NEC

SX-Aurora TSUBASA

SX-Aurora TSUBASA
VEOS NUMA Mode Guide
for Partitioning Mode

Proprietary Notice

The information disclosed in this document is the property of NEC Corporation (NEC) and/or its licensors. NEC and/or its licensors, as appropriate, reserve all patent, copyright, and other proprietary rights to this document, including all design, manufacturing, reproduction, use and sales rights thereto, except to the extent said rights are expressly granted to others.

The information in this document is subject to change at any time, without notice.

(1) Linux is the registered trademark of Linus Torvalds in the United States and other countries.

(2) In addition, mentioned company name and product name are a registered trademark or a trademark of each company.

Preface

The “SX-Aurora TSUBASA VEOS NUMA Mode Guide for Partitioning Mode” is a document for those using partitioning mode of Vector Engine installed into SX-Aurora TSUBASA. To utilize partitioning mode, VEOS supports NUMA mode. So, this document explains the basic usage of VEOS NUMA mode.

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.

VEOS NUMA mode is available, when veos-2.1.0-1.el7.x86_64 or later and mmm-1.2.13-0.el7.x86_64 or later are installed.
The versions of VEOS and MMM can be confirmed by the following way.

```
$ rpm -q veos
veos-2.1.0-1.el7.x86_64
$ rpm -q mmm
mmm-1.2.13-0.el7.x86_64
```

