SX-Aurora TSUBASA

Program Execution Quick Guide

SX-Aurora TSUBASA

Proprietary Notice

Preface

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

This guide assumes the following installation, setup and knowledge.

- VEOS and necessary software installation are completed.
- Users are able to log in to a system and to use a job scheduler (NEC Network Queuing System V : NQSV)
- Users have knowledge of Fortran compiler (nfort), C compiler (ncc), C++ compiler (nc++) and NEC MPI.

It corresponds to version veos-2.0.0-1.el7.x86_64 and beyond.
The version of veos can be confirmed by the following way.

```
$ rpm -q veos
veos-2.0.0-1.el7.x86_64
```

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.

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.

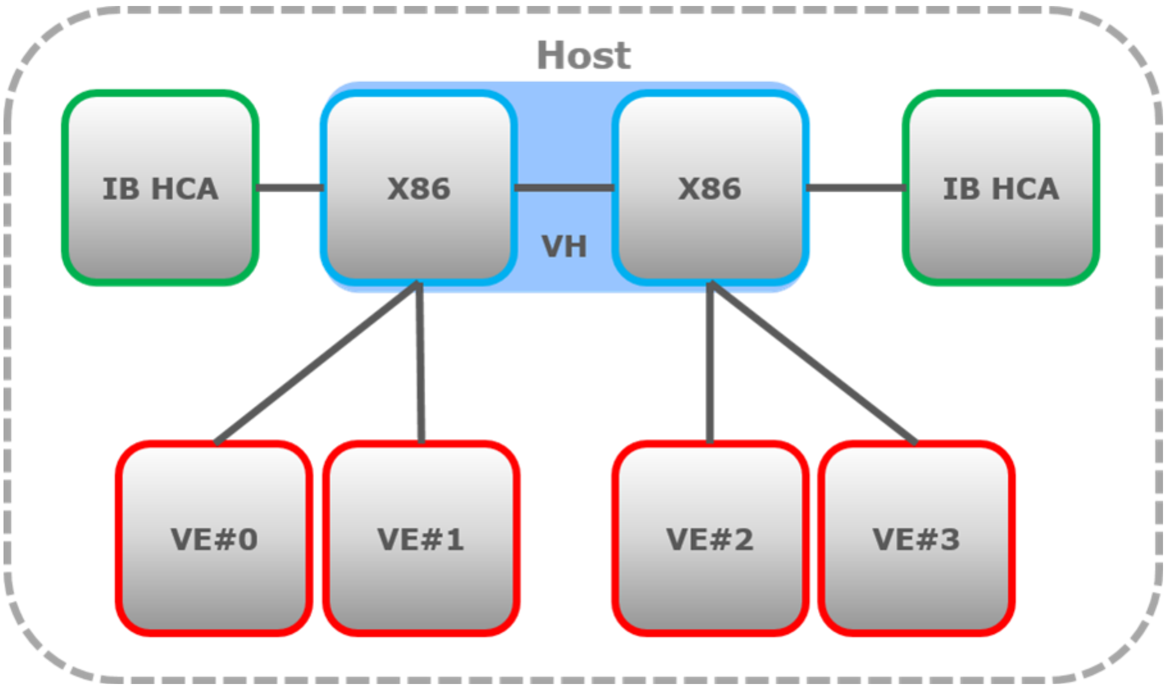

Figure 1 Configuration example of SX-Aurora TSUBASA

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
```

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 please select a matching NEC MPI version compatible with a compiler version to be used.

- NEC MPI version 2.x.x for a compiler version 2.y.y
- NEC MPI version 1.x.x for a compiler version 1.y.y

A compiler option --version allows you to find the compiler version. If the version is 2.x.x, the packages support glibc. If the version is 1.x.x, the packages support musl-libc.

```
(For the case that a compiler version is 1.6.0 and the version of NEC MPI version 1.0.1 must be used)
$ /opt/nec/ve/bin/ncc --version
 ncc (NCC) 1.6.0 (Build 12:05:08 Nov  5 2018)
Copyright (C) 2018 NEC Corporation.

$ ls -l /opt/nec/ve/mpi
drwxr-xr-x. 5 root root  69 11月  8 11:10 1.0.1
drwxr-xr-x. 5 root root  69 11月 27 09:25 2.0.0
drwxr-xr-x. 2 root root 133 11月 27 09:25 libexec

$ source /opt/nec/ve/mpi/1.0.1/bin/necmpivars.sh
 (For the case in which the version of NEC MPI is 1.0.1)
 $ source /opt/nec/ve/mpi/1.0.1/bin/necmpivars.sh
```

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
```

Chapter3 Program Execution

3.1 Interactive Program Execution

3.1.1 Execution of FORTRAN/C/C++ Programs

(1) In case of 1VE model

Execute a program directly.

