NEC

SX-Aurora TSUBASA

SX-Aurora TSUBASA

Program Execution Quick Guide

Proprietary Notice

Preface

The SX-Aurora TSUBASA Program Execution Quick Guide is the document for those using SX-Aurora TSUBASA for the first time. It explains the basic usage of the SX-Aurora TSUBASA software products; the compilers, MPI, NQSV, PROGINF, and FTRACE.

This guide assumes the following installation, setup, and knowledge.

- VEOS and the necessary software have been installed and set up.
- Users are able to log in to the system or use the job scheduler NQSV (NEC Network Queuing System V) or PBS.
- Users have knowledge of Fortran compiler (nfort), C compiler (ncc), C++ compiler (nc++), and NEC MPI.

This guide assumes the version of VEOS is 3.0.2 or later.
The version of VEOS can be confirmed by the following way.

```
$ rpm -q veos
veos-3.0.2-1.el8.x86_64
```

VH/VE hybrid MPI execution is available in NEC MPI version 2.3.0 and later. The NEC MPI for VE30 is available at version 3.3.0 and later. The version of NEC MPI corresponds to the following directory "/opt/nec/**<ve>**/mpi/**<version>**". **<ve>** is ve3 for VE30, otherwise ve.

```
$ ls -d /opt/nec/ve/mpi/2.3.0
/opt/nec/ve/mpi/2.3.0
```

Definitions and Abbreviations

abbreviation	definition
Vector Engine, VE	Vector Engine, VE is a center of SX-Aurora TSUBASA and are the part where a vector operation is performed. It's PCI Express card and it's loaded into x86 server and it's used.
Vector Host, VH	It is a server that is a host computer holding Vector Engine
IB	An abbreviation of InfiniBand.
HCA	An abbreviation of Host Channel Adapter. The hardware to communicate with other nodes using InfiniBand.
MPI	An abbreviation of Message Passing Interface. The standard specifications to do a parallel computing over nodes. It's possible to use MPI for communication among processes on a single node. The use with OpenMP is also possible.
PBS	Job scheduler by Altair Engineering, Inc. PBS Professional is the commercial version and OpenPBS is its open source version.
chunk	A group of resources users request under PBS. The resources in a chunk are always allocated from a VH.
chunk set	A set of one or more identical chunks.

Contents

Chapter1　Outline of SX-Aurora TSUBASA

SX-Aurora TSUBASA consists of the vector engine which does application data processing (VE) and the x86/Linux node (VH) which does OS processing mainly.
A program of SX-Aurora TSUBASA starts from VH which offers the OS function, and is carried out on each VE. Therefore when executing SX-Aurora TSUBASA program, it's necessary to designate and carry out the VE number and the number of VE.

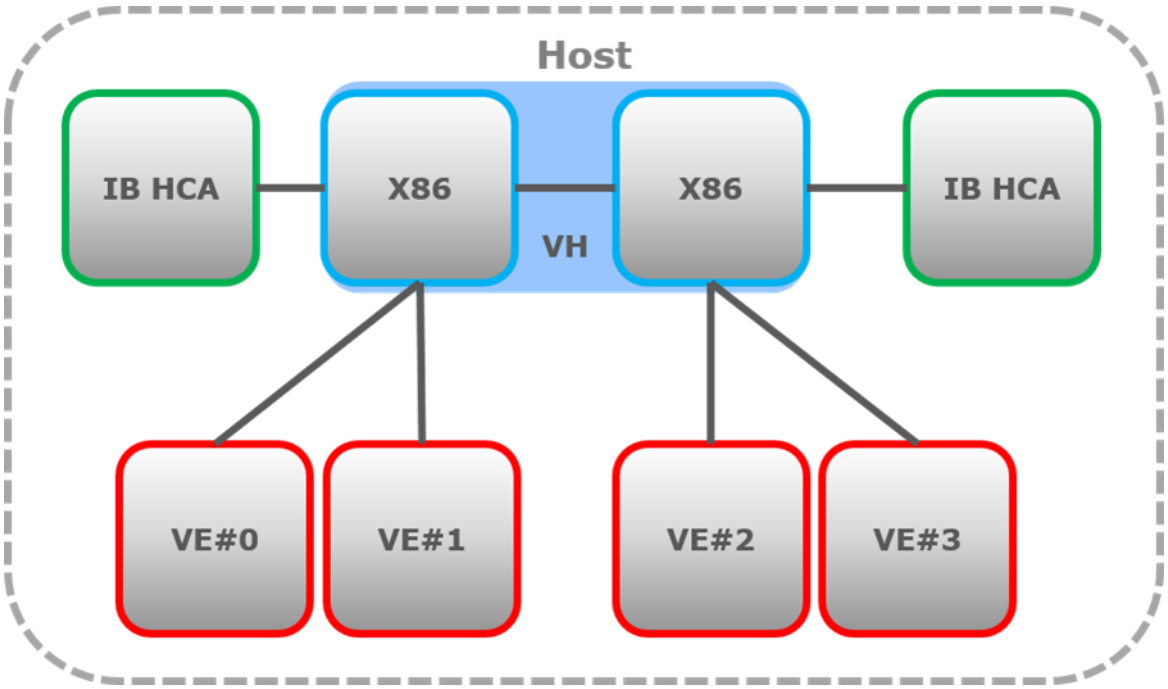

Figure 1 Configuration example of SX-Aurora TSUBASA

1.1 Confirmation of VE Composition

It's possible to acquire the composition situation of VE and HCA (IB) by the vecmd command.

```
$ /opt/nec/ve/bin/vecmd topo tree
Vector Engine MMM-Command v1.1.2
Command:
topo -N 0,1 tree
--------------------------------------------------------------------------------
SYS-1028GQ-TRT
(QPI Link)
+-80:00.0-+-80:02.0---82:00.0 [VE0] [SOCKET1]
          +-80:03.0---83:00.0 [VE1] [SOCKET1]
          +-80:01.0---81:00.0 [IB0] [SOCKET1] mlx5_0
--------------------------------------------------------------------------------
   Result: Success
```

A number part of indicated VE0 and VE1 is the VE number.