NEC MPI runtime options for NUMA mode are available, when nec-mpi-runtime-2.3.0-1.el7.x86_64 or later is installed.
The versions of NEC MPI can be confirmed by the following way.

```
$ rpm -q nec-mpi-runtime
nec-mpi-runtime-2.3.0-1.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.
VEOS	It is the software running on Linux/Vector Host, providing OS functionality for VE programs running on Vector Engine
MPI	An acronym 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.
NUMA	An acronym of Non-Uniform Memory Access
Partitioning mode	It is one of VE HW setting in which the VE cores, last level caches and its main memory are partitioned into two segments
NUMA mode	It is one of VEOS setting which controls VE when partitioning mode is enabled.
Normal mode	It means that partitioning mode is disabled. It also means that NUMA mode is disabled.

Contents

List of figures

Chapter1　Partitioning Mode / NUMA Mode

For better performance of your programs, VE supports the partitioning mode in which the VE cores, last level cache (LLC) and its main memory are partitioned into two segments. One of advantages of partitioning mode is avoiding conflicts of LLC. One of disadvantages of partitioning mode is that one VE core utilizes only the half bandwidth of main memory.

VEOS is a Linux based software. Linux supports system call and commands for NUMA. Therefore, VEOS treats two segments in partitioning mode as two NUMA nodes, in order to utilize these system call interfaces and commands. The mode of VEOS corresponding to “partitioning mode” is “NUMA mode”.

The term “Normal mode” means that partitioning mode is disabled. It also means that NUMA mode is disabled.

Figure 1 shows the difference between normal mode and portioning mode. The number of cores and the size of memory are examples and may be different.

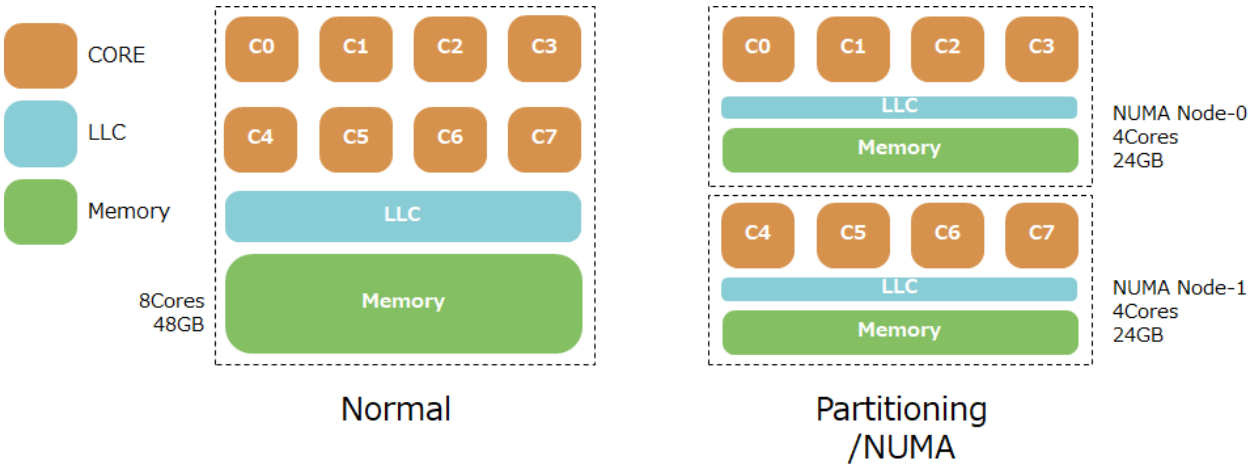

Figure 1 Overview of VE operation mode

The normal mode is the best for almost all of your programs because your programs can use all cores and memory of a VE card. Therefore the default mode is normal. But some programs may be faster on the NUMA mode. So, the mode is configurable.

VEOS NUMA mode is supported since VEOS version 2.1.

Chapter2　Changing Mode

This chapter explains how to change and check the mode.

Note　There are some command lines started with '#' prompt in this chapter. They should be executed with super user privileges.

2.1　Mode Change

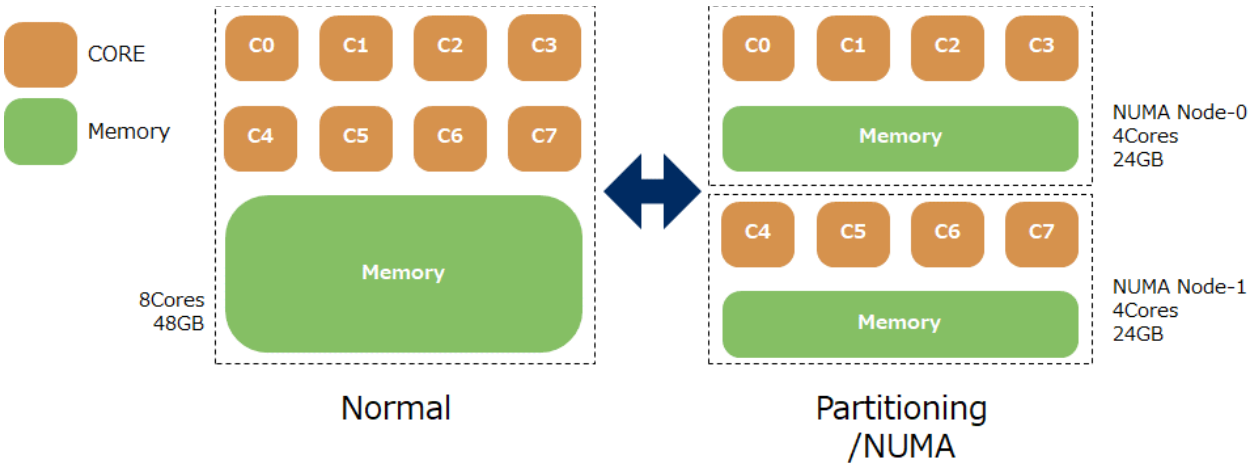

Figure 2 Changing VE operation mode

You can change the mode by **/opt/nec/ve/bin/vecmd**. When you change the mode to partitioning mode (NUMA mode), please execute the below commands. At the place of <VE node number>, please specify the number of VE node which you want to change the mode of. When you don't specify "-N <VE node number>" option, vecmd command changes setting of all VE node.

VE10/VE20

```
# /opt/nec/ve/bin/vecmd -N <VE node number> vconfig set partitioning_mode on
# /opt/nec/ve/bin/vecmd -N <VE node number> state set off
# /opt/nec/ve/bin/vecmd -N <VE node number> state set mnt
# /opt/nec/ve/bin/vecmd -N <VE node number> reset card
```

VE30

```
# systemctl stop ve-os-launcher@<VE node number>
# /opt/nec/ve/bin/vecmd -N <VE node number> partition on
# systemctl start ve-os-launcher@<VE node number>
```

When you change the mode to normal mode, please execute the below commands:

VE10/VE20

```
# /opt/nec/ve/bin/vecmd -N <VE node number> vconfig set partitioning_mode off
# /opt/nec/ve/bin/vecmd -N <VE node number> state set off
# /opt/nec/ve/bin/vecmd -N <VE node number> state set mnt
# /opt/nec/ve/bin/vecmd -N <VE node number> reset card
```

VE30

```
# systemctl stop ve-os-launcher@<VE node number>
# /opt/nec/ve/bin/vecmd -N <VE node number> partition off
# systemctl start ve-os-launcher@<VE node number>
```

The configuration of VE operation mode is preserved through reboot of VH/Linux.

2.2 Mode Check

You can check the mode by **/opt/nec/ve/bin/venumainfo** command.

In case of normal mode, **venumainfo** displays “available: 1 nodes(0)“. Additionally, the command displays information of whole VE as NUMA node 0.

