\Orchestrating a brighter world NEC

VEOS概要設計

Revision 4.1

NEC

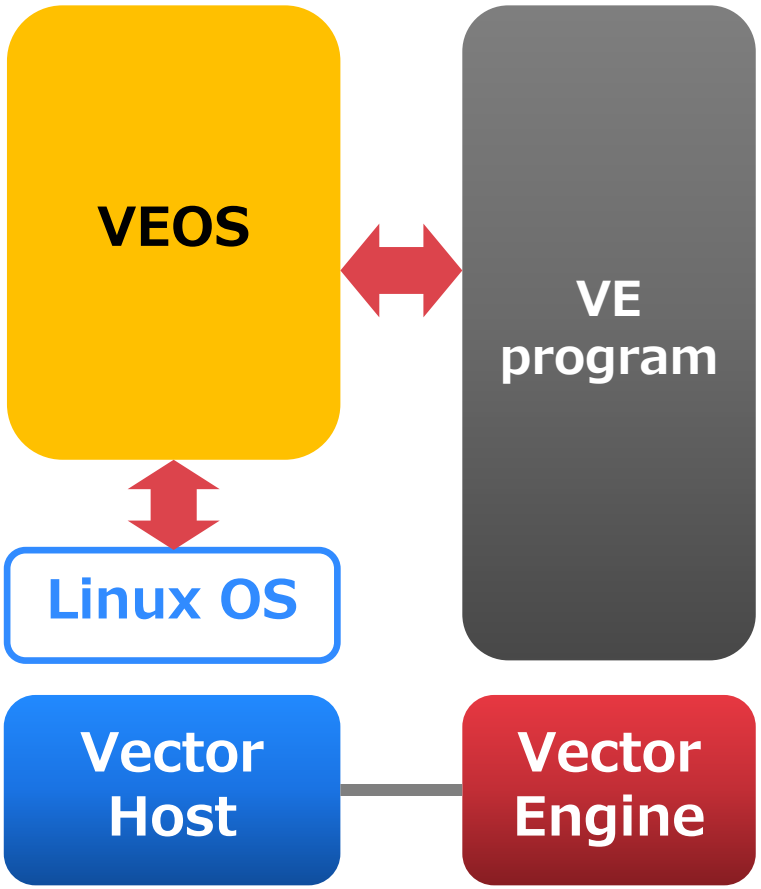
目次

このドキュメントについて

- このドキュメントではVEOSの概要設計について説明されています
- このドキュメントは VEOS v3.0.2 以降に対応します

VEOSとは

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



VEOSの機能:

- VEプログラムのロード
- システムコール処理
- プロセス管理
- メモリ管理
- シグナル処理
- OSコマンドのサポート
  - gdb, ps, free, top, sar etc.

Utilizing Linux capability

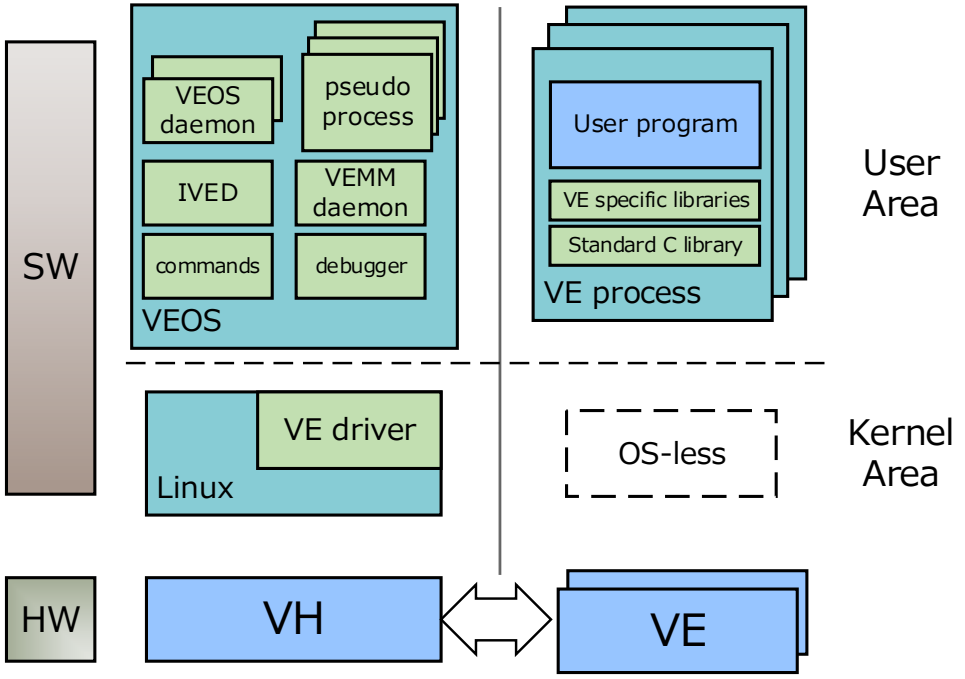
VEOSは２つの見方があります
VEプログラムから：Operating System
Linuxから: VE programの代理人

