NEC

SX-Aurora TSUBASA

SX-Aurora TSUBASA
パーティショニングモードのための
VEOS NUMA モードガイド

輸出する際の注意事項

本製品（ソフトウェアを含む）は、外国為替および外国貿易法で規定される規制貨物（または役務）に該当することがあります。

その場合、日本国外へ輸出する場合には日本国政府の輸出許可が必要です。

なお、輸出許可申請手続きにあたり資料等が必要な場合には、お買い上げの販売店またはお近くの当社営業拠点にご相談ください。

は　し　が　き

『パーティショニングモードのための VEOS NUMA モードガイド』は、SX-Aurora TSUBASA に搭載されているベクトルエンジンのパーティショニングモードを使うための方に向けたドキュメントです。パーティショニングモードを活用するため、VEOS は NUMA モードを提供しており、その基本的な使用方法について説明します。

このガイドは、次のことを前提にしています。

- SX-Aurora TSUBASA を使用するための VEOS、および関連するソフトウェアのインストールが完了していることと
- ユーザがログイン、またはジョブスケジューラ(NEC Network Queuing System V：NQSV)を使用できるよう準備ができていること
- ユーザは Fortran コンパイラ(nfort)、C コンパイラ(ncc)、C++コンパイラ(nc++)、NEC MPI の知識があること

veos-2.1.0-1.el7.x86_64 以降と mmm-1.2.13-0.el7.x86_64 以降がインストールされているときに、VEOS の NUMA モードが利用可能です。

VEOS と MMM のバージョンは、以下の方法で確認できます。

```
$ 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-2.3.0-1.el7.x86_64 以降がインストールされているときに、NEC MPI の NUMA モード向けオプションが利用可能です。

NEC MPI のバージョンは、以下の方法で確認できます。