```
$ /opt/nec/ve/bin/venumainfo
VE Node: 0
available: 1 nodes(0)
node 0 cpus: 0 1 2 3 4 5 6 7
node 0 size: 49152 MB
node 0 free: 49024 MB
```

In case of NUMA mode, **venumainfo** displays “available: 2 nodes(0-1)”. Additionally, it displays information of NUMA node 0 and NUMA node 1.

```
$ /opt/nec/ve/bin/venumainfo
VE Node: 0
available: 2 nodes(0-1)
node 0 cpus: 0 1 2 3
node 0 size: 24576 MB
node 0 free: 24448 MB
node 1 cpus: 4 5 6 7
node 1 size: 24576 MB
node 1 free: 24576 MB
```

Please note the number of cores and size of memory in the above are example.

Chapter3　Suitable Mode for Your Program

If your programs is composed of one process which has threads whose number is less than or equal to the half number of installed cores, the normal mode is the best because total memory bandwidth of the normal mode is better than the NUMA mode (partitioning mode) for such process. If your program is composed of multi processes or threads, it is difficult to say clearly which the best is. But generally we can say that the NUMA mode is the best if your program has the following implementations.

3.1　Multi-process Program

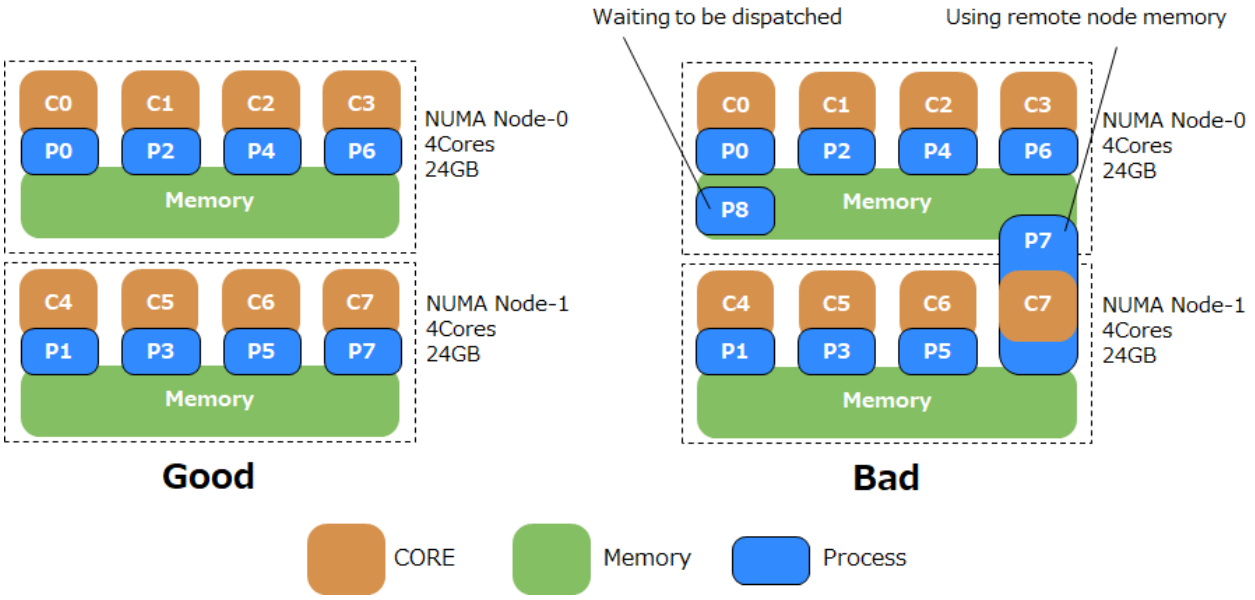

Figure 3 Example of execution of multi-process program

Note　**N** is the number of CPU cores on the NUMA node.

- Good for NUMA
  - Each node has only **N** or less processes.
  - Total size of used memory on each NUMA node is less than the memory size of the NUMA node.
- Bad for NUMA
  - A node has **N**+1 or more processes. Then context switching of some processes is necessary.
  - Total size of used memory on a NUMA node is greater than the memory size of the

NUMA node. Then the memory of the NUMA node where the process is not running will be allocated and used.

3.2 Multi-threaded Program

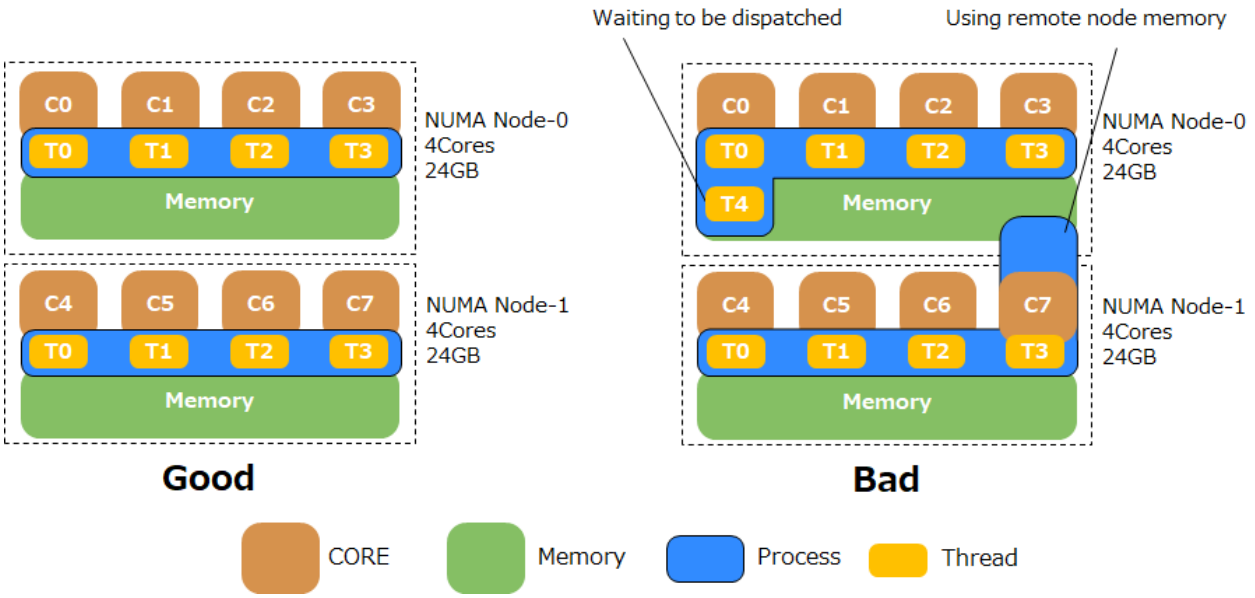

Figure 4 Example of execution of multi-threaded program

Note N is the number of CPU cores on the NUMA node.
