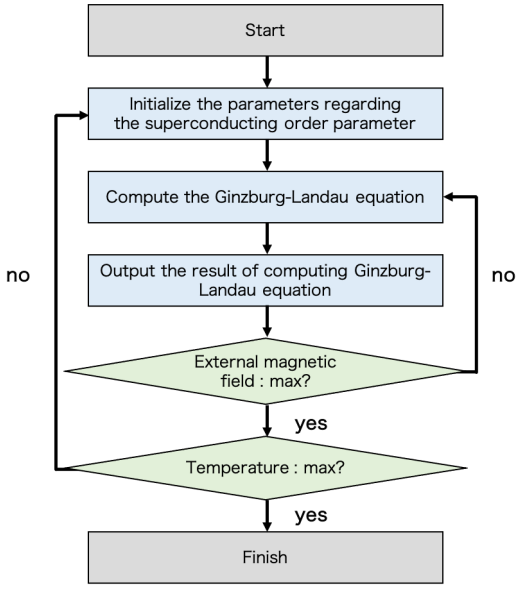


Figure 1. Flow chart of the FICT

1.2. Overview of the SX-Aurora TSUBASA

Figure 2 shows an example system architecture of the SX-Aurora TSUBASA system. A compute node where the SX-Aurora TSUBASA processor is hosted is composed of vector host (VH) and vector engine (VE). In the inside of the compute node, VE and VH are connected to a PCI Bus as shown in Fig. 2. In this architecture, the computation is performed on VE and VH works supplementarily with VE. For example, when VE requires I/O-related system calls, VH performs such system calls instead of VE.

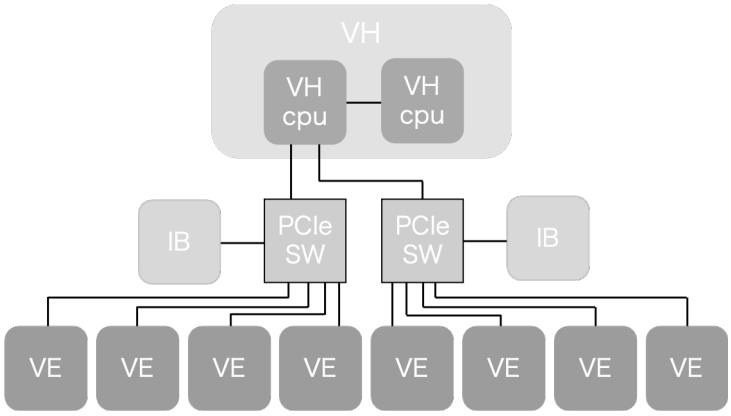
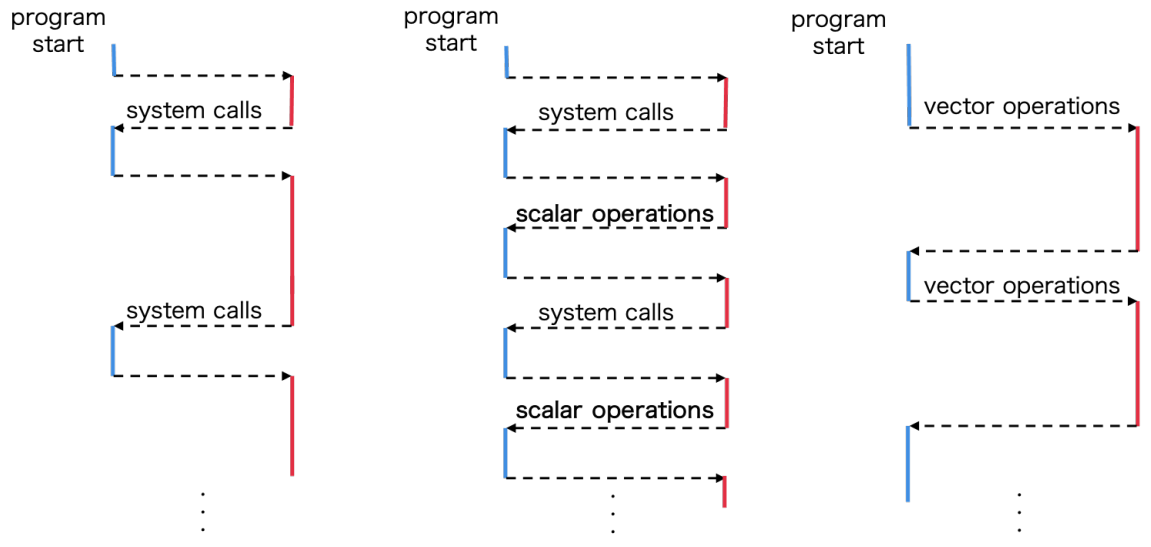


Figure 2. An example system architecture of the SX-Aurora TSUBASA system (A300-8)

Accompanied the change in the architecture, the SX-Aurora TSUBASA offers three execution models. In this paper, we refer to these three models: standard (OS Offload) execution model, VH Call execution model, and Vector Engine Offload (VEO) execution model. Figure 3 compares these three execution models.