Chapter2 Compilation

2.1 Compilation of FORTRAN/C/C++

```
(For Fortran)
$ /opt/nec/ve/bin/nfort a.f90
(For C)
$ /opt/nec/ve/bin/ncc   a.c
(For C++)
$ /opt/nec/ve/bin/nc++  a.cpp
```

The option -fopenmp below enables the OpenMP features.

```
(For Fortran)
$ /opt/nec/ve/bin/nfort -fopenmp a.f90
(For C)
$ /opt/nec/ve/bin/ncc -fopenmp a.c
(For C++)
$ /opt/nec/ve/bin/nc++ -fopenmp a.cpp
```

2.2 Compilation of MPI Programs

Firstly, execute the following command each time you log in, in order to setup the MPI compilation environment. This setting is available until you log out.

```
(For bash)
$ source /opt/nec/<ve>/mpi/<version>/bin/necmpivars.sh
(For csh)
% source /opt/nec/<ve>/mpi/<version>/bin/necmpivars.csh
```

where **<version>** is the directory name corresponding to the version of NEC MPI you use, and **<ve>** is ve3 for VE30, otherwise ve.

Use the MPI compilation commands corresponding to each programing language to compile and link MPI programs as follows:

```
(For Fortran)
$ mpinfort a.f90
(For C)
$ mpincc   a.c
```

```
(For C++)
$ mpinc++  a.cpp
```

If you compile MPI programs executed on VH, specify the option -vh. Then the MPI program is compiled with gfortran, gcc, or g++.

```
(For Fortran)
$ mpinfort -vh a.f90
(For C)
$ mpincc   -vh a.c
(For C++)
$ mpinc++  -vh a.cpp
```

Chapter3　Program Execution

3.1　Interactive Program Execution

3.1.1　Execution of FORTRAN/C/C++ Programs

(1)　In the case of 1 VE models

Execute a program directly.

```
$ ./a.out
```

(2)　The way of specifying a particular VE number to run a program in the models with two or more VEs

The VE number can be specified with the command ve_exec -N or environment variable VE_NODE_NUMBER. The following examples specify VE#1.

- The way of using the command ve_exec -N

```
$ /opt/nec/ve/bin/ve_exec -N 1 a.out
```

- The way of using the environment variable

```
(For bash)
$ export VE_NODE_NUMBER=1
$ ./a.out
(For csh)
% setenv VE_NODE_NUMBER 1
% ./a.out
```

Note 1　When a program is executed without specifying a VE number like $ ./a.out, VE#0 is selected.

Note 2　When the command ve_exec -N and environment variable VE_NODE_NUMBER are used together, the command ve_exec -N takes precedence.

Note 3　The number of OpenMP threads can be specified with the environment variable OMP_NUM_THREADS or VE_OMP_NUM_THREADS. When both are specified, the environment variable VE_OMP_NUM_THREADS takes precedence for programs executed on VEs.

3.1.2 Execution of MPI Programs

Firstly, execute the following command each time you log in, in order to setup the MPI execution environment. This setting is available until you log out.

```
(For bash)
$ source /opt/nec/<ve>/mpi/<version>/bin/necmpivars.sh
(For csh)
% source /opt/nec/<ve>/mpi/<version>/bin/necmpivars.csh
```

(1) Execution on one VE

Specify an MPI executable file in the mpirun command or the mpiexec command, specifying the number of MPI processes to launch with the -np option and the VE number to use with the -ve option.

When the -np option is not specified, one process is launched.

When the -ve option is not specified, VE#0 is used.

The following command example executes an MPI program on VE#3 using 4 processes.

```
$ mpirun -ve 3 -np 4 ./a.out
```

(2) Execution on multiple VEs on a VH

Specify the range of VE numbers with the -ve option and the total number of processes to launch with the -np option

The following command example executes an MPI program on from VE#0 through VE#7, using 16 processes in total (2 processes per VE).

```
$ mpirun -ve 0-7 -np 16 ./a.out
```

(3) Execution on multiple VEs on multiple VHs

Specify the name of a VH with the -host option.

The following command example executes an MPI program on VE#0 and VE#1 on each of two VHs (host1 and host2), using 16 processes per VH (8 processes per VE, totally 32 processes).

```
$ mpirun -host host1 -ve 0-1 -np 16 -host host2 -ve 0-1 -np 16 ./a.out
```

(4) Hybrid execution on VHs and VEs

Following with -vh option, specify options for MPI processes executed on VH, for example, the number of MPI processes, an MPI executable file. Separate Specifications of MPI executable files on VH and VE with ":".

The following command example executes MPI program vh.out on host1 using 4 processes, and at the same time MPI program ve.out on VE#0 and VE#1 on each of two VHs (host1 and host2), using 16 processes per VH (8 processes per VE, totally 32 processes).

```
$mpirun -vh -host host1 -np 4 vh.out : -host host1 -ve 0-1 -np 16 -host host2 -ve 0-1 -np 16 ./ve.out
```

3.2 Batch Program Execution with NQSV

This section explains the way to execute a program of SX-Aurora TSUBASA using NQSV. The following examples only describe the basic procedure to execute a program. Please refer to "NEC Network Queuing System V (NQSV) User's Guide [Operation]" about details of NQSV.