VEOSはLinux/x86上で動作するため、VE上にはOSジッタがありません

VEOSの基本的な機能

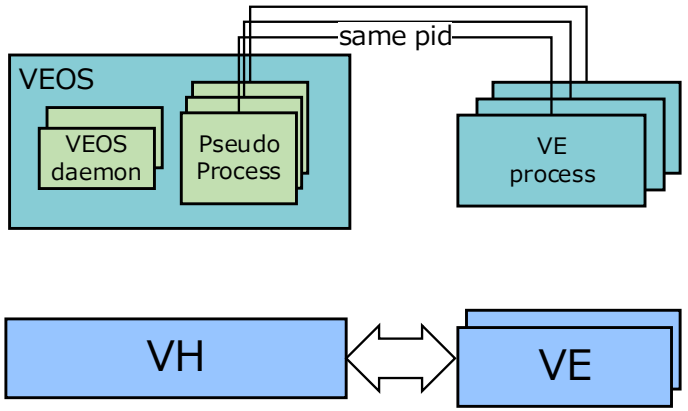
コンポーネント

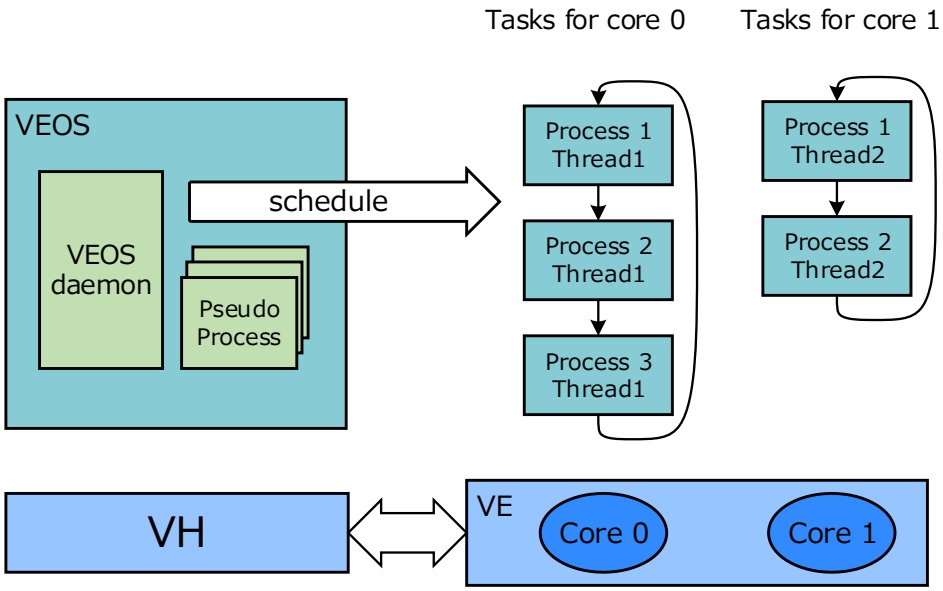
- VEOS
  - VEOSデーモン
    - １つのVEを管理するデーモン
    - プロセス管理やメモリ管理などを行います
  - 代理プロセス (ve_execコマンド)
    - １つのVEプロセスを管理するプロセス
    - VEプロセスへプログラムをロードし、VEプロセスからのシステムコール要求を処理します
  - IVED
    - VE間のリソース管理を行うデーモン
  - VEMMデーモン
    - InfiniBandドライバと通信を行うデーモン
  - コマンド
    - “ps”や”free”などの移植されたコマンド
  - デバッガ
    - 移植されたデバッガ(gdb)
- VEドライバ
  - VEOSデーモンと代理プロセスへのリソースアクセスを提供するドライバ
- VEプロセス
  - ユーザプログラムを実行するプロセス
  - 下記ライブラリがリンクされる可能性があります
    - 標準Cライブラリ
    - VE固有のライブラリ