- In the standard execution model, a program is started on VH. After that, however, it is mostly executed on VE. Only when the program needs to accomplish operating system tasks, system calls are automatically offloaded to VH from VE. The advantage of this execution model is that when researchers write the program, they do not need to know the system architecture of the SX-Aurora TSUBASA. Furthermore, most of programs which are executed on the SX-ACE system can be easily ported to the SX-Aurora TSUBASA system without much modification. On the other hand, the disadvantage might be that researchers have no way of explicitly using VH when researchers use this model.
- In the VH Call execution model, a program is initially started on VH. After that, it is mainly executed on VE. Basically, this execution model is the same as the standard execution model explained above but it allows users to explicitly invoke the VH Call for requesting the processing of some program portion on VH. Thus, the system calls are automatically offloaded to VH. In addition, some portion of calculations can be offloaded explicitly by users to VH from VE. The merit of the VH Call execution model is that a program containing scalar-friendly calculations is executed faster in the VH Call execution model than in the standard execution model. On the other hand, the disadvantage of this execution model is that researchers must have a fairly detailed knowledge of the SX-Aurora TSUBASA composed of VH and VE as well as the program semantics regarding this execution model.
- In the VEO execution model, a program is initially started and mainly executed on VH. Researchers can explicitly offload some portion of the program to VE from VH. The advantage and disadvantage of the VEO execution model are almost the same as the VH Call execution model. Regarding the disadvantage, the program to be executed on VH must be written in C unlike the VH Call execution model.



(a) Standard (b) VH Call (c) VEO

Figure 3. Three execution models of the SX-Aurora TSUBASA

1.3. Computational Characteristics of the FICT

We first describe the computational characteristics of the FICT as obtained through the program execution analysis. Next, we analyze the code that outputs the result of the Ginzburg-Landau equation.

1.3.1. FICT execution analysis information

Analysis information on program execution is obtained with a performance analysis tool named PROGINF [13], which is shipped with the SX-Aurora TSUBASA. Table 1 contains Real Time (the program execution time), Vector Time (the total time taken to execute vector operations in the program execution) and V. Op. Ratio (the vectorization ratio which is the ratio of vector operations in the program to all operations in the program) obtained as the analysis information on the FICT execution.

Table 2 shows the analysis information on each function composing the FICT with a performance analysis tool named FTRACE [13] for the SX-Aurora TSUBASA, which shows Frequency (the call count of each function in program execution), EXCLUSIVE TIME (the execution time of each function), V. Op. Ratio (the vectorization ratio), REQ. B/F (the B/F rate (the memory bandwidth per the performance) needed by the program) and PROC.NAME (the function name). In the case that SYS. B/Y (the rate of the highest observed memory bandwidth of the system to the observed peak performance of the system) is smaller than REQ. B/F of the program, the performance of the system does not achieve the peak performance of the system. Although SYS. B/F of SX-Aurora TSUBASA is under REQ. B/F of the FICT, SYS. B/F of SX-Aurora TSUBASA is higher than SYS. B/F of scalar processors [2]. Therefore, the FICT is suited for SX-Aurora TSUBASA. Also, according to the results, the vectorization ratio of the GRAD subroutine that computes the Ginzburg-Landau equation is lower than the MAIN function. Also, the execution time of the GRAD subroutine itself is approximately five times longer than the execution time of the MAIN program.

Table 1. FICT execution analysis information from PROGINF

Topic	Value
Real Time (sec)	1413.805725
Vector Time (sec)	418.584454
V. Op. Ratio (%)	94.480496

Table 2. FICT execution analysis information from FTRACE

Frequency	EXCLUSIVE TIME [sec] (%)	V. Op. Ratio	REQ. B/F	PROC.NAME
474965635	1416.492 (82.9)	92.43	1.86	GRAD
1	291.970 (17.1)	98.22	1.52	MAIN
474965636	1708.462 (100.0)	94.00	1.76	total

1.3.2. I/O characteristic of the FICT

Figure 4 shows an I/O-intensive portion contained in the FICT. The FICT is written in Fortran. When `WRITE` at line 10 is executed, the data written at lines 11–14 is transferred to VH from VE (data transfer) and then outputted to a file on VH. Data transfer occurs every time `WRITE` is executed. When the FICT is executed, a loop at lines 1–17 is executed 2 times, a loop at lines 3–16 is executed 51 times and a loop at lines 5–15 is executed 240 times. Therefore, when the FICT is executed, data transfer occurs 24,480 times.

```
1       DO nt=ntmin,ntmax,ndt
2             ... etc. ...
3         DO nh=nhmin,nhmax,ndh
4             ... computing the Ginzburg-Landau equation ...
5           DO nx=1,N
6             x=dx*0.5d0*(dble(nx)+dble(nx-1))
7             ... etc. ...
8             ua=0.5d0*(ay(nx)+ay(nx-1))
9             da=(ay(nx)-ay(nx-1))/dx
10            WRITE(1110000000+100000*nt+nh,*)
11      &       x,x0,x1
12      &      ,ev(nx-1),et(nx-1),dabs(ev(nx-1)),dabs(et(nx-1))
13            ... etc. ...
14      &      ,DABS(g*(-k4*uv*dt))
15          END DO
16        END DO
17       END DO
```