It is impossible to execute the multi-threaded program whose process has N+1 or more threads over NUMA nodes. Then only one NUMA node will be used with the context switching of some threads.

- Good for NUMA
  - Each node has one process which has **N** or less threads.
  - Total size of used memory on each NUMA node is less than the memory size of the NUMA node.
- Bad for NUMA
  - A node has one process which has **N**+1 or more threads. Then context switching of some threads is necessary.
  - Total size of used memory on a NUMA node is greater than the memory size of the NUMA node. Then the memory of the NUMA node where the process is not running will be allocated and used.

Chapter4　Program Execution in NUMA Mode

This chapter explains how to choose the NUMA node for your program. Please note a NUMA node means a segment of partitioning mode.

4.1　VEOS View

When you execute a program without any NUMA options, then VEOS creates a process for the program on a NUMA node whose load is lower. When the process requests memory allocation, memory belonging a local NUMA node on which the process is running (local memory) is allocated first. If the local memory is full, memory belonging an opposite NUMA node (remote memory) is allocated. VEOS defines the policy as **MPOL_DEFAULT**.

You can use options to specify using NUMA node and memory policy to **ve_exec** command:

--cpunodebind

Specify the NUMA node to execute a VE program

--localmembind

Allocate memory only from the NUMA node on which a process is running. VEOS defines the policy as **MPOL_BIND**.

The **--membind** option is not supported to avoid the bad performance if the node specified by **--membind** is different from the node specified by **--cpunodebind**.

You can also use an environment variable, **VE_NUMA_OPT**, to specify the options for NUMA. The environment variable takes effect when you execute VE program through binfmt.

Examples:

- **ve_exec** option