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

(2) The way to designate the VE number besides the 1VE model and execute a program.

There are an occasion using the ve_exec command and a way to designate it in environment variable VE_NODE_NUMBER. The following example is the way to use VE#1.

- The way of using ve_exec

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

- The way of designated as an environment variable

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

Note 1 When carrying out with $ ./a.out without setting an environment variable VE_NODE_NUMBER, a.out is carried out in VE#0.

Note 2 When designating the VE number at the same time at an environment variable VE_NODE_NUMBER and ve_exec -N, designation of ve_exec -N is given priority to.

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 four 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 (two 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 (eight processes per

VE, totally 32 processes).

```
$ mpirun -host host1 -ve 0-1 -np 16 -host host2 -ve 0-1 -np 16 ./a.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 --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 multiple VEs on a VH

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

```
(script2.sh)
:
#PBS --venum-lhost=4     # Number of VEs
source /opt/nec/ve/mpi/2.0.0/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 multiple VEs on multiple VHs

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 --venum-lhost=8     # Number of VEs per logical host
#PBS --use-hca=1         # Number of HCAs
source /opt/nec/ve/mpi/2.0.0/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
```

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.

Chapter4 I/O Acceleration

When you set the appropriate library to environment variable "VE_LD_PRELOAD" and execute your program, your program's I/O will be accelerated.

4.1 ScaTeFS Direct I/O

When read/write I/O size is larger than defined value (default value is 1MB), VE process with the library performs direct I/O to ScaTeFS. Set "libscatefsib" to VE_LD_PRELOAD before executing VE program.
Requirement: ScaTeFS is installed and ScaTeFS I/O client is set up in VH.

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

4.2 Accelerated I/O

Accelerated I/O library provides a high speed I/O to VE program which doesn't use InfiniBand. Using the library allows VE program to use I/O path for InfiniBand which is used by MPI program. Set "libveaccio" VE_LD_PRELOAD before executing VE program.
Requirement: VH isn't connected to the other VH by InfiniBand, and VH doesn't use ScaTeFS.
The kernel parameter "vm.nr_hugepages" is more than or equal to 256 per VE
(Please see sysctl(8) man page to set the kernel parameter)

```
(For bash)
$ export VE_LD_PRELOAD=libveaccio.so.1
$ ./a.out
 (For csh)
% setenv VE_LD_PRELOAD libveaccio.so.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. When the PROGINF function is used, the -proginf option is designated and a program is compiled. 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 -proginf 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

  Start Time (date)       :        Sat Feb 17 12:43:08 2018 JST
  End   Time (date)       :        Sat Feb 17 12:44:49 2018 JST
```

When compiling of MPI program, the -proginf option is unnecessary. YES 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.

```
$ mpincc source.c
$ export NMPI_PROGINF=YES
$ mpirun -np 4 ./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 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)                  :      912.000 [0,0]      912.000 [0,0]      912.000


Overall Data:
=============

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)                 :        3.562
```

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
```

In case of a MPI 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.

```
$ 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
:
```

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 to specify an individual file name for each MPI procsss can be used to avoid the gmon.out to be overwritten by MPI processes. For example, 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}
exec $*

(run a.out through gprof-mpi.sh)
$ mpirun -np 2 ./gprof-mpi.sh ./a.out

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

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

(show analyzed information for MIP rank 1)
$ /opt/nec/ve/bin/ngprof gmon.out.0:1.19391
```

Note

The profiler can be used with version 2.0.0 or later of Fortran compiler, C/C++ compiler and NEC MPI.

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.

ps, pmap, time, gdb, automake, top, free, 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 ps command for SX-Aurora TSUBASA.

```
$ export -n VE_NODE_NUMBER; /opt/nec/ve/bin/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 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 : See the following methods for each libc environment.

✧ glibc

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
```

✧ musl-libc

Add library paths to /etc/opt/nec/ve/musl/ld-musl-ve.path.

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

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

Appendix A History

A.1 History Table

Aug. 2018	Rev. 1
Dec. 2018	Rev. 2

A.2 Change Notes

Rev.2

- 2.2 Instructions to use an MPI version for glibc and musl-libc added.
- Added Chapter 4 I/O Acceleration
- Added 5.3 Profiler
- Added items in Chapter 6

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

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

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

  (6) How to set library search paths?

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