Figure 4. A portion of I/O operations in the FICT code

2. Acceleration Approach

To understand which execution model is effective for the acceleration of the FICT, we have modified this original FICT code to conform to the three execution models suggested by the SX-Aurora TSUBASA architecture. In this section, we summarize how we attempted to accelerate the original FICT code, by focusing just on the improvement of the vectorization ratio and the efficiency improvement of I/O operations from VE to VH.

2.1. Improving the Vectorization Ratio

The vectorization ratio is an important criterion in achieving higher performance on vector processors. Figure 5 is the portion of the original GRAD subroutine. As shown, the loop from lines 1–4 and the loop from lines 7–10 seem to be easily combined as a single loop. According to the developer of the FICT, lines 5 and 6 remained on purpose to keep the readability and maintainability of the FICT code. There sometimes exits code structures that prevent the acceleration of execution codes in actual users' programs and sometimes researchers as users do not want the code to be modified because the code itself could lose the readability and maintainability.

In the FICT code, from a developer's perspective, two loop structures exist that prevent vectorization. In this paper, each of the two loop structures was combined so that the vectorization ratio was improved. This acceleration approach was commonly conducted in all our three different implementations based on the standard, VH Offload and VEO execution models.

```
1       DO nx=1,N
2         fv(nx)=0.0d0
3         ... etc. ...
4       END DO
5       uv=0.5d0*(ev(1)+ev(0))
6       ... etc. ...
7       DO nx=1,N
8         uv=0.5d0*(ev(nx)+ev(nx-1))
9         ... etc. ...
10      END DO
```

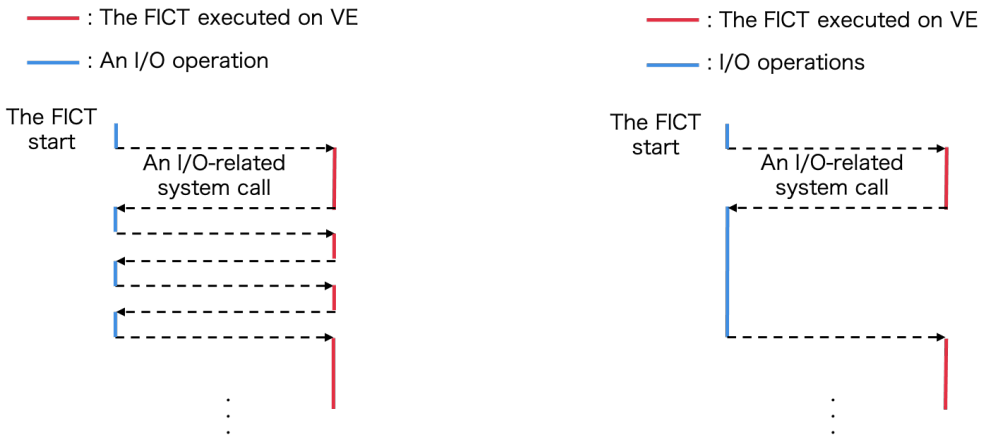
Figure 5. A portion of the original GRAD subroutine code

2.2. Efficiency Improvement of Data Transfer

Under the SX-Aurora TSUBASA system architecture composed of VH and VE, the I/O operations heavily affect the total performance of the application execution. This is because the vector processor as VE is expected to be the main processor for the application execution and the general purpose processor as VH is considered a helper processor. More specifically, I/O operations from the code running on VE trigger I/O-related system calls in the operating system running on VH. At the same time, VE must wait for the completion of system calls on VH. For this reason, performing the efficient I/O operations from the code running on VE by reducing the number of I/O operations helps developers shorten the total execution time of their application codes.

A possible approach to realize such efficient I/O operations for acceleration is to simply reduce the number of data transfers from VE to VH. In Fig. 7, which shows a portion of the original FICT code, an I/O operation at lines 10–13 triggers an I/O-related system call (Fig. 6a). We must modify this procedure as shown in Fig. 6b so that many I/O operations are not triggered by an I/O-related system call. For this modification, we moved the I/O operations part at lines 10–13 to the inside of the loop at lines 1–17 as shown in Fig. 4.

In the following subsections, how the reduction of data transfers is realized is explained in each execution model in addition to the above modification.