```
$ /opt/nec/ve/bin/ve_exec --cpunodebind=0 ./a.out
```

- Environment variable and **ve_exec**

```
$ VE_NUMA_OPT="--localmembind" /opt/nec/ve/bin/ve_exec a.out
```

- Environment variable and binfmt

```
$ VE_NUMA_OPT="--cpunodebind=0 --localmembind" ./a.out
```

4.2 MPI Program View

4.2.1 Overview

When you execute an MPI program using NUMA mode, MPI communication performance differs between within the same NUMA node and over different NUMA nodes even within same VE. If you execute a program without explicitly specifying NUMA nodes, the correspondence between MPI processes and NUMA nodes is automatically determined (Figure 5). In this case, the execution may run well or may not in terms of program performance. You may want to explicitly specify the NUMA node for better and stable performance. And, in the hybrid execution with shared-memory parallelism and MPI and the number of processes each VE node is equal to two, those processes are executed with different NUMA nodes, even if the NUMA nodes are not explicitly specified (Figure 6).

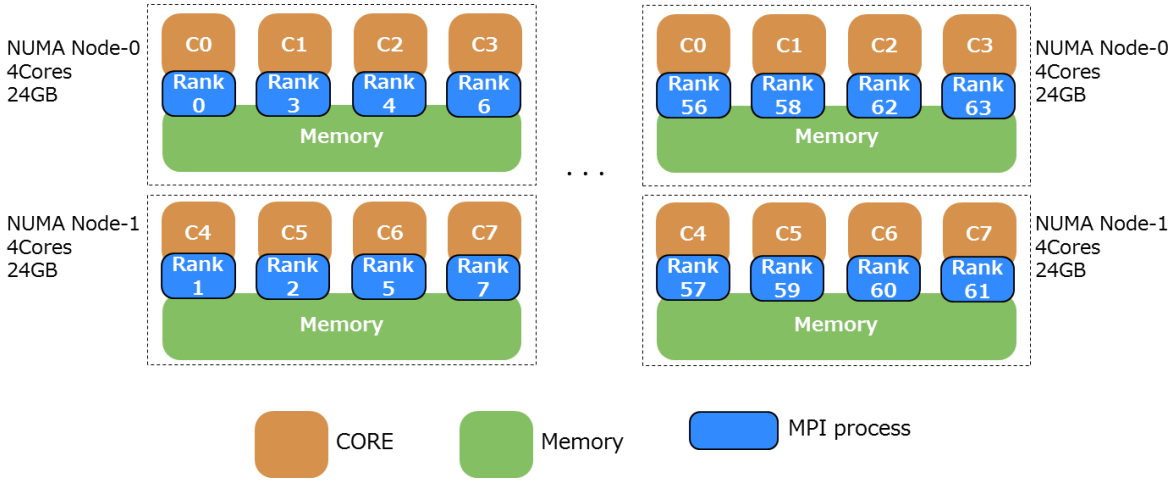

Figure 5 Example of execution without NUMA node number specified

(by default, NUMA node is automatically selected)

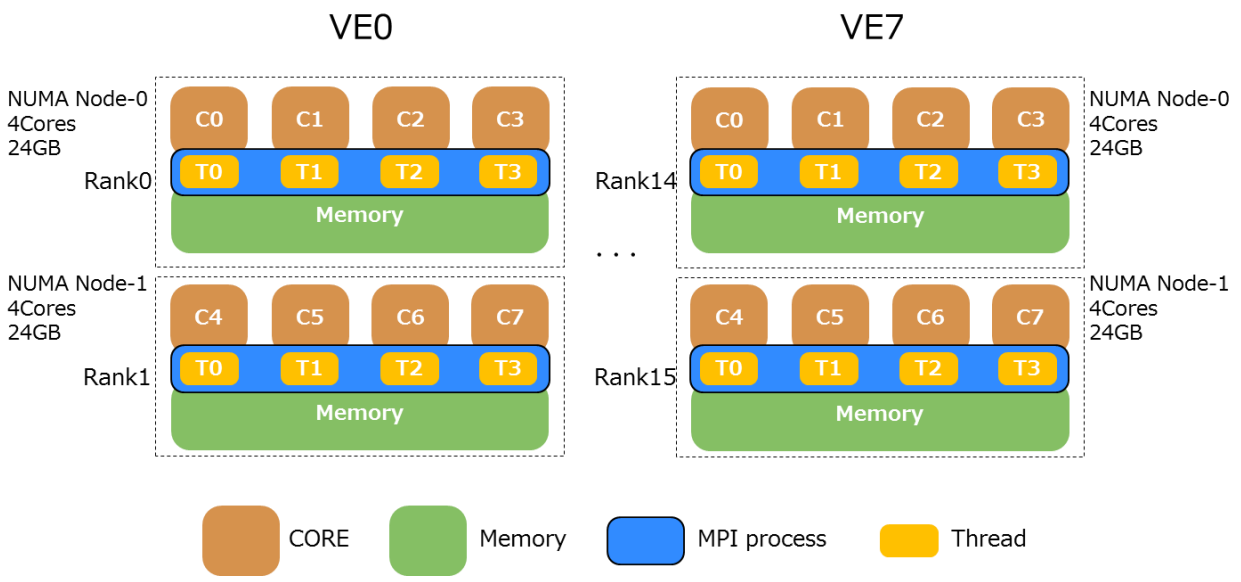

Figure 6 Example of execution without NUMA node number specified

(hybrid execution with shared-memory parallelism and MPI)

4.2.2 Example to Specify NUMA Node

The following is an example of execution of a.out with a total of 64 processes, assigning the first half of 8 processes per 1VE to NUMA node 0 and the second half of them per 1VE to NUMA node 1, using 1 node (VH) and 8 VEs (Figure 7).