プロセス管理

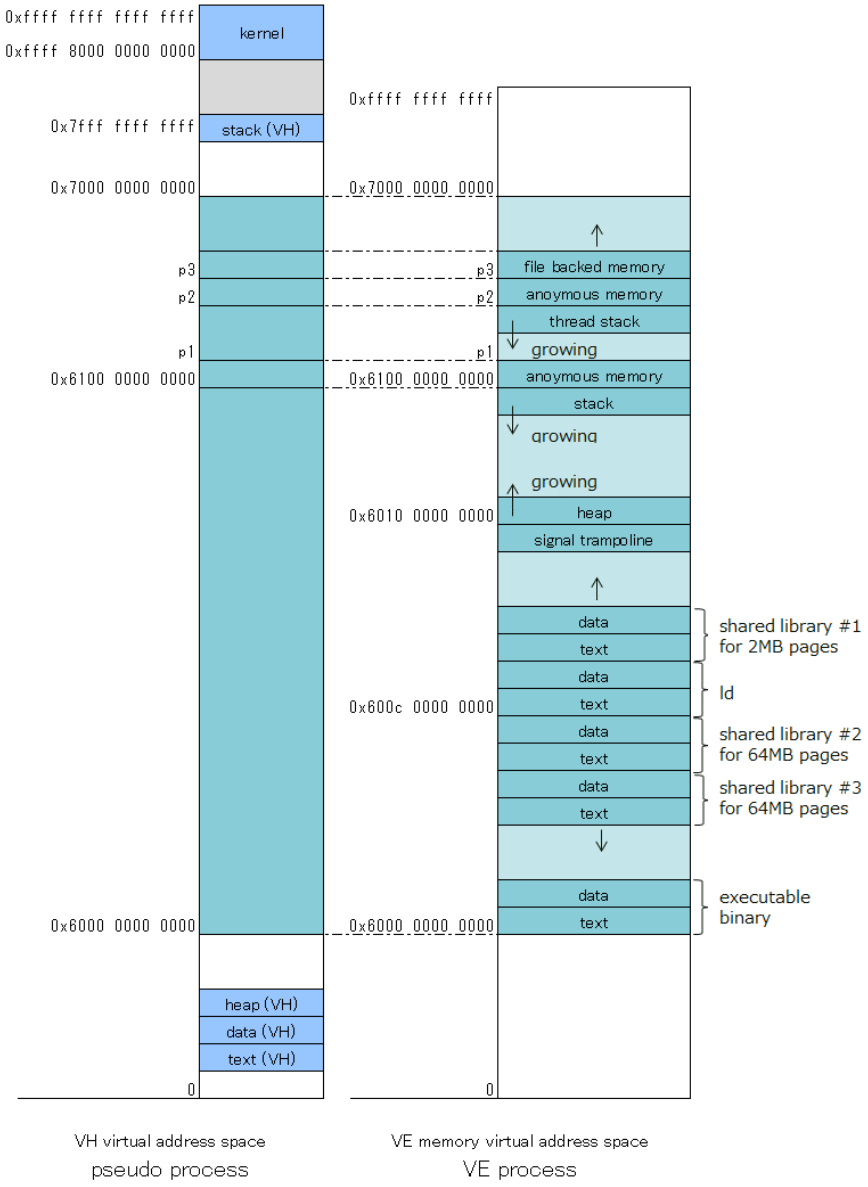
- マルチプロセスとマルチスレッドがサポートされています
- VEプロセスの識別
  - VEプロセスは対応する代理プロセスのPIDにより識別されます
- プロセスの状態
  - VEOSはLinuxから独立してVEプロセスの状態を管理します
    - 実行中、待機中など
- スケジューリング
  - VEOSは、 VEプロセスのスレッドが開始されると、スレッドを実行するためのVEコアを選択します
  - VEOSは、各VEコアに対し、ラウンドロビンスケジュールを用いてVEプロセスのスレッドをスケジューリングします
  - VEコアに実行すべきスレッドが一つもない場合、VEコア間でスレッドが移行することがあります
- 親プロセスと子プロセスの関係
  - VEOSはリソース制限やCPUアフィニティなどを継承するため、VEプロセスの関係を管理します
    - VEOSは１つのVE内の関係を維持します