(a) A portion of the FICT code before modification (b) A portion of the FICT code after modification

Figure 6. Procedures of I/O operations

```
      DO nt=ntmin,ntmax,ndt
            ... etc. ...
        DO nh=ntmin,ntmax,ndt
            ... computing the Ginzburg-Landau equation ...
          DO nx=1,N
            x=dx*0.5d0*(dble(nx)+dble(nx-1))
            ... etc. ...
            ua=0.5d0*(ay(nx)+ay(nx-1))
            da=(ay(nx)-ay(nx-1))/dx
            WRITE(1110000000+100000*nt+nh,*)
     &      ,x
               ... etc. ...
     &      ,DABS(g*(-k4*uv*dt))
          END DO
        END DO
        CLOSE(1110000000+100000*nt+nh)
      END DO
```

Figure 7. A portion of the FICT code before modification

```
      DO nt=ntmin,ntmax,ndt
            ... etc. ...
        DO nh=nhmin,nhmax,ndh
            ... computing the Ginzburg-Landau equation ...
          DO nx=1,N
            x=dx*0.5d0*(dble(nx)+dble(nx-1))
            ... etc. ...
            ua=0.5d0*(ay(nx)+ay(nx-1))
            da=(ay(nx)-ay(nx-1))/dx
            w(1,nx,nh)=x
            ... etc. ...
     &      ,w(63,nx,nh)=DABS(g*(-k4*uv*dt))
          END DO
        END DO
        ... outputting the results ...
      END DO
```

Figure 8. A portion of the FICT code after modification

2.2.1. Modification in the standard execution model

The OS Offload execution model is the standard execution model of the SX-Aurora TSUBASA architecture. The merit of this standard execution model is that it is designed to be executed as the as-is code running on the SX-ACE system without requiring developers to be aware of the complex architecture composed of VH and VE. However, the demerit of this execution model is that it does not allow developers to directly touch the underlying low-level I/O mechanisms. In the standard execution model, further efforts could not be carried out.

2.2.2. Modification in the VH Call execution model

To utilize the VH Call programming semantics, the developers must write each function to be executed on VH and VE in a separate file. Also, the developers must write such functions on VH and VE conforming to the program semantics regarding the VH Call execution model. A

suite of dedicated functions as shown in lines 15–18 in Fig. 10 are prepared so that the developers utilize the VH Call execution model. The files executed on VE and written in Fortran must be complied with *nfort*, and the ones on VH with *gfortran*. Figure 9 shows an example of Makefile for this execution model.

Figure 10 is the code fragmentation to be executed on VE, and Fig. 11 on VH. In lines 15–18 in Fig. 10, results of computing the Ginzburg-Landau equation are transferred to VH from VE. After that, these results are output to files on VH (Fig. 11). The result of executing this code portion is the same result when lines 10–13 in Fig. 7 are executed.

```
all: VE VH
VE: a fortran file name executed on VE
        nfort -o $@ $^ -lvhcall_fortran -w -report-all
VH: a fortran file name executed on VH
        gfortran -o $@ $^ -fpic -shared
```

Figure 9. Makefile in the VH Call execution model

```
1       ... configuration for using VH Call ...
2       DO nt=ntmin,ntmax,ndt
3             ... etc. ...
4         DO nh=nhmin,nhmax,ndh
5             ... computing the Ginzburg-Landau equation ...
6           DO nx=1,N
7             x=dx*0.5d0*(dble(nx)+dble(nx-1))
8             ... etc. ...
9             da=(ay(nx)-ay(nx-1))/dx
10            w(1,nx,nh)=x
11            ... etc. ...
12            w(63,nx,nh)=DABS(g*(-k4*uv*dt))
13          END DO
14        END DO
15        ir=fvhcall_args_set(ca,fvhcall_intent_inout,1,w)
16        ir=fvhcall_args_set(ca,fvhcall_intent_inout,2,nt)
17        ir=fvhcall_invoke_with_args(sym, ca)
18        CALL fvhcall_args_clear(ca)
19      END DO
```

Figure 10. A portion of the FICT code on VE in the VH Call execution model

```
1       SUBROUTINE vh_write(w,nt)
2         DO nh=nhmin,nhmax,ndh
3           DO nx=1,N
4             WRITE(1110000000+100000*nt+nh,*)
5     &      ,w(1,nx,nh)
6               ... etc. ...
7     &      ,w(63,nx,nh)
8           END DO
9           CLOSE(1110000000+100000*nt+nh)
10        END DO
11      END SUBROUTINE vh_write
```

Figure 11. A portion of the FICT code on VH in the VH Call execution model

2.2.3. Modification in the VEO execution model

Likewise in the case of the VH Call execution model, to utilize the VEO programming semantics, the developers must write each function to be executed on VH and VE in a separate file. Unlike the VH Call execution model, the files to be executed on VH must be written in C. Also, the developers must write such functions on VH and VE conforming to the program semantics regarding the VEO execution model. A suite of dedicated functions as shown in lines 4–8 in Fig. 13 are prepared so that the developers utilize the VEO execution model. The files executed on VE and written in Fortran must be complied with *nfort*, and the ones on VH and in C with *gcc*. Figure 12 shows an example of Makefile for this execution model.