3.2.1 Job Execution Type

NQSV supports both batch-type and interactive-type as job execution type.

- Batch-type

  It is executed by submitting a script, using qsub command.

- Interactive-type

  It is possible to execute job interactively, using qlogin command.

3.2.2 Execution of FORTRAN/C/C++ Programs

A script example of the FORTRAN/C/C++ when carrying out a batch execution. 1VE is used for SX-Aurora TSUBASA program.

```
(script.sh)
:
#PBS --cpunum-lhost=1  # Number of CPUs
#PBS --venum-lhost=1   # Number of VE
./a.out
```

qsub command is used to submit a job as follows.

```
$ /opt/nec/nqsv/bin/qsub script.sh
```

qlogin command is used to start a job as follows.

```
$ /opt/nec/nqsv/bin/qlogin --venum-lhost=1 …
$ ./a.out
```

Note The allocation of VEs automatically performed by NQSV. Therefore, the user don't designate environment variable VE_NODE_NUMBER and ve_exec -N.

3.2.3 Execution of MPI Programs

(1) Execution on specific VEs out of VEs assigned by NQSV

The following example shows how to execute an MPI program with 32 processes using logical VE#0 through VE#3 on logical host #0, and eight processes per VE.

```
(script2.sh)
:
#PBS --cpunum-lhost=1   # Number of CPUs
#PBS --venum-lhost=4    # Number of VEs
source /opt/nec/<ve>/mpi/<version>//bin/necmpivars.sh
mpirun -host 0 -ve 0-3 -np 32 ./a.out
```

It's put in by the qsub command as follows.

```
$ /opt/nec/nqsv/bin/qsub script2.sh
```

(2) Execution on all VEs assigned by NQSV

The following example shows how to execute an MPI program with 32 processes, on four logical hosts, eight VEs each logical hosts, and one process each VE.

```
(script3.sh)
:
#PBS -T necmpi
#PBS -b 4               # Number of logical hosts
#PBS --cpunum-lhost=1   # Number of CPUs
#PBS --venum-lhost=8    # Number of VEs per logical host
#PBS --use-hca=2        # Number of available HCAs
source /opt/nec/<ve>/mpi/<version>/bin/necmpivars.sh
mpirun -np 32 ./a.out
```

It's put in by the qsub command as follows.

```
$ /opt/nec/nqsv/bin/qsub script3.sh
```

(3) Hybrid execution on all VHs and VEs assigned by NQSV

The following example shows how to execute MPI program vh.out with 12 processes , on 4 logical hosts, and at the same time MPI program ve.out with 32 processes, on 4 logical hosts, 8 VEs each logical hosts, and 1 process each VE.

```
(script4.sh)
:
#PBS -T necmpi
#PBS -b 4                # Number of logical hosts
#PBS --cpunum-lhost=4    # Number of CPUs per logical host
#PBS --venum-lhost=8     # Number of VEs per logical host
#PBS --use-hca=2         # Number of available HCAs
source /opt/nec/<ve>/mpi/<version>/bin/necmpivars.sh
mpirun -vh -np 12 vh.out : -np 32 ./ve.out
```

It's put in by the qsub command as follows.

```
$ /opt/nec/nqsv/bin/qsub script4.sh
```

The specifications described above are available in the interactive job, too.

Note The allocation of VEs and VHs to MPI processes is automatically performed by NQSV and users do not need to explicitly specify them.

3.3 Program Execution under PBS

This section explains how to run programs for the SX-Aurora TSUBASA under PBS. The description assumes that PBS installed on the system has been configured for the SX-Aurora TSUBASA. Refer to the chapter “Support for NEC SX-Aurora TSUBASA” in “Altair PBS Professional Administrator’s Guide” for the configuration. This section illustrates the most basic usage. Refer to the chapter “Submitting Jobs to NEC SX-Aurora TSUBASA” in “Altair PBS Professional User’s Guide” for advanced usage.

3.3.1 Overview

Under PBS, you can submit a batch job by executing the command **qsub** specifying a jobscript file. To submit an interactive job, you can use the command **qsub -I** specifying a jobscript file. The status of jobs can be viewed with the command **qstat**. Deletion of jobs is accomplished with the command **qdel** specifying the job IDs. These commands are in the directory /opt/pbs/bin by default, and set the command search path appropriately.

In jobscript files, specify resources you use in the PBS directive starting with the prefix "#PBS " as the following example shows, in which the resources nves and mpiprocs specify the number of VEs and that of MPI processes, respectively, resulting in execution of eight MPI processes on four VEs. The PBS directive starting with "-l select" is called a selection directive.

```
#PBS -l select=nves=4:mpiprocs=8
```

A list of the resource requests in the form "*resource*=*value*" concatenated with the character "**:**" like "nves=4:mpiprocs=8" is called a chunk. The resources requested in a chunk are always allocated from a VH. Therefore, for example, the value of the resource nves shall be less than or equal to four on the VHs that have four VEs.
You can request multiple identical chunks by specifying "*number***:**" immediately before the chunk as the following selection directive shows, which requests four sets of the chunks, each of which specifies one VE and two MPI processes. A set of identical chunks is called a chunk set.

```
#PBS -l select=4:nves=1:mpiprocs=2
```

3.3.2 Execution of Fortran, C, or C++ Programs

The following jobscript runs an SX-Aurora TSUBASA program written in Fortran, C, or C++ using one VE.

```
#!/bin/bash
#PBS -l select=nves=1

./a.out
```

NOTE) Do not specify the environment variable VE_NODE_NUMBER or execute the command **ve_exec** with the option **-N** because the assignment of particular VEs is performed by PBS.