メモリ管理

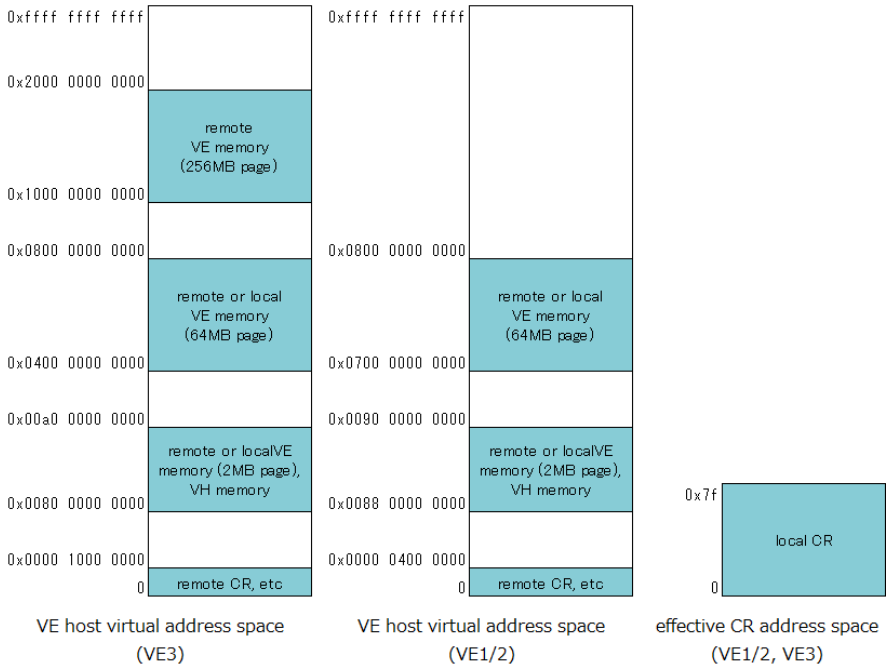
- 仮想アドレス管理
  - 実行可能バイナリのテキストとデータや、ヒープ、スタックは静的に割り当てられたアドレスに置かれます
  - 匿名メモリなどその他の領域は動的に割り当てられたアドレスに置かれます
  - VEメモリ仮想アドレス空間は、代理プロセスの仮想アドレス空間のサブセットです
  - 匿名メモリとファイルバックメモリがサポートされています
  - ファイルバックメモリのデータは、mmap(), munmap(), msync() and exit()といったシステムコールによりVEとVHの間で転送されます
- 物理VEメモリ管理
  - 実行可能バイナリ・共有ライブラリがロードされる際、または匿名・ファイルバックメモリが要求される際に、VEOSが物理メモリを割り当てます
  - HWがプリサイス例外をサポートしていないため、デマンドページングやコピーオンライトはサポートされていません
  - 共有マップが要求される場合、匿名メモリとファイルバックメモリはVEプロセス間の物理メモリを共有します
  - 読み取り専用保護によるプライベートマッピングが要求される場合、匿名メモリとファイルバックメモリは、VEプロセス間で物理メモリを共有する可能性があります
  - 実行可能バイナリと共有ライブラリは読み取り専用の保護がついたプライベートなファイルバックメモリです。したがって、VE間で物理メモリを共有します
  - ヒープとスタックの拡張がサポートされています
- ページサイズ
  - 64 MB / 2MB
  - 実行可能ファイルのテキストとデータや、ヒープ、スタックのページサイズは、実行バイナリのアライメントと等しくなります
  - 共有ライブラリのテキストとデータのページサイズは、共有ライブラリのアライメントと等しくなります
  - 匿名メモリ又はファイルバックメモリのデフォルトページサイズは、実行可能バイナリのアライメントと等しくなります。VEプログラムは、要求時にページサイズを指定することができます