```
$ rpm -q nec-mpi-runtime
nec-mpi-runtime-2.3.0-1.el7.x86_64
```

備考

(1) Linux は Linus Torvalds氏の米国およびその他の国における登録商標あるいは商標です。

(2) その他、記載されている会社名、製品名は、各社の登録商標または商標です。

用語定義・略語

用語・略語	説　明
ベクトルエンジン (VE、Vector Engine)	SX-Aurora TSUBASAの中核であり、ベクトル演算を行う部分です。PCI Expressカードであり、x86サーバーに搭載して使用します。
ベクトルホスト (VH、Vector Host)	ベクトルエンジンを保持するサーバー、つまり、ホストコンピュータを指します。
VEOS	VEOSは、ベクトルエンジン上で動作するプログラムに対してOS機能を提供する、Linux/ベクトルホスト上で動作するソフトウェアです。
MPI	Message Passing Interfaceの略語です。主にノード間で並列コンピューティングを行うための標準規格です。同一ノード上のプロセス間の通信にMPIを使用することも可能です。OpenMPとの併用も可能です。
NUMA	Non-Uniform Memory Accessの略語です
パーティショニングモード	VEコアと、ラストレベルキャッシュ、メインメモリを2つのセグメントに分割して使うためのVEハードウェアのモードです。
NUMAモード	パーティショニングモードが有効なときに、VEを制御するためのVEOSのモードです。
ノーマルモード	パーティショニングモードが無効であることを意味しています。また、NUMAモードが無効なことを意味することもあります。

目 次

図目次

第1章　パーティショニングモード/NUMA モード

より良い性能を引き出すため、ベクトルエンジンはコア、ラストレベルキャッシュ(LLC)、及びメインメモリを 2 つのセグメントとして分割するパーティショニングモードをサポートします。パーティショニングモードの利点の一つは、LLC の衝突を回避することです。パーティショニングモードの欠点の一つは、一つのコアがメインメモリの半分の帯域しか利用できないことです。