In the case of OpenMP programs, specify the number of threads using the resource ompthreads. The following jobscript runs an OpenMP program with eight threads using one VE.

```
#!/bin/bash
#PBS -l select=nves=1:ompthreads=8

./a.out
```

NOTE) Do not specify the environment variable OMP_NUM_THREADS because the value is automatically set by PBS and any user-defined values are overwritten.

3.3.3 Execution of MPI Programs

(1) Execution on VEs

The following jobscript runs 32 MPI processes using four VEs, each of which executes eight MPI processes. In the selection directive, the chunk “nves=1:mpiprocs=8” specifies the number of MPI processes executed on a VE and the leading “4:” indicates the number of VEs to use.

Also, specify the total number of MPI processes using the option -np in the mpirun command line.

```
#!/bin/bash
#PBS -l select=4:nves=1:mpiprocs=8

source /opt/nec/<ve>/mpi/<version>/bin/necmpivars.sh
mpirun -np 32 ./a.out
```

(2) Execution of Hybrid Parallel (MPI and OpenMP) Programs

The following jobscript runs 16 MPI processes with four threads each using eight VEs, each of which executes two MPI processes.

```
#!/bin/bash
#PBS -l select=8:nves=1:mpiprocs=2:ompthreads=4

source /opt/nec/<ve>/mpi/<version>/bin/necmpivars.sh
mpirun -np 16 ./a.out
```

(3) Execution of VH-VE Hybrid MPI Programs

When you execute MPI processes on VHs, specify where the MPI processes should run using the environment variable NEC_PROCESS_DIST. The following jobscript runs two MPI processes on a VH and four MPI processes on each of eight VEs, resulting in 34 MPI process execution in total.

```
#!/bin/bash
#PBS -l select=ncpus=2:mpiprocs=2+8:nves=1:mpiprocs=4
#PBS -v NEC_PROCESS_DIST=s2+4

source /opt/nec/<ve>/mpi/<version>/bin/necmpivars.sh
mpirun -vh -np 2 vh.out : -np 32 ./ve.out
```

In the selection directive, specify the number of CPU cores that run MPI processes on a VH using the resource ncpus like "ncpus=2:mpiprocs=2" above. You can specify different chunk sets such as "ncpus=2:mpiprocs=2" and "8:nves=1:mpiprocs=4" by concatenating them with the character "+" in the selection directive as the example shows. The environment variable NEC_PROCESS_DIST specifies the placement of MPI processes in every chunk in the selection directive. In the example above, the first chunk set specifies the number of MPI processes on a VH following the character "s", and the second the number of MPI processes on each VE.

Because the ranks of MPI processes are determined by the order of the chunks specified in the selection directive and environment variable NEC_PROCESS_DIST, the order of MPI processes on VHs and VEs in the PBS directives has to match that in the mpirun command line.

Chapter4　I/O Acceleration

When you set the environment variable described in this chapter and execute your program, your program's I/O will be accelerated.

4.1　ScaTeFS Direct I/O

When the read/write I/O size is larger than the defined value (1MB by default), a VE process performs the direct I/O to ScaTeFS using the library. Set the value of the environment variable VE_LD_PRELOAD to "libscatefsib" before executing VE programs.
Requirement:　ScaTeFS is installed and the ScaTeFS I/O client is set up in VHs.

```
(For bash)
$ export VE_LD_PRELOAD=libscatefsib.so.1
$ ./a.out
 (For csh)
% setenv VE_LD_PRELOAD libscatefsib.so.1
% ./a.out
```

When you execute programs with NQSV, please set the --use-hca option.

```
#!/bin/bash
#PBS -b 1
#PBS --venum-lhost=1
#PBS --use-hca=2        # Number of available HCAs

VE_LD_PRELOAD=libscatefsib.so.1 ./a.out
```

When you run programs under PBS, set the value of the environment variable VE_LD_PRELOAD in the jobscript as follows:

```
#!/bin/bash
#PBS -l select=nves=1

VE_LD_PRELOAD=libscatefsib.so.1 ./a.out
```

4.2 Accelerated I/O

The Accelerated I/O library improves I/O performance by efficient data transfer between VH and VE. The feature is enabled when the environment variable VE_ACC_IO[1] is 1[2], and it is disabled when the VE_ACC_IO is 0. Please set VE_ACC_IO before a user execute VE programs. Please note that the feature is unavailable for a static linked VE program.

Requirement: The system administrator reserves HugePages for the Accelerated I/O through the kernel parameter "vm.nr_hugepages".

Please refer to "SX-Aurora TSUBASA Installation Guide" about the number of HugePages for Accelerated I/O and configuration steps. The ve-set-hugepages service which is available after VEOS v2.9.1 configures the HugePages automatically.

```
 (For bash)
$ export VE_ACC_IO=1
$ ./a.out
 (For csh)
% setenv VE_ACC_IO 1
% ./a.out
```

When you execute programs with NQSV, please set the environment variable in the script for a batch execution.

```
#!/bin/bash
#PBS -b 1
#PBS --venum-lhost=1

export VE_ACC_IO=1
./a.out
```

---

[1] The VE_ACC_IO is available if the VEOS version is v2.3.0 or later. If the VEOS version is earlier than 2.3.0, the VE_ACC_IO is not available. In this case, set the environment variable VE_LD_PRELOAD to libveaccio.so.1. To confirm VEOS version, please use the following command.

```
$ rpm -q veos
veos-2.3.0-1.el7.x86_64
```

[2] The feature is enabled by default after VEOS v3.0.2.