ユーザモードDMAと通信レジスタ

- ユーザモードDMA
  - VEプロセスでは、ユーザモードDMAを使用し、以下に記載したメモリ間で、データを転送することができます
  1. VEプロセスに割り当てられたVEメモリと、他のプロセスに割り当てられたVEメモリ
  2. VHメモリと、VEプロセスに割り当てられたVEメモリ
  - VEプロセスは２つのDMAディスクリプタテーブルを使用することができます
  - ユーザーモードDMAはVEプロセスの1個以上のスレッドがVEコア上で実行されている際に実行されます
  - ユーザーモードDMAはVEプロセスの全スレッドがVEコアでされていない際には停止します
  - DMAのターゲットアドレスやソースアドレスはVEホスト仮想アドレスという名の特殊なアドレスにより特定されます
  - VEOSはVEホスト仮想アドレスからVEメモリへのマッピングを設定します

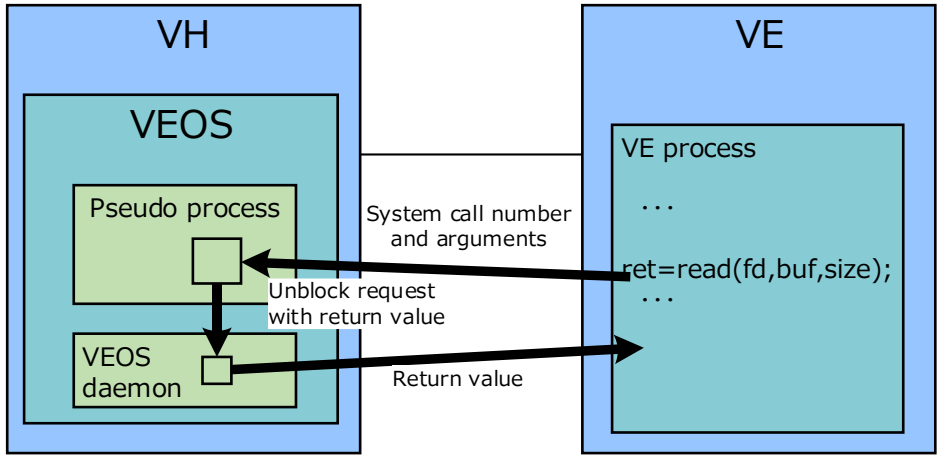
- 通信レジスタ (CR)
  - CRは64ビットのレジスタで、プロセスのスレッド間またはMPIプロセス間におけるオペレーションの同期もしくは排他制御のために使用されます
  - VEプロセスはローカルVEおよびリモートVEのCRにアクセスできます
    - ローカルVEのCRにアクセス
      - CRアクセス命令を使用します
      - CRは実効CRアドレスという名の特殊なアドレスを使用して指定します
    - リモートVEのCRにアクセス
      - ホストメモリアクセス命令を使用します
      - CRは、ユーザモードDMAでも使用されるVEホスト仮想アドレスを使用して指定します
  - VEOSは、実効CRアドレス又はVEホスト仮想アドレスから実際のCRへのマッピングを設定します



注意：User mode DMAとCRは、MPIライブラリや他のシステムライブラリで使用されています。
User mode DMAを使用するための低レベルAPIが提供されています。
CRはユーザープログラムから直接使用するためのものではありません。