VEOS は Linux ベースのソフトウェアです。Linux は NUMA をサポートしており、NUMA 用のシステムコールやコマンドがあります。VEOS でも、これらのシステムコールインタフェースやコマンドを活用するため、VEOS はパーティショニングモードの 2 つのセグメントを、二つの NUMA ノードとして扱います。パーティショニングモードに対応する VEOS のモードは、NUMA モードです。

ノーマルモードという用語は、パーティショニングモードが無効であることを意味します。NUMA モードが無効であることを意味する場合もあります。

図 1 は、ノーマルモードと、パーティショニングモードの違いを示しています。図中のコア数や、メモリサイズは例であり、異なる場合があります。

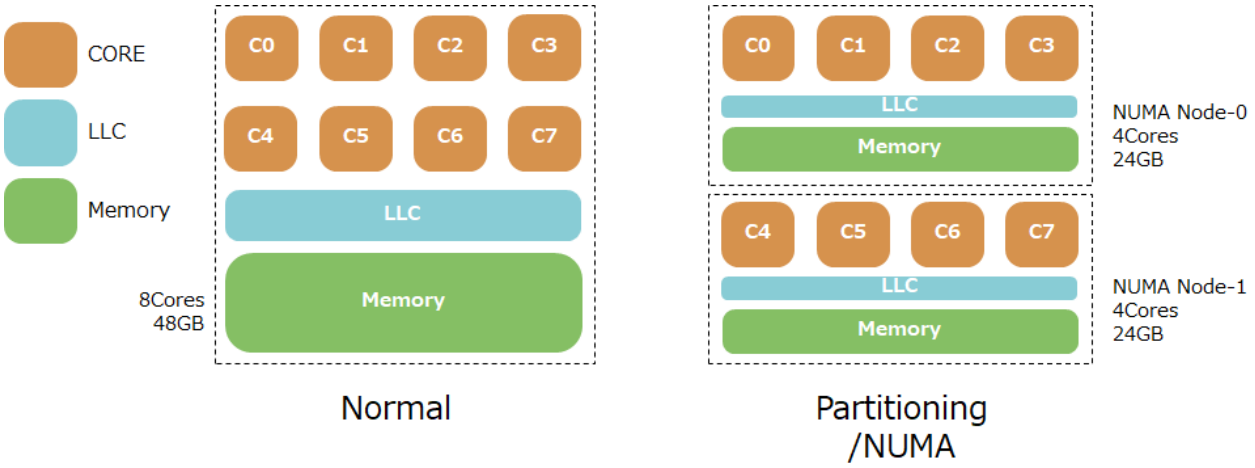

図 1 VE 動作モードの概要

ノーマルモードでは、VE カードのすべてのコアとメモリをプログラムが利用できるため、ほとんどの場合においてノーマルモードが適しています。そのため、デフォルトのモードはノーマルモードです。しかし、NUMA モードの方が速いプログラムもあります。そのため、モードは設定可能になっています。

VEOS の NUMA モードは、VEOS バージョン 2.1 からサポートしています。

第2章　モードの変更

この章は、モードの変更方法と確認方法を説明します。

補足　この章で「#」プロンプトで始まる行があります。この行に記載されているコマンドは、特権ユーザの権限で実行する必要があります。

2.1　モードの変更

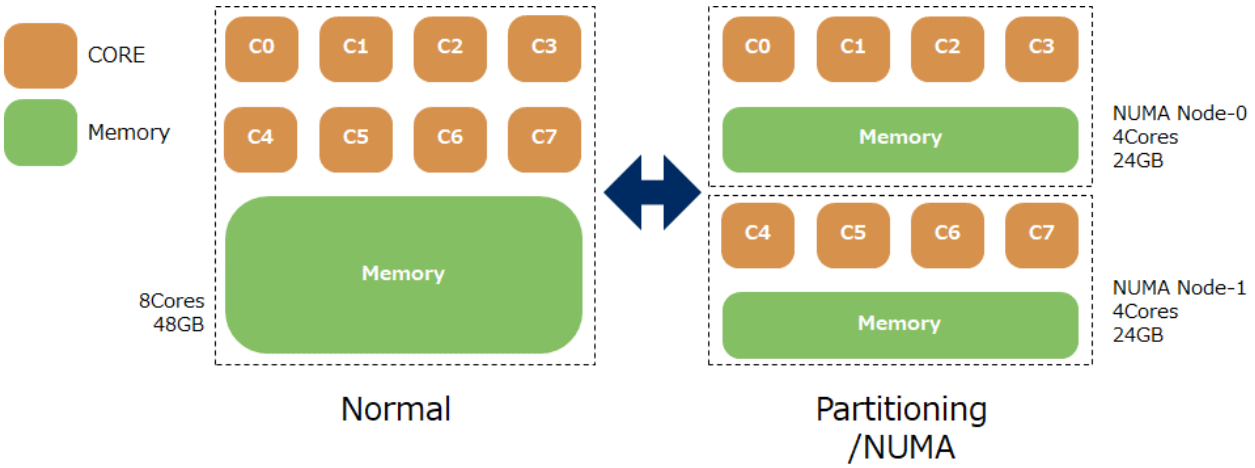

図 2 VE 動作モードの変更

VE 動作モードの変更は**/opt/nec/ve/bin/vecmd** コマンドを使います。パーティショニングモード(NUMA モード)に変更するためには、次のコマンドを実行します。<VE node number>の部分は、モードを変更したい VE の番号を指定してください。” -N <VE node number>” オプションを指定しない場合は、vecmd コマンドは全ての VE の設定を変更します。

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>
```