When you run programs under PBS, set the value of the environment variable VE_ACC_IO in the jobscript as follows:

```
#!/bin/bash
#PBS -l select=nves=1

export VE_ACC_IO=1
./a.out
```

Chapter5　Performance Profiling

When confirming the execution performance of the program, the PROGINF function and the FTRACE function are used.

5.1　PROGINF Function

PROGINF provides program execution analysis information throughout the execution of program. After that YES or DETAIL is designated in environment variable VE_PROGINF and a program is executed. Performance information on the whole program is output at the time of an execution end of a program.

```
$ /opt/nec/ve/bin/ncc source.c
$ export VE_PROGINF=YES
$ ./a.out

          ********  Program  Information  ********
  Real Time (sec)                          :          100.795725
  User Time (sec)                          :          100.686826
  Vector Time (sec)                        :           41.125491
  Inst. Count                              :        82751792519
  V. Inst. Count                           :        11633744762
  V. Element Count                         :       881280485102
  V. Load Element Count                    :       268261733727
  FLOP count                               :       625104742151
  MOPS                                     :        11778.920848
  MOPS (Real)                              :        11765.127159
  MFLOPS                                   :         6209.015275
  MFLOPS (Real)                            :         6201.744217
  A. V. Length                             :           75.752090
  V. Op. Ratio (%)                         :           94.002859
  L1 Cache Miss (sec)                      :            6.364831
  VLD LLC Hit Element Ratio (%)            :           90.032527
  Memory Size Used (MB)                    :          918.000000
  Non Swappable Memory Size Used (MB)      :           84.000000

  Start Time (date)        :       Tue Nov 17 12:43:08 2020 JST
  End   Time (date)        :       Tue Nov 17 12:44:49 2020 JST
```

(Output may vary depending on the environment variables and VE models)

In the case of MPI programs, YES or DETAIL is designated in environment variable NMPI_PROGINF and a program is executed. As a result, performance information on the whole MPI program execution is output.

```
$ source /opt/nec/<ve>/mpi/<version>/bin/necmpivars.sh
$ mpincc source.c
$ export NMPI_PROGINF=YES
$ mpirun -np 4 -ve 0-1 ./a.out

MPI Program Information:
========================
Note: It is measured from MPI_Init till MPI_Finalize.
      [U,R] specifies the Universe and the Process Rank in the Universe.
      Times are given in seconds.


Global Data of 4 Vector processes       :          Min [U,R]          Max [U,R]      Average
=================================

Real Time (sec)                         :       258.752 [0,1]      258.769 [0,0]      258.760
User Time (sec)                         :       258.632 [0,0]      258.672 [0,3]      258.661
Vector Time (sec)                       :       163.308 [0,3]      165.063 [0,2]      164.282
Inst. Count                             : 255247993643 [0,0] 255529897274 [0,3] 255372547702
V. Inst. Count                          :  19183106540 [0,0]  19190366299 [0,3]  19186786385
V. Element Count                        : 731572775534 [0,2] 731612551928 [0,3] 731597913441
V. Load Element Count                   : 213554974007 [0,0] 213586395765 [0,3] 213566855461
FLOP Count                              : 580774521087 [0,3] 580807048542 [0,0] 580790784573
MOPS                                    :      4464.705 [0,2]      4465.784 [0,3]      4465.280
MOPS (Real)                             :      4462.927 [0,0]      4464.222 [0,3]      4463.583
MFLOPS                                  :      2245.220 [0,3]      2245.688 [0,0]      2245.373
MFLOPS (Real)                           :      2244.435 [0,3]      2244.588 [0,1]      2244.519
A. V. Length                            :        38.124 [0,3]        38.138 [0,0]        38.130
V. Op. Ratio (%)                        :        79.541 [0,3]        79.559 [0,0]        79.551
L1 Cache Miss (sec)                     :        36.603 [0,2]        38.208 [0,3]        37.331
VLD LLC Hit Element Ratio (%)           :        87.174 [0,1]        87.176 [0,2]        87.175
Memory Size Used (MB)                   :       677.000 [0,1]       933.000 [0,0]       741.000
Non Swappable Memory Size Used (MB)     :       115.000 [0,0]       179.000 [0,2]       131.000


Overall Data of 4 Vector processes
==================================

Real Time (sec)                         :       258.769
```

```
User Time (sec)                          :      1034.645
Vector Time (sec)                        :       657.127
GOPS                                     :        14.966
GOPS (Real)                              :        14.960
GFLOPS                                   :         8.981
GFLOPS (Real)                            :         8.978
Memory Size Used (GB)                    :         2.895
Non Swappable Memory Size Used (GB)      :         0.512


VE Card Data of 2 VEs
=====================

Memory Size Used (MB) Min                :      1354.000 [node=0,ve=1]
Memory Size Used (MB) Max                :      1610.000 [node=0,ve=0]
Memory Size Used (MB) Avg                :      1482.000
Non Swappable Memory Size Used (MB) Min  :       230.000 [node=0,ve=1]
Non Swappable Memory Size Used (MB) Max  :       294.000 [node=0,ve=0]
Non Swappable Memory Size Used (MB) Avg  :       262.000
```

(Output may vary depending on the environment variables and VE models)