システムコール

- VEプロセスはシステムコールをVHにオフロードします
  - 代理プロセスがシステムコールの要求を処理します
  - 代理プロセスは必要に応じてVEOSデーモンやLinuxを要求します
  - システムコールは代理プロセスの権限で処理されます

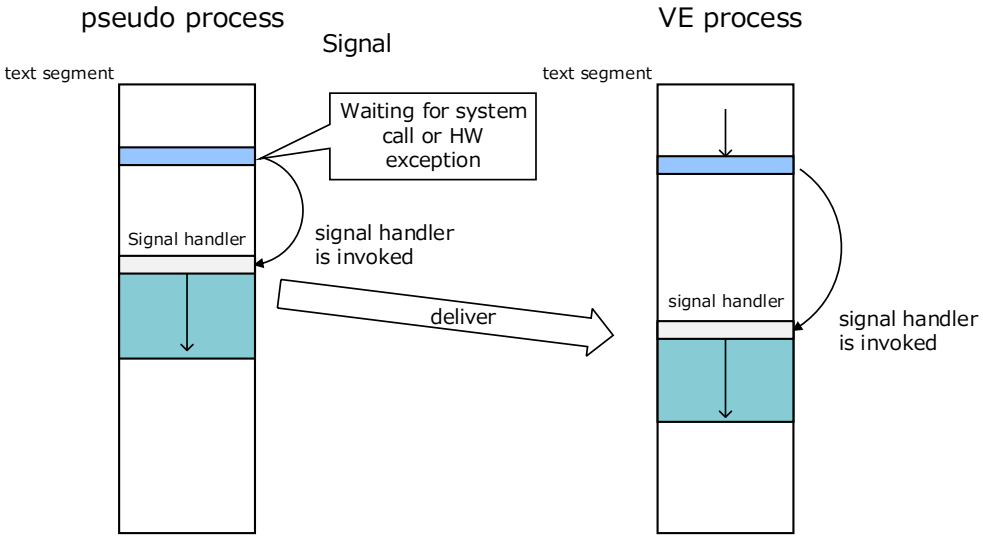


- システムコールの順序
  (単純なケース)

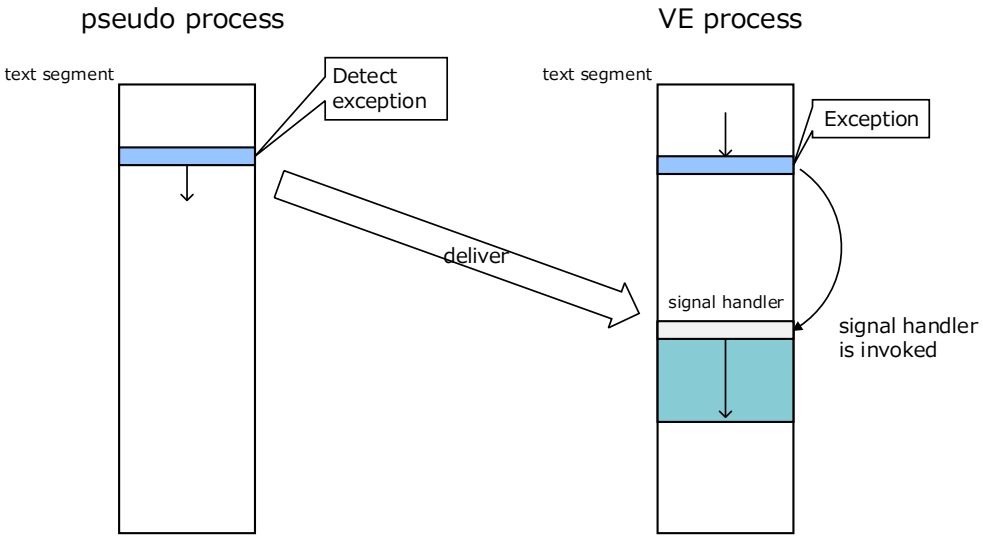
1. VEプロセスはシステムコール番号と引数をVHメモリに格納します
2. VEプロセスがVEコアを停止し、割り込みを発生させます
3. 代理プロセスがシステムコールを処理します
4. 代理プロセスは、戻り値とともに、ブロック解除要求をVEOSデーモンへ送ります
5. VEOSデーモンはVEコアのレジスタに戻り値を格納しVEコアを開始します