Figure 13 is the code fragmentation to be executed on VH, and Fig. 14 is on VE. In lines 4–8 in Fig. 13, parameters which are required to compute the Ginzburg-Landau equation and store results of this computation are transferred to VE. After that, the Ginzburg-Landau equation is calculated on VE (Fig. 14). The result of executing this code portion is the same result when lines 10–13 in Fig. 7 are executed.

```
all: vefortran fortran
vefortran: a fortran file name executed on VE
        nfort -c -o libvefortran.o $^ /opt/nec/ve/bin/mk_veorun_static -o $@
          libvefortran.o
fortran: a C file name executed on VH
        gcc -o $@ $^ -I/opt/nec/ve/veos/include -L/opt/nec/ve/veos/lib64 -Wl,-
          rpath=/opt/nec/ve/veos/lib64 -lveo
```

Figure 12. Makefile in the VEO execution model

```
1   int main(){
2     ... configuration for using VEO ...
3     for(nt=ntmin; nt<=ntmax; nt=nt+ndt){
4       veo_args_set_stack(argp,VEO_INTENT_OUT,0,(char *)&w,sizeof(w));
5       veo_args_set_stack(argp,VEO_INTENT_IN,1,(char *)&nt,sizeof(nt));
6       id = veo_call_async_by_name(ctx,handle,"veo_",argp);
7       veo_call_wait_result(ctx,id,&retval);
8       veo_args_clear(argp);
9       ... outputting the results ...
10    }
11  }
```

Figure 13. A portion of the FICT code on VH in the VEO execution model

2.3. Approach Summary

We have applied seven implementations of the FICT based on the above approach by focusing on the improvement of the vectorization ratio and the efficiency improvement in I/O operations. Table 3 shows what tuning efforts are done in each implementation. OR indicates the original code of the FICT.

```
1       SUBROUTINE veo(w,nt)
2       ... etc. ...
3       DO nh=nhmin,nhmax,ndh
4         ... computing the Ginzburg-Landau equation ...
5         DO nx=1,N
6           x=dx*0.5d0*(dble(nx)+dble(nx-1))
7           ...etc. ...
8           w(1,nx,nh)=x
9           ... etc. ...
10          w(63,nx,nh)=DABS(g*(-k4*uv*dt))
11        END DO
12      END DO
13      END SUBROUTINE veo
```

Figure 14. A portion of the FICT code on VE in the VEO execution model

Table 3. The FICT codes we have implemented

Execution model	ID	Vectorization improvement (Section 2.1)	I/O improvement (Section 2.2)
Standard	OR	no	no
	A1	yes	no
VH Call	B1	yes	no
	B2	no	yes
	B3	yes	yes
VEO	C1	yes	no
	C2	no	yes
	C3	yes	yes

3. Evaluation

As summarized in Section 2.3, we have developed seven implementations of the FICT based on three execution models suggested by the SX-Aurora TSUBASA by focusing on the improvement of the vectorization ratio and the efficiency of the I/O operations. In this section, we compare these three execution models in terms of performance and ease of programming. In all evaluations, we used a single core of processors because the maximum loop length in the FICT, 240, was smaller than the vector length of SX-Aurora TSUBASA, 256.

3.1. Experimental Environments

For the evaluation purpose, the SX-Aurora TSUBASA system the detail specification of which is shown in Tab. 4 and Tab. 5 was used. “Vector length” is the number of elements that VE can perform per instruction. The system is composed of 2 Intel Xeon Gold 6126 [15] with 96 GB memory as VH and SX-Aurora TSUBASA processor Type 10B as 8 VEs. Figure 2 illustrates the inside architecture of the system (A300-8). The Intel processors as VH and VEs are connected with PCI (Gen 3)16. In this experiment, a pair of VH and VE were used.

Table 4. Spec. of VE

VE type	Type 10B
Vector length	256
Number of cores	8
Frequency	1.4 GHz
Performance / core (DP)	268.8 GFlops
Performance / core (SP)	537.6 GFlops
LLC capacity	16 MB
Memory bandwidth	1.2 TB/s
Memory capacity	48 GB

Table 5. Spec. of VH

Processor	Intel Xeon Gold 6126 × 2 socket
Number of cores	12 × 2
Frequency	2.6 GHz
Performance / core (DP)	83.2 GFlops
Performance / core (SP)	166.4 GFlops
LLC capacity	19.25 MB × 2
Memory bandwidth	128 GB/s × 2
Memory capacity	96 GB × 2