5.2 FTRACE Function

FTRACE measures performance information of every function and output it. When using the FTRACE function, a program is compiled with the -ftrace option and executed. An analysis information file (ftrace.out) is output after the execution of a program. To confirm the performance information, an analysis information file (ftrace.out) is designated and the ftrace command is carried out.

```
$ /opt/nec/ve/bin/nfort -ftrace source.f90
$ ./a.out
$ /opt/nec/ve/bin/ftrace -f ftrace.out

*----------------------*
  FTRACE ANALYSIS LIST
*----------------------*

Execution Date : Tue May  8 15:22:15 2018 JST
Total CPU Time : 0:03'21"561 (201.561 sec.)

FREQUENCY   EXCLUSIVE        AVER.TIME      MOPS    MFLOPS   V.OP   AVER.     VECTOR L1CACHE CPU PORT VLD LLC
```

```
PROC.NAME
          TIME[sec](  %)     [msec]                  RATIO V.LEN      TIME     MISS     CONF HIT E.%
     25100  96.105( 47.7)      3.829     1455.0     728.7 39.20   8.0    46.967   17.785    0.314 93.16 funcA
     25100  82.091( 40.7)      3.271     1703.3     853.1 36.95   7.6    46.462   18.024    0.314 98.29 funcB
  13124848   7.032(  3.5)      0.001      772.7     229.6  0.00   0.0     0.000    4.184    0.000  0.00 funcC
       253   6.007(  3.0)     23.745    35379.0   19138.0 97.21  99.8     5.568    0.181    1.128 89.40 funcD
     25100   3.684(  1.8)      0.147    45327.6   21673.3 98.35 114.3     3.455    0.218    1.076 94.75 funcE
     25100   3.611(  1.8)      0.144    51034.2   25382.3 98.37 111.0     3.451    0.143    1.076 88.64 funcF
         2   2.447(  1.2)   1223.578     1262.9      79.3  0.00   0.0     0.000    1.044    0.000  0.00 funcG
         2   0.317(  0.2)    158.395    32624.9   11884.9 96.79  99.1     0.272    0.034    0.000  7.07 funcH
         1   0.217(  0.1)    216.946     1318.8      69.1  0.00   0.0     0.000    0.089    0.000  0.00 funcI
         2   0.025(  0.0)     12.516     1254.8       0.0  0.00   0.0     0.000    0.011    0.000  0.00 funcJ
         1   0.019(  0.0)     19.367    54199.2   33675.0 97.87 100.3     0.019    0.000    0.010 94.02 funcK
         4   0.004(  0.0)      0.948    57592.4   24101.4 97.88 121.4     0.004    0.000    0.000  4.72 funcL
         1   0.001(  0.0)      0.861      517.9       3.2  0.00   0.0     0.000    0.000    0.000  0.00 funcM
-------------------------------------------------------------------------------------------------------------
  13225514 201.561(100.0)      0.015     4286.1    2147.5 76.91  34.7   106.197   41.712    3.917 89.99 total
```

(Output may vary depending on VE models)

In case of a MPI program, FTRACE Function is available for MPI program executed on VE. When using the FTRACE function, a program is compiled with the -ftrace option and executed. After the execution of a program, performance information is output by a different analysis information file (*1) every MPI process. When designating 1 analysis file as the ftrace command, performance information on the MPI process is output. When designating all analysis information files, measurement information on the whole MPI program execution is output.

(*1) The file name will be "ftrace.out.group ID.rank number". The group ID and the rank number are respectively the value of environment variable MPIUNIVERSE and MPIRANK in NEC MPI.

```
$ source /opt/nec/<ve>/mpi/<version>/bin/necmpivars.sh
$ mpinfort -ftrace source.f90
$ mpirun -np 4 ./a.out
$ ls ftrace.out.*
ftrace.out.0.0 ftrace.out.0.1 ftrace.out.0.2 ftrace.out.0.3
$ /opt/nec/ve/bin/ftrace -f ftrace.out.* (A result of measurement of the whole MPI program execution is output.)
```

```
*----------------------*
  FTRACE ANALYSIS LIST
*----------------------*

Execution Date : Sat Feb 17 12:44:49 2018 JST
Total CPU Time : 0:03'24"569 (204.569 sec.)

FREQUENCY  EXCLUSIVE       AVER.TIME     MOPS    MFLOPS  V.OP  AVER.    VECTOR L1CACHE CPU PORT VLD LLC PROC.NAME
           TIME[sec](  %)    [msec]                     RATIO V.LEN      TIME    MISS     CONF HIT E.%

     1012    49.093( 24.0)    48.511  23317.2  14001.4  96.97  83.2    42.132   5.511    0.000   80.32 funcA
   160640    37.475( 18.3)     0.233  17874.6   9985.9  95.22  52.2    34.223   1.973    2.166   96.84 funcB
   160640    30.515( 14.9)     0.190  22141.8  12263.7  95.50  52.8    29.272   0.191    2.544   93.23 funcC
   160640    23.434( 11.5)     0.146  44919.9  22923.2  97.75  98.5    21.869   0.741    4.590   97.82 funcD
   160640    22.462( 11.0)     0.140  42924.5  21989.6  97.73  99.4    20.951   1.212    4.590   96.91 funcE
 53562928    15.371(  7.5)     0.000   1819.0    742.2   0.00   0.0     0.000   1.253    0.000    0.00 funcG
        8    14.266(  7.0)  1783.201   1077.3     55.7   0.00   0.0     0.000   4.480    0.000    0.00 funcH
   642560     5.641(  2.8)     0.009    487.7      0.2  46.45  35.1     1.833   1.609    0.007   91.68 funcF
     2032     2.477(  1.2)     1.219    667.1      0.0  89.97  28.5     2.218   0.041    0.015   70.42 funcI
        8     1.971(  1.0)   246.398  21586.7   7823.4  96.21  79.6     1.650   0.271    0.000    2.58 funcJ
-------------------------------------------------------------------------------------------------------------------
 54851346   204.569(100.0)     0.004  22508.5  12210.7  95.64  76.5   154.524  17.740   13.916   90.29 total

   ELAPSED     COMM.TIME  COMM.TIME   IDLE TIME  IDLE TIME  AVER.LEN      COUNT  TOTAL LEN PROC.NAME
   TIME[sec]      [sec]  / ELAPSED      [sec]  / ELAPSED    [byte]                 [byte]

      12.444      0.000                 0.000                  0.0          0       0.0  funcA
       9.420      0.000                 0.000                  0.0          0       0.0  funcB
       7.946      0.000                 0.000                  0.0          0       0.0  funcG
       7.688      0.000                 0.000                  0.0          0       0.0  funcC
       7.372      0.000                 0.000                  0.0          0       0.0  funcH
       5.897      0.000                 0.000                  0.0          0       0.0  funcD
       5.653      0.000                 0.000                  0.0          0       0.0  funcE
       1.699      1.475                 0.756                  3.1K    642560       1.9G funcF
       1.073      1.054                 0.987                  1.0M      4064       4.0G funcI
       0.704      0.045                 0.045                 80.0         4     320.0  funcK
-------------------------------------------------------------------------------------------------------------------

FREQUENCY  EXCLUSIVE       AVER.TIME     MOPS    MFLOPS  V.OP  AVER.    VECTOR L1CACHE CPU PORT VLD LLC PROC.NAME
           TIME[sec](  %)    [msec]                     RATIO V.LEN      TIME    MISS     CONF HIT E.%

     1012    49.093( 24.0)    48.511  23317.2  14001.4  96.97  83.2    42.132   5.511    0.000   80.32 funcA
      253    12.089           47.784  23666.9  14215.9  97.00  83.2    10.431   1.352    0.000   79.40  0.0
      253    12.442           49.177  23009.2  13811.8  96.93  83.2    10.617   1.406    0.000   81.26  0.1
```

```
      253     12.118           47.899  23607.4  14180.5  97.00  83.2    10.463   1.349    0.000   79.36  0.2
      253     12.444           49.185  23002.8  13808.2  96.93  83.2    10.622   1.404    0.000   81.26  0.3
:
-------------------------------------------------------------------------------------------------------------

 54851346   204.569(100.0)     0.004  22508.5  12210.7  95.64  76.5   154.524  17.740   13.916   90.29 total

   ELAPSED      COMM.TIME  COMM.TIME   IDLE TIME  IDLE TIME  AVER.LEN      COUNT  TOTAL LEN PROC.NAME
   TIME[sec]        [sec]  / ELAPSED        [sec]  / ELAPSED    [byte]               [byte]

      12.444        0.000                  0.000                 0.0           0        0.0  funcA
      12.090        0.000      0.000       0.000      0.000      0.0           0        0.0   0.0
      12.442        0.000      0.000       0.000      0.000      0.0           0        0.0   0.1
      12.119        0.000      0.000       0.000      0.000      0.0           0        0.0   0.2
      12.444        0.000      0.000       0.000      0.000      0.0           0        0.0   0.3
:
```