```
$ mpirun -ve 0-7 –numa 0-1 -np 64 ./a.out
```

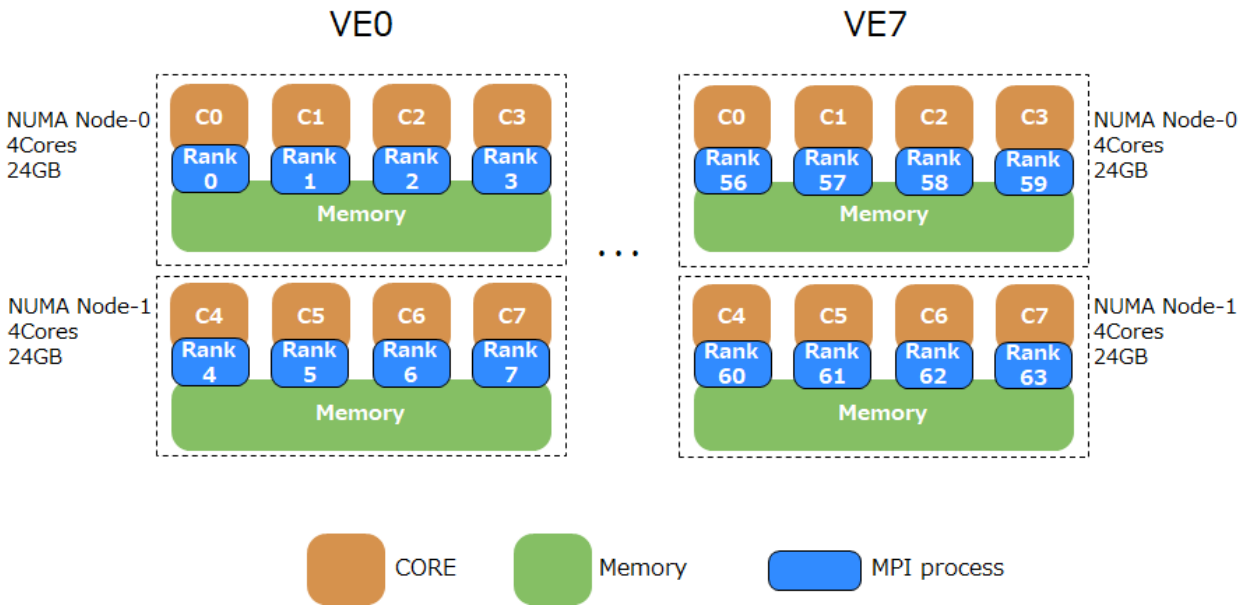

Figure 7 Example of ranks assigned to specified NUMA nodes

4.3 Multi-threaded Program View

4.3.1 OpenMP / Automatic Parallelization

If your OpenMP program or automatic-parallelized program are compiled by NEC compiler 2.1.0 or later, the default number of threads is the same as the number of CPU cores on the target VE Node in normal mode or the target NUMA node in NUMA mode. Therefore you don't need to take care of it.

If your OpenMP program or automatic-parallelized program are compiled by NEC compiler 2.0.8, the default number of threads is the same as the number of CPU cores on the target VE node regardless of the current mode. Please specify environment variable **OMP_NUM_THREADS**=4 when you execute the program in NUMA mode.

4.3.2 Pthread

The following program can create the appropriate number of threads on both NUMA and normal mode.