3.2. Performance of the SX-Aurora TSUBASA

First, we measured the execution time of the original FICT code (OR) on the SX-ACE system, the SX-Aurora TSUBASA system and a compute node of the OCTOPUS system (the Xeon 6126 system) [1], which has the same processors as the VH of the SX-Aurora TSUBASA system. The specification of the SX-ACE is shown in Tab. 6. Figure 15 shows the execution time of the FICT on the SX-ACE system, the SX-Aurora TSUBASA system and the Xeon 6126 system. The graph indicates that the execution time of OR on the SX-Aurora TSUBASA system was 1413.81 seconds, on the SX-ACE system the execution time was 2451.61 seconds and on the Xeon 6126 system was 7828.43 seconds. According to this result, the SX-Aurora TSUBASA system delivers 1.7 times higher performance to the FICT than the SX-ACE system. Moreover, this result indicates that SX-Aurora TSUBASA is more suited for the FICT than scalar processors. Although the performance per core of SX-Aurora TSUBASA is almost four times that of SX-ACE, the execution time of the FICT is reduced by only about 40 % on the SX-Aurora TSUBASA system. As stated in Section 1.3.1, since SYS. B/F of SX-Aurora TSUBASA is smaller than REQ. B/F of the FICT, the performance of SX-Aurora TSUBASA does not achieve the theoretical performance of SX-Aurora TSUBASA.

Table 6. Spec. of the SX-ACE system

Vector length	256
Number of cores	4
Performance / core (DP)	64 GFlops
Memory bandwidth	256 GB/s
Memory capacity	64 GB

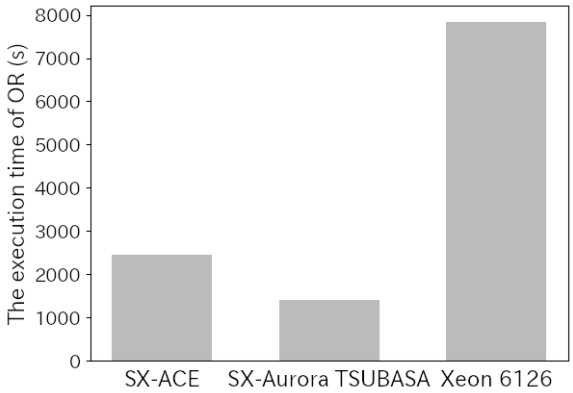


Figure 15. The execution time of OR on the SX-ACE system, the SX-Aurora TSUBASA system and the Xeon 6126 system

3.3. Effect of Improving the Vectorization Ratio

Next, we observed and compared how the vectorization ratio and the execution time were improved with the improvement of the vectorization ratio. Figures 16 and 17 show the vectorization ratio and the execution time of OR and A1. The vectorization ratio of the FICT increased from 94.4 % to 99.4 %. The execution time of A1 was reduced to 447.02 seconds. This result indicates that the improvement of the vectorization ratio reduced the execution time of A1 by 68.4 % compared to the original code of the FICT (OR).

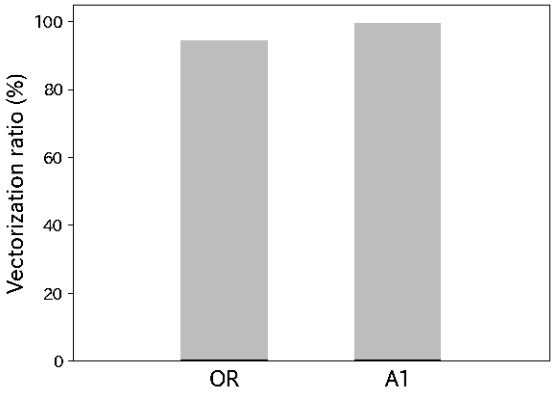


Figure 16. The vectorization ratio of the FICT

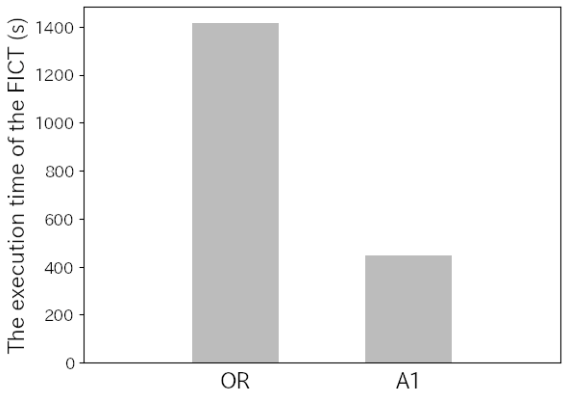


Figure 17. The execution time with improvement of the vectorization ratio

3.4. Effect of the Efficiency Improvement of Data Transfer

Next, we observed and compared how the number of data transfers was reduced and the execution time was improved with the approach summarized in Section 2.2. Table 7 shows the number of data transfers of OR, B2 and C2. The number of data transfers of OR was 24,480 and that of B2 and C2 was 2. Figure 18 shows the execution time of OR, B2 and C2. The execution time of B2 was 1372.26 seconds and that of C2 was 1320.19 seconds. This result indicates that reduction of data transfers in the VH Call and VEO execution models slightly reduced the execution time of the FICT compared to that of the original FICT on the standard execution model.