(Output may vary depending on VE models)

5.3 Profiler

When a source file is compiled and linked with the -pg option, the performance measurement file (gmon.out) is output after the program is executed. The file gmon.out can be displayed and analyzed by the ngprof command.

```
$ /opt/nec/ve/bin/nfort -pg a.f90
$ ./a.out
$ /opt/nec/ve/bin/ngprof ./a.out
(The performance information is output)
```

If the profiler is used for an MPI program, the environment variable VE_GMON_OUT_PREFIX and GMON_OUT_PREFIX to specify an individual file name for each MPI procsss can be used to avoid the gmon.out to be overwritten by MPI processes. To change the filename of gmon.out output by programs executed on VE, the environment variable VE_GMON_OUT_PREFIX is specified. To change the filename of gmon.out output by programs executed on VH, the environment variable GMON_OUT_PREFIX is specified.
The following shell script, gprof-mpi.sh, helps save the performance measurement file into gmon.out.<MPI-universe>:<MPI-rank>.<pid> for each MPI process.

```
 (gprof-mpi.sh)
#!/bin/bash
# change the performance measurement file name to gmon.out.<MPI-universe>:<MPI-rank>.<pid>
export VE_GMON_OUT_PREFIX=gmon.out.${MPIUNIVERSE}:${MPIRANK}
export GMON_OUT_PREFIX=gmon.out.${MPIUNIVERSE}:${MPIRANK}
exec $*

(setup MPI environment)
$ source /opt/nec/<ve>/mpi/<version>/bin/necmpivars.sh

(compile MPI program)
$ mpincc -pg a.c -o ve.out
$ mpincc -vh -pg a.c -o vh.out

(run a.out through gprof-mpi.sh)
$ mpirun -np 1 ./gprof-mpi.sh ./ve.out : -vh -np 1 ./gprof-mpi.sh ./vh.out

$ ls gmon.out.*
gmon.out.0:0.19390  gmon.out.0:1.19391

(show analyzed information for MPI rank 0 executed on VE)
$ /opt/nec/ve/bin/ngprof ve.out gmon.out.0:0.19390

(show analyzed information for MPI rank 1 executed on VH)
```

```
$ /usr/bin/gprof vh.out gmon.out.0:1.19391
```

Chapter6　General Questions and Answers

(1)　Are commands which are well known in Linux available?

Answer : Yes. For example, the following commands for SX-Aurora TSUBASA are available.

ve-ps, ve-pmap, ve-time, ve-gdb, ve-automake, ve-top, ve-free, ve-vmstat, etc.

These commands are present in /opt/nec/ve/bin.

(2)　Is there a way to examine whether an executable file is for SX-Aurora TSUBASA?