```
#define _GNU_SOURCE
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>

#define handle_error_en(en, msg) ¥
        do { errno = en; perror(msg); exit(EXIT_FAILURE); } while (0)

int
main(int argc, char *argv[])
{
        int s, num_threads;
        cpu_set_t cpuset;
        pthread_t thread;

        thread = pthread_self();

        /* Check the actual affinity mask assigned to the thread */

        s = pthread_getaffinity_np(thread, sizeof(cpu_set_t), &cpuset);
        if (s != 0)
                handle_error_en(s, "pthread_getaffinity_np");

        num_threads = CPU_COUNT(&cpuset);

        /* Creat the appropriate number of threads */
         ...;
}
```

4.4 APIs for NUMA

4.4.1 ve_get_numa_node

```
int ve_get_numa_node(unsigned *node);
```

ve_get_numa_node() gets the NUMA node number on which the caller process is running.

- The NUMA node number is stored to the integer pointed by “node”.
- 0 is returned on success. -1 is returned and errno is set on failure.
- This is implemented in libsysve.

Chapter5 NOTE

- VEOS
  - The **numactl** command is not supported.
  - The **ve_exec**, **taskset**, **prlimit**, **strace** and **time** commands support **VE_NUMA_OPT**.
    - If the commands are executed without **VE_NODE_NUMBER**, VE program will be start on the VE node#0. If VE node#0 is offline, the command will fail with an error message, “Node ‘0' is Offline”.
    - If a user doesn’t specify the NUMA node, VEOS checks the current load of each NUMA node to choose the NUMA node and chooses it. But this choosing is not always the best. Therefore user’s choosing is always recommended to get the best performance.
  - New thread/process creation
    - When a process invokes **clone()**/**fork()**/**vfork()**/**exec()**, new thread/process is created on the same NUMA node.
  - Migration
    - Process and memory migrations are not supported.
      - If the **taskset** command, **sched_setaffinity**() or **pthread_setaffinity_np**() is invoked with another node’s cores, it will be unsuccessful.
- NEC Network Queuing System V (NQSV)
  - NQSV is a job scheduler software for SX-Aurora TSUBASA. NQSV R1.09 supports automatic switching function of VE NUMA mode. By specifying VE NUMA mode when submitting a job, it is possible to automatically switch the partitioning mode of the VE that executes the job. For details, see "NQSV Usage Guide [Administration] 12.7 Automatic Switching of VE NUMA Mode". Refer to the following notes when using NQSV for job operations.
    - Using execution hosts with partitioning mode ON in R1.08 or earlier versions
      - The partitioning mode of all execution hosts bound to the same queue should be the same. If execution hosts with partitioning mode ON and OFF are mixed in a queue, it may result in unexpected behavior, such as job execution error.
    - VEs allocation
      - The allocation of VEs automatically performed by NQSV. Therefore, the user

don't designate environment variable VE_NODE_NUMBER and **ve_exec** -N.

Appendix A History

A.1 History table

June 2019	Rev. 1
September 2019	Rev. 2
October 2020	Rev. 3
December 2021	Rev. 4
March 2023	Rev. 5

A.2 Change Notes

Rev.2

- Added the description of “—numa” option of “mpirun” command to 4.2.2. Example to Specify NUMA Node

Rev.3

- Added the description regarding VE operation mode change by vecmd command without –N option to 2.1 Mode Change.

Rev.4

- Added the description of VE NUMA mode automatic switching function to Chapter 5 Note.

Rev.5

- This revision covers VEOS v3.0.2 or later
- Added descriptions for VE30