ノーマルモードに変更するためには、次のコマンドを実行します。

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>
```

VE 動作モードの設定は保存され、VH/Linux を再起動しても引き継がれます。

2.2 モードの確認

/opt/nec/ve/bin/venumainfo コマンドでモードを確認することができます。

ノーマルモードでは、**venumainfo** は「available: 1 nodes(0)」と表示します。加えて、このコマンドは、VE 全体を NUMA ノード 0 として表示します。

```
$ /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
```

NUMA モードでは、**venumainfo** コマンドは、「available: 2 nodes(0-1)」と表示します。加えて、このコマンドは、NUMA ノード 0 と NUMA ノード 1 の情報を表示します。

```
$ /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
```

これらの実行結果におけるコアの数とメモリのサイズは例ですので、異なる結果が表示される場合があります。

第3章　プログラムに適するモード

もしプログラムが 1 プロセスで構成されており、プログラムが生成するスレッドが、VE に実装されているコアの数の半分以下の場合、ノーマルモードが適しています。このようなプログラムにとっては、NUMA モード(パーティショニングモード)よりノーマルモードの方が利用可能なメモリ帯域が大きいためです。一方で、プログラムが複数のプロセスから構成されている場合や、多くのスレッドを生成する場合は、適しているモードを明確に言うことは難しいです。この章では、プログラムにとって NUMA モードが適している場合と適さない場合を説明します。

3.1　マルチプロセスプログラム

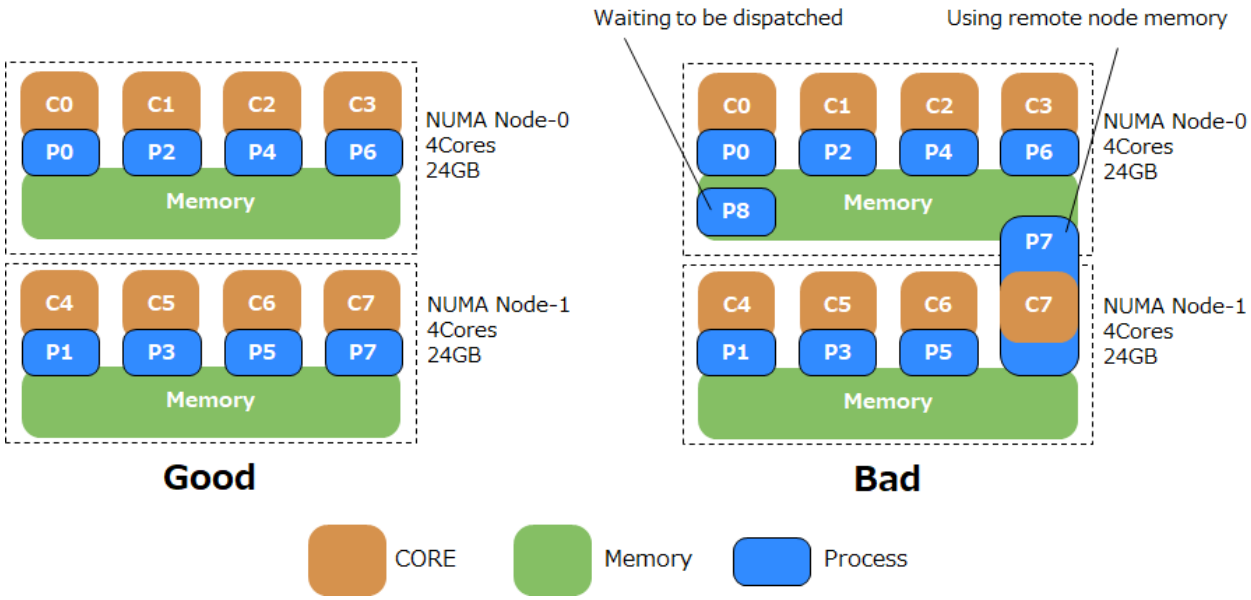

図 3 マルチプロセスプログラムの実行例

N を NUMA ノードに実装されているコアの数として説明します。

- NUMA モードが適する場合 (Good)