Table 7. The number of data transfers

ID	The number of data transfers
OR	24480
B2	2
C2	2

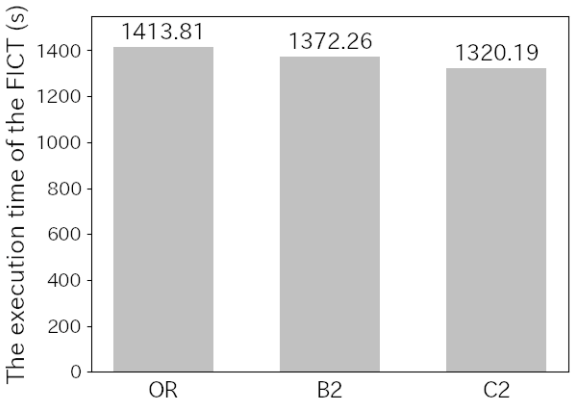


Figure 18. Execution time with reduction of data transfers

3.5. Performance Acceleration of the Three Execution Models

Next, we measured the execution time of OR, A1, B3 and C3 to investigate the three implementations based on the three execution models by focusing on the improvement of vectorization and I/O operational efficiency. Figure 19 shows the result of this measurement. The execution time of A1 was reduced to 447.02 seconds. On the other hand, the execution time of B3 was 446.07 seconds and that of C3 was 443.29 seconds. This result indicates that the implementations of the FICT based on the VH Call and VEO execution models were slightly superior in performance to that based on the standard execution model as the result of our tuning effort that focused on the improvement of the vectorization ratio and the I/O operational efficiency.

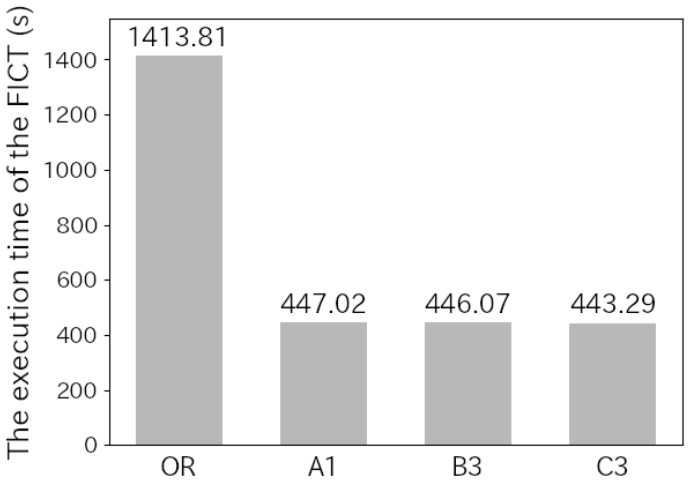


Figure 19. The execution time of the FICT with acceleration approaches

3.6. Code Modification for Acceleration

We investigated how many lines must be modified to implement seven FICT codes in Tab. 8. The number of lines modified to implement A1, B1 and C1 is the same. In the VH Call and VEO execution models, the number of lines modified to implement B2 and C2 was approximately 4.2 times larger than A1. To implement C3, we had to rewrite the code written in Fortran to be executed on VH to C.

Table 8. Modification for the FICT code

Execution model	ID	The number of modified lines	Programming language modification
Standard	OR	0	no
	A1	42	no
VH Call	B1	42	no
	B2	175	no
	B3	217	no
VEO	C1	42	no
	C2	180	no
	C3	222	yes

Conclusion

In this paper, we reported the experience of accelerating the FICT using the newly emerged SX-Aurora TSUBASA. Specifically, we applied seven implementations focusing on the three execution models which the SX-Aurora TSUBASA suggested as well as the improvement of vectorization and I/O operations as tuning methods. Hopefully, this will alleviate the FICT users' concerns and questions about how the three execution models are to be selected. Our evaluation showed that the execution time of the implementation conforming to the VEO execution model was 0.83 % shorter in comparison with the implementation conforming to the standard execution model (OS Offload), which we suggested as the standard execution model. On the other hand, we also showed that the VEO and VH Call execution models require us to write or modify the code by recognizing the structure of the SX-Aurora TSUBASA system, while the standard execution model allows us to easily accelerate the FICT running on the SX-ACE system, which focuses on the improvement of the vectorization and I/O operations. In the case of the FICT, the VEO execution model is the best if researchers respect performance. However, the standard execution model would be the best if researchers respect readability and maintainability of the FICT.