Answer : It is possible to check it by the nreadelf command.

```
$ /opt/nec/ve/bin/nreadelf -h a.out
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           NEC VE architecture
  Version:                           0x1
  Entry point address:               0x600000004580
  Start of program headers:          64 (bytes into file)
  Start of section headers:          4760248 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         7
  Size of section headers:           64 (bytes)
  Number of section headers:         27
  Section header string table index: 24
```

(3)　Is there a way to check the state of the process which is being carried out on VE?

Answer : It is possible to refer to the state of the process which is being carried out in VE by the ve-ps command for SX-Aurora TSUBASA.

```
$ export -n VE_NODE_NUMBER; /opt/nec/ve/bin/ve-ps -ef
VE Node: 6
UID         PID  PPID  C STIME TTY          TIME CMD
User1     30970     1 75 17:44 ?        00:00:02 ./IMB-MPI1
VE Node: 7
UID         PID  PPID  C STIME TTY          TIME CMD
User1     30977     1 59 17:44 ?        00:00:02 ./IMB-MPI1
VE Node: 5
UID         PID  PPID  C STIME TTY          TIME CMD
User1     30958     1 99 17:44 ?        00:00:02 ./IMB-MPI1
VE Node: 4
UID         PID  PPID  C STIME TTY          TIME CMD
User1     30957     1 99 17:44 ?        00:00:02 ./IMB-MPI1
VE Node: 2
UID         PID  PPID  C STIME TTY          TIME CMD
User1     30919     1  0 17:44 ?        00:00:02 ./IMB-MPI1
VE Node: 3
UID         PID  PPID  C STIME TTY          TIME CMD
User1     30920     1 99 17:44 ?        00:00:02 ./IMB-MPI1
VE Node: 1
UID         PID  PPID  C STIME TTY          TIME CMD
User1     30918     1  0 17:44 ?        00:00:02 ./IMB-MPI1
VE Node: 0
UID         PID  PPID  C STIME TTY          TIME CMD
User1     30917     1  0 17:44 ?        00:00:02 ./IMB-MPI1
```

When in case of use NQSV, use the qstat command.

```
$/opt/nec/nqsv/bin/qstat
RequestID       ReqName  UserName Queue     Pri STT S   Memory      CPU   Elapse R H M Jobs
--------------- -------- -------- -------- ---- --- - -------- -------- -------- - - - ----
48682.bsv00     run1.sh  user1 batchq          0 RUN -    4.71M     0.00      126 Y Y Y    1
```

(4) Is there a way to check whether an object was created for musl-libc or glibc?

Answer : You can use /opt/nec/ve/bin/ve-libc-check script as below.

```
$ /opt/nec/ve/bin/ve-libc-check ./a.out
  This is compiled with musl-libc: /home/userxxx/a.out
```

If a specified object was compiled with musl-libc, the message in the above box is shown. If a specified object was compiled with glibc, the script doesn't show any

message.

Note 1 musl-libc is obsoleted at the end of March, 2019.
If you are now using musl-libc environment, please migrate to glibc environment by the following procedure.
- Install new SX-Aurora TSUBASA software of glibc environment.
- Recompile your programs in the glibc environment.

Please see the Installation guide for detail.

Note 2 The script "ve-libc-check" can't determine used library for a specified object whose source file is "*.s". Additionally, "ve-libc-check" can't determine used library for VE program, which was compiled by glibc and dynamically links or loads a library compiled by musl-libc.

(5) What kind of environment variables can I use?

Answer : For example, you can use the following variables.

VE_NODE_NUMBER

It specifies VE node number on which a program will be executed.

VE_LD_LIBRARY_PATH

This environment variable provides a library path for finding dynamic libraries.

VE_LD_PRELOAD

This environment variable sets the pre-loading shared libraries' path for dynamic linker.

(6) How to set library search paths?

Answer :

Add a setting file whose name is "*.conf" to /etc/opt/nec/ve/ld.so.conf.d, then execute ldconfig for SX-Aurora TSUBASA.

```
(Example)
$ cat /etc/opt/nec/ve/ld.so.conf.d/local_lib.conf
/usr/local/local_lib/lib
$ sudo /opt/nec/ve/glibc/sbin/ldconfig
```

(7) Can I use gdb for debugging of VE program?

Answer : Yes. ve-gdb for SX-Aurora TSUBASA is available.

Appendix A History

A.1 History Table

Aug. 2018	Rev. 1
Dec. 2018	Rev. 2
May. 2019	Rev. 3
Sep. 2019	Rev. 4
Jan. 2020	Rev. 5
Dec. 2020	Rev. 6
May. 2021	Rev. 7
Jun. 2023	Rev. 8

A.2 Change Notes

Rev. 5

- 4.2 Accelerated I/O
  - The way to enable accelerated I/O is changed

Rev. 6

- 3.2.3 Execution of MPI Programs
  - The example of use with NQSV is changed
- 4.1 ScaTeFS Direct I/O
  - The example of use with NQSV is changed
- 4.2 Accelerated I/O
  - The requirement is changed
- 5.1 PROGINF Function
  - The output example is changed

Rev. 7

- 3.3 Program Execution under PBS
  - How to run jobs under PBS is illustrated.
- 4.1 ScaTeFS Direct I/O
  - How to use the ScaTeFS direct I/O under PBS is explained.
- 4.2 Accelerated I/O
  - How to use the accelerated I/O under PBS is explained.

Rev. 8

- Chapters
  - The descriptions for VE30 are added.
- 4.2 Accelerated I/O
  - The description for VEOS v3.0.2 environment is added.
- 6 General Questions and Answers
  - Updated VE command names by adding prefix "ve-" before the commands.

SX-Aurora TSUBASA System Software

SX-Aurora TSUBASA

Program Execution Quick Guide

8th Edition Jun. 2023

NEC Corporation