  - 各 NUMA ノードで **N** 以下のプロセスを実行する場合。
  - 各 NUMA ノード内のプロセスが利用するメモリの合計が、NUMA ノードのメモリサイズ以下の場合。
- NUMA モードが適さない場合 (Bad)
  - ある NUMA ノードで、**N**+1 以上のプロセスを実行する場合。この場合は、プロセス間のコンテキストスイッチが発生します。
  - ある NUMA ノード内のプロセスが利用するメモリの合計が、NUMA ノードのメモリサイズを超える場合。この場合、プロセスが実行されている NUMA ノードとは異なる NUMA ノードのメモリが確保されて利用されます。

3.2 マルチスレッドプログラム

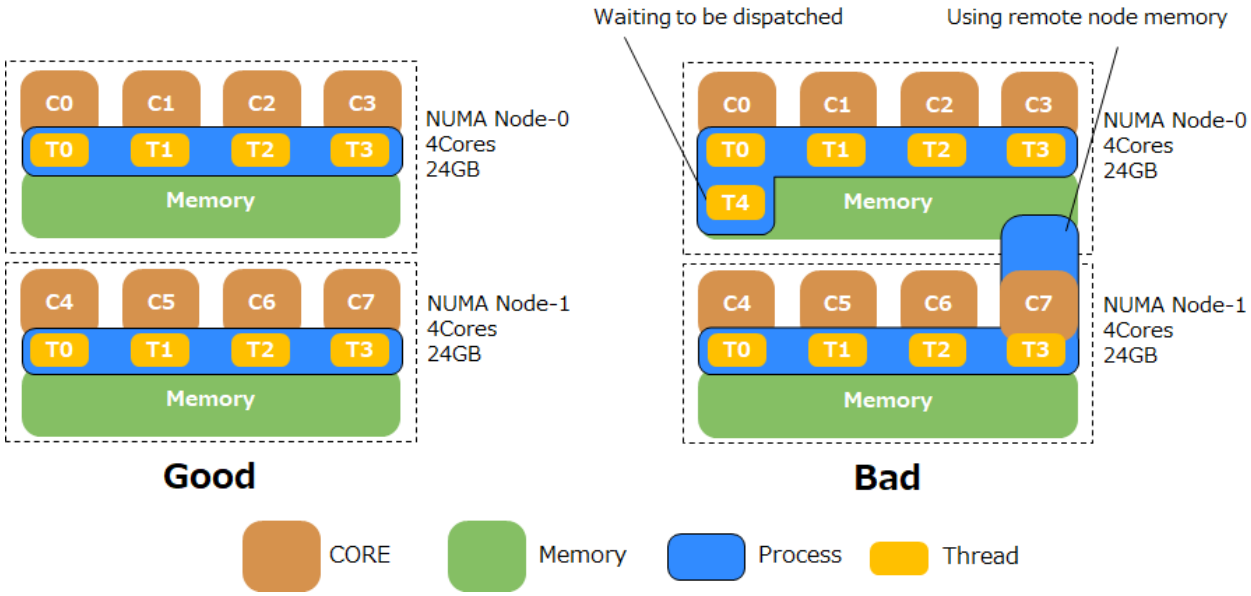

図 4 マルチスレッドプログラムの実行例

N を NUMA ノードに実装されているコアの数として説明します。なお、**N**+1 以上のスレッドを生成しても、NUMA ノードを超えてプログラムを実行することはできません。一方の NUMA ノードで **N**+1 のスレッドが実行され、スレッド間のコンテキストスイッチが発生します。

- NUMA モードが適する場合 (Good)
  - 各 NUMA ノードで **N** 以下のスレッドを生成するプロセスを一つ実行する場合。
  - 各 NUMA ノード内のプロセスが利用するメモリの合計が、NUMA ノードのメモリサイズ以下の場合。
- NUMA モードが適さない場合 (Bad)
  - ある NUMA ノードで **N**+1 以上のスレッドを生成するプロセスを実行する場合。この場合は、スレッド間のコンテキストスイッチが発生します。
  - ある NUMA ノード内のプロセスが利用するメモリの合計が、NUMA ノードのメモリサイズを超える場合。この場合、プロセスが実行されている NUMA ノードとは異なる NUMA ノードのメモリが確保されて利用されます。

第4章　NUMA モードでのプログラムの実行

この章では、プログラムを実行する NUMA ノードを指定する方法を説明します。この NUMA ノードは、パーティショニングモードのセグメントに相当します。

4.1　VEOS観点