Acknowledgements

We thank the Cybermedia Center (CMC) of Osaka University which provided the SX-ACE system and the SX-Aurora TSUBASA system. This research is partly supported by the Joint Usage and Research (No. jh200032) of JHPCN. Also, we thank the technical team of the NEC at CMC for useful advice and direction. We are grateful to Kouki Otsuka for valuable discussions on the theoretical aspects of chiral superconductivity. This work is supported by the JSPS Core-to-core program No. JPJSCCA20170002.

References

1. Cybermedia Center of Osaka University: OCTOPUS. `http://www.hpc.cmc.osaka-u.ac.jp/octopus/`, accessed: 2021-08-08

2. Egawa, R., Fujimoto, S., Yamashita, T., *et al.*: Exploiting the Potentials of the Second Generation SX-Aurora TSUBASA. In: 2020 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS). pp. 39–49. IEEE (2020). `https://doi.org/10.1109/PMBS51919.2020.00010`

3. Egawa, R., Komatsu, K., Momose, S., *et al.*: Potential of a modern vector supercomputer for practical applications: performance evaluation of SX-ACE. The Journal of Supercomputing 73(9), 3948–3976 (2017). `https://doi.org/10.1007/s11227-017-1993-y`

4. Ginzburg, V.L., Landau, L.D.: On the theory of superconductivity. Journal of Experimental and Theoretical Physics 20, 1064–1082 (1950), English translation in: L.D. Landau:

Collected papers. Oxford: Pergamon Press, 546–568 (1965). https://doi.org/10.1016/B978-0-08-010586-4.50078-X

5. Grinenko, V., Ghosh, S., Sarkar, R., *et al.*: Split superconducting and time-reversal symmetry-breaking transitions in Sr_2RuO_4 under stress. Nature Physics 17(6), 748–754 (2021). https://doi.org/10.1038/s41567-021-01182-7

6. Kaneyasu, H., Enokida, Y., Nomura, T., *et al.*: Features of Chirality Generated by Paramagnetic Coupling to Magnetic Fields in the 3 K-Phase of Sr_2RuO_4. In: JPS Conference Proceedings 30, 011039-1-6 (2020). https://doi.org/10.7566/JPSCP.30.011039

7. Kaneyasu, H., Enokida, Y., Nomura, T., *et al.*: Properties of the H-T phase diagram of the 3-K phase in eutectic Sr_2RuO_4-Ru: Evidence for chiral superconductivity. Physical Review B 100(21), 214501 (2019). https://doi.org/10.1103/PhysRevB.100.214501

8. Komatsu, K., Momose, S., Isobe, Y., *et al.*: Performance Evaluation of a Vector Supercomputer SX-Aurora TSUBASA. In: SC18: International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 685–696. IEEE / ACM (2018). https://doi.org/10.1109/SC.2018.00057

9. Luke, G.M., Fudamoto, Y., Kojima, K.M., *et al.*: Time-reversal symmetry-breaking superconductivity in Sr_2RuO_4. Nature 394(6693), 558–561 (1998). https://doi.org/10.1038/29038

10. Maeno, Y., Ando, T., Mori, Y., Ohmichi, E., *et al.*: Enhancement of Superconductivity of Sr_2RuO_4 to 3 K by Embedded Metallic Microdomains. Physical Review Letters 81(17), 3765–3768 (1998). https://doi.org/10.1103/PhysRevLett.81.3765

11. Maeno, Y., Hashimoto, H., Yoshida, K., *et al.*: Superconductivity in a layered perovskite without copper. Nature 372(6506), 532–534 (1994). https://doi.org/10.1038/372532a0

12. Matsumoto, M., Sigrist, M.: Quasiparticle States near the Surface and the Domain Wall in a $p\_x \pm ip\_y$ -Wave Superconductor. Journal of the Physical Society of Japan 68, 994–1007 (1999). https://doi.org/10.1143/JPSJ.68.994

13. NEC: PROGINF/FTRACE Users Guide. https://www.hpc.nec/documents/sdk/pdfs/g2at03e-PROGINF_FTRACE_User_Guide_en.pdf, accessed: 2020-12-07

14. Sigrist, M., Ueda, K.: Phenomenological theory of unconventional superconductivity. Reviews of Modern Physics 63(2), 239–311 (1991). https://doi.org/10.1103/RevModPhys.63.239

15. Voevodin, V.V., Antonov, A.S., Nikitenko, D.A., *et al.*: Supercomputer Lomonosov-2: Large Scale, Deep Monitoring and Fine Analytics for the User Community. Supercomputing Frontiers and Innovations 6(2), 4–11 (2019). https://doi.org/10.14529/jsfi190201

16. Xia, J., Maeno, Y., Beyersdorf, P.T., *et al.*: High Resolution Polar Kerr Effect Measurements of Sr_2RuO_4: Evidence for Broken Time-Reversal Symmetry in the Superconducting State. Physical Review Letters 97(16), 167002 (2006). https://doi.org/10.1103/PhysRevLett.97.167002