シグナル

- 別のプロセスにより送られるシグナル
  - 代理プロセスのシグナルハンドラはシグナル要求をVEOSに送ります
  - 次にVEプロセスがVEコアで実行される際、VEOSはシグナルをVEプロセスに送ります
  - VEプロセスがシグナルにより終了した場合、代理プロセスはSIGKILLにより強制終了されます



- HW例外によるシグナル
  - 代理プロセスはHW例外を検出しシグナル要求をVEOSに送ります
  - 次にVEOSがVEコア上で実行される際、VEOSはシグナルをVEプロセスに送ります
  - 回復不能なHW例外に対してVEプロセスがシグナルハンドルを登録すると、シグナルハンドラの実行後にVEプロセスが終了します



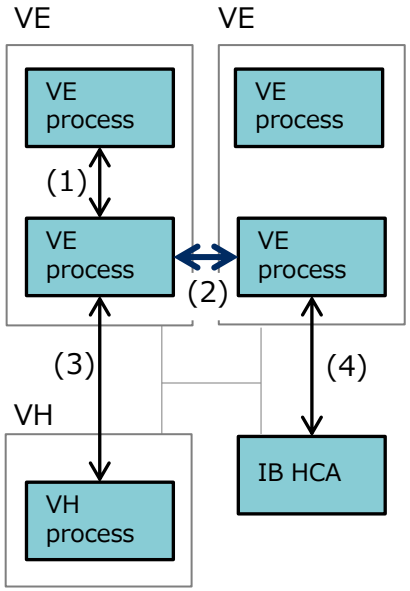
機能一覧(主要機能)

カテゴリ	機能	デザイン	概要
プログラムローダ	ファイルフォーマット	ELF, DWARF	プログラムローダがELFファイルを認識し、テキストとデータをVEにロードします
	ダイナミックリンク	サポートされています	共有ライブラリのダイナミックリンクはサポートされています
	ダイナミックロード	サポートされています	共有ライブラリのダイナミックロードはサポートされています(dlopen(), dlsym(), など)
	セグメントのアラインメント	2MB / 64MB	セグメントのアライメントはリンク時に決定されます
プロセス管理	マルチプロセス	Fork-execモデル	VEプログラムはfork()システムコールを使用してプロセスを作成することができます。VEプログラムはexecve()システムコールを使用して新しいVEプログラムを実行することができます
	マルチスレッド	POSIXスレッド	VEプログラムはPOSIX APIを使用してスレッドを作成することができます
	スケジューリング	各VEコア上のラウンドロビン	スレッドは優先度に関係なく特定の順序で実行されます。ブロックされた状態のスレッドはスキップされます。
	プリエンプション	サポートされています	タイムスライスを使い果たした場合、コンテキストスイッチが発生し、次のスレッドがVEコアで実行を開始します
	タイムスライス	1秒	プロセスのスレッドが実行される時間
	タイマーインターバル	100ミリ秒	VH側で実行される、スケジューラのタイムハンドラーの呼び出し間隔
メモリ管理	仮想アドレス空間	サポートされています	プロセスには独自の仮想アドレス空間があります
	メモリの割り当て	ダイナミック	VEプログラムがロードされるとメモリが割り当てられます。ヒープとスタックの拡張がサポートされています。 匿名またはファイルバックメモリの割り当てもサポートされています
	ファイルバックメモリ	サポートされています	ファイルのデータは、システムコールによりVEとVH間で転送されます
	デマンドページング	サポートされていません	仮想メモリが割り当てられると、物理メモリが割り当てられます
	コピーオンライト	サポートされていません	プライベート読み込み・書き込みエリアには、固有の物理メモリがあります
	物理メモリ共有	サポートされています	