NUMA オプションなしでプログラムを実行した場合、VEOS は、負荷の低い NUMA ノードにおいて、プログラムを実行するプロセスを生成します。プロセスがメモリ確保を要求した場合、プロセスが実行されている NUMA ノードに属するメモリ(ローカルメモリ)が初めに割り当てられます。ローカルメモリが一杯になった場合は、もう一方の NUMA ノードに属するメモリ(リモートメモリ)が割り当てられます。VEOS はこのポリシーを **MPOL_DEFAULT** と定義します。

NUMA ノードとメモリポリシーを指定するために、**ve_exec** コマンドにオプションを指定することができます。

--cpunodebind

VE プログラムを実行する NUMA ノードを指定します。

--localmembind

プロセスが実行されている NUMA ノードからのみメモリを確保するようにします。VEOS は、このポリシーを MPOL_BIND と定義します。

--membind オプションは、**--cpunodebind** で指定された NUMA ノードと異なる NUMA ノードを指定された場合の性能低下を避けるためにサポートしていません。

VE_NUMA_OPT 環境変数を使ってこれらの NUMA オプションを指定することができます。環境変数は、binfmt を使って VE プログラムを実行したときも効果があります。

例：

- **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プログラム観点

4.2.1 概要

NUMA モードを利用して MPI プログラムを実行する場合、同一 VE 内の MPI プロセス間通信であっても、同一 NUMA ノード内と異なる NUMA ノードの間では MPI 通信性能が異なります。NUMA ノードを明示的に指定せずにプログラムを実行した場合、既定値の動作として各 MPI プロセスと NUMA ノードの対応は自動的に決定されますが（図 5）、その実行はプログラム実行性能の観点で必ずしも適切とは限りません。安定した高い性能を得るためには、各 MPI プロセスを実行する NUMA ノードを適切に指定して実行してください。なお、共有並列処理と MPI を併用したハイブリッド実行で VE あたりのプロセス数が 2 の場合は、NUMA ノードを明示的に指定しなくても、それらのプロセスは異なる NUMA ノード上で動作します（図 6）。

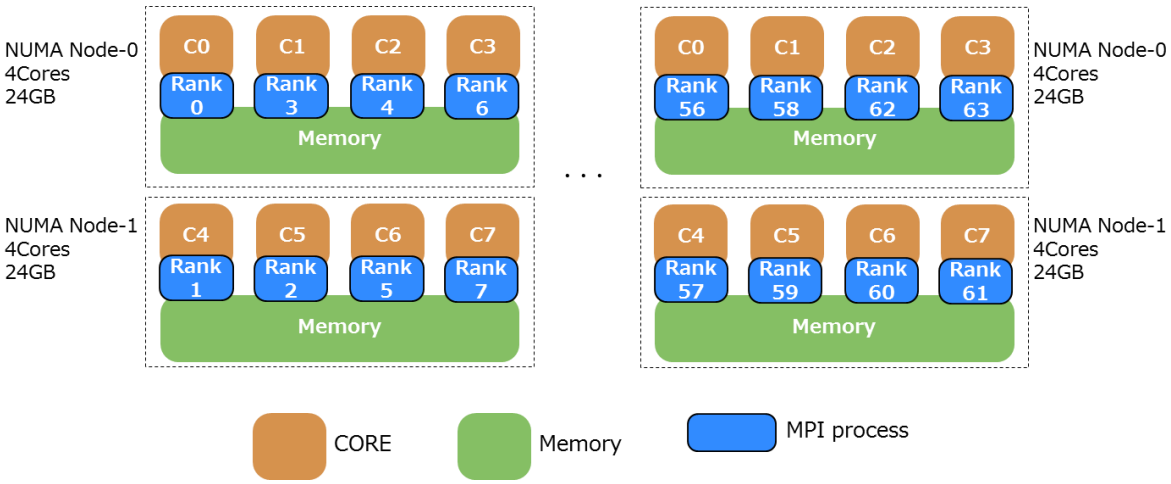

図 5 NUMA ノード未指定の例（既定値の動作：NUMA ノード自動選択）

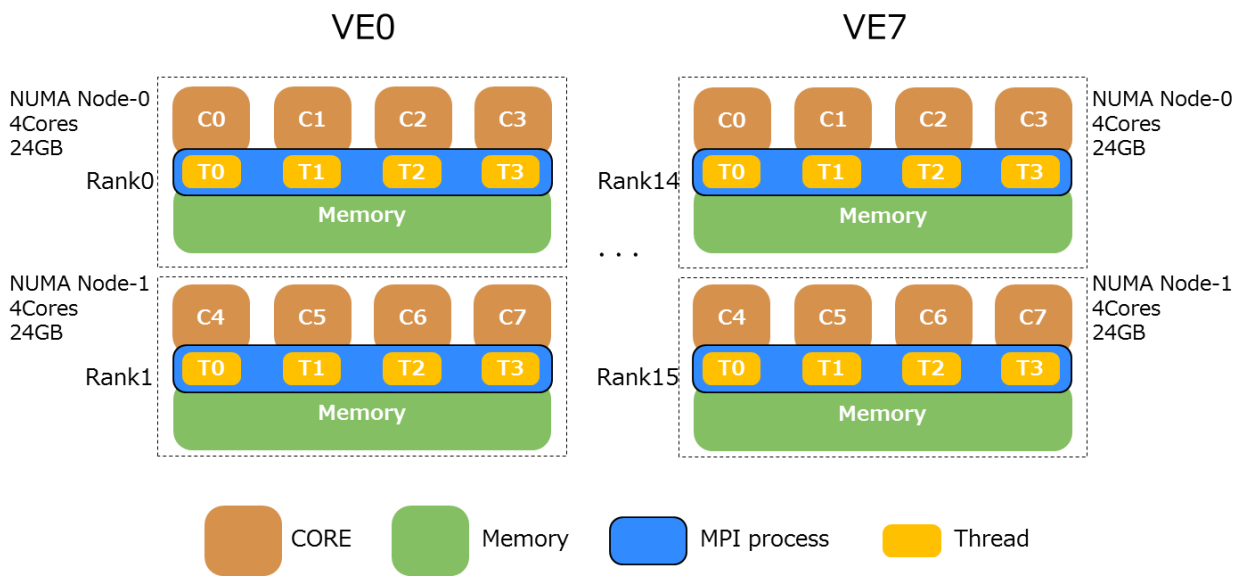

図 6 NUMA ノード未指定の例（共有並列+MPI ハイブリッド実行：両 NUMA ノードに分散）

4.2.2 NUMA モードを指定する例

8VE を利用し、1VE あたり前半 4 プロセス（ランクが若いプロセス）を NUMA ノード 0 へ、後半 4 プロセスを NUMA ノード 1 へ割り当て、合計 64 プロセスで a.out を実行する場合（図 7）、以下のコマンドを実行します。

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

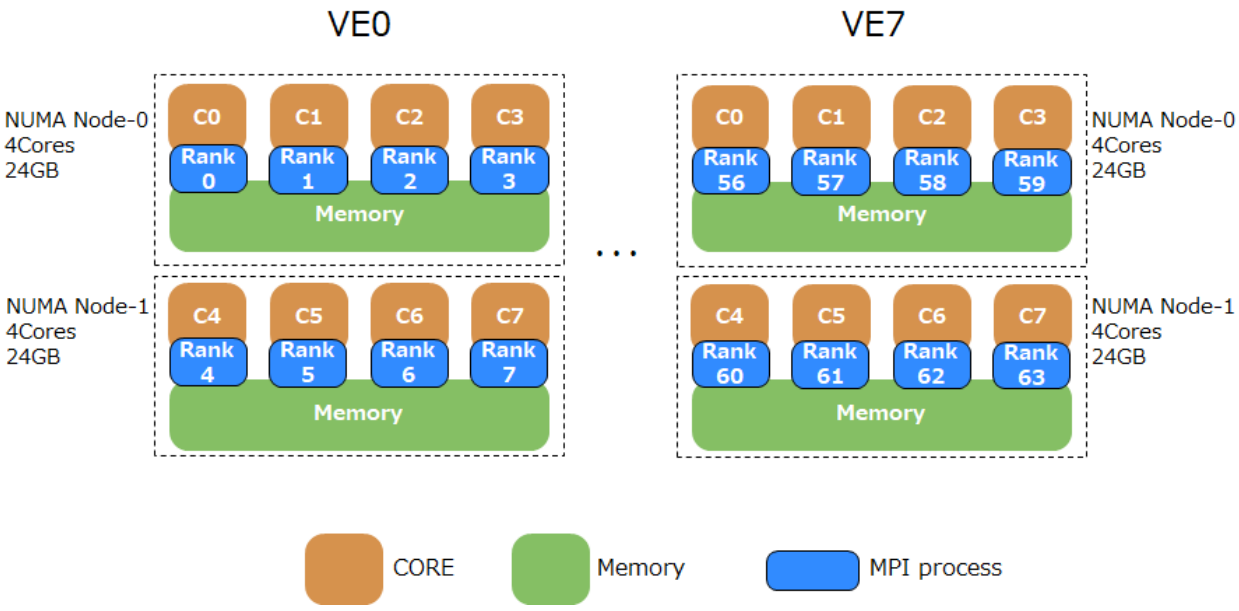

図 7 NUMA ノード指定とランク割り当ての例

4.3 マルチスレッドプログラム観点

4.3.1 OpenMP / 自動並列化

2.1.0 以降の NEC コンパイラによってコンパイルされた OpenMP プログラムまたは自動並列化プログラムの場合、デフォルトのスレッド数は、ノーマルモードでは VE に搭載されているコアの数となり、NUMA モードでは NUMA ノードに属するコアの数となります。そのため、スレッドの数をユーザが考慮する必要はありません。

2.0.8 の NEC コンパイラによってコンパイルされた OpenMP プログラムまたは自動並列化プログラムの場合、現在のモードに関わらず、デフォルトのスレッド数は、VE に搭載されているコアの数となります。NUMA モードでプログラムを実行する場合は、環境変数 **OMP_NUM_THREADS**=4 を設定してください。なお、4 は、NUMA ノードに属するコアの数に合わせて変更してください。

4.3.2 Pthread

次のプログラムは、NUMA モードでも、ノーマルモードでも適切な数のスレッドを生成します。

```
#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 NUMAのためのAPI

4.4.1 ve_get_numa_node

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

ve_get_numa_node() は呼び出し元のプロセスが実行されている NUMA ノードの番号を取得します。

- NUMA ノード番号は、「node」で指定された整数に格納されます。
- 成功時に 0 が返ります。失敗時に errno が設定され、-1 が返ります。
- libsysve に実装されています。

第5章　注意事項

- VEOS
  - **numactl** コマンドはサポートしていません。
  - **ve_exec** と **taskset**、**prlimit**、**strace**、**time** コマンドは環境変数 **VE_NUMA_OPT** をサポートしています。
    - 環境変数 **VE_NODE_NUMBER** なしで実行された場合は、VE プログラムは VE ノード#0 で実行されます。VE ノード#0 がオフラインの場合、“Node '0' is Offline”というエラーメッセージが出力されて、失敗します。
    - ユーザが NUMA ノードを指定しない場合、VEOS は各 NUMA ノードの現在の負荷を確認し、NUMA ノードを選択します。しかし、選択が最適とは限りません。そのため、最高性能を得るため、ユーザが NUMA ノードを選択することを推奨します。
  - 新しいスレッドやプロセスの生成
    - プロセスが **clone()**/**fork()**/**vfork()**/**exec()**を呼び出したとき、新しいスレッドやプロセスは同一の NUMA ノードで生成されます。
  - マイグレーション
    - プロセスとメモリの NUMA ノード間のマイグレーションはサポートしていません。
      - **taskset** コマンドや **sched_setaffinity**()、**pthread_setaffinity_np**() によって別の NUMA ノードのコアが指定された場合は、失敗します。

- NEC Network Queuing System V (NQSV)
  - NQSV は SX-Aurora TSUBASA 向けのジョブスケジューラソフトウェアです。NQSV R1.09 で VE NUMA モードの自動切り替え機能をサポートしました。ジョブ投入時に VE NUMA モードを指定することで、ジョブを実行する VE のパーティショニングモードを自動的に切り替えることが可能です。詳細は、「NQSV 利用の手引[管理編] 12.7 VE NUMA モードの自動切り替え」を参照してください。NQSV でジョブ運用を行う場合は下記の注意事項をご確認ください。
    - R1.08 以前の NQSV でパーティショニングモード ON の実行ホストを利用する場合
      - キューにバインドする実行ホストの全ての VE のパーティショニングモードは同じにしてください。パーティショニングモード ON と OFF の VE が混在する場合、ジョブ実行がエラーになるなど、期待した動作にならない可能性があります。
    - NQSV の VE の割り当てについて
      - VE の割り当ては NQSV が行います。ユーザは、環境変数 **VE_NODE_NUMBER** と ve_exec -N を指定しないでください。

付録 A　発行履歴

A.1　発行履歴一覧表

2019 年	6 月	初版
2019 年	9 月	第 2 版
2020 年	10 月	第 3 版
2021 年	12 月	第 4 版
2023 年	3 月	第 5 版

A.2　追加・変更点詳細

2 版
- 4.2.2. NUMAモードを指定する例 にmpirunのオプション-numaの説明を追加

3 版
- 2.1 モードの変更に vecmd コマンドに–N オプションを指定しない時のVE動作モード変更の説明を追加

4 版
- 第5章 注意事項に VE NUMA自動切り替え機能の説明を追加

5 版
この版はVEOS v3.0.2以降に対応します。
- VE30 向けの説明を追加

SX-Aurora TSUBASA

パーティショニングモードのための VEOS NUMAモードガイド

2023年 3月 第5版

日本電気株式会社

東京都港区芝五丁目７番１号

TEL(03)3454-1111（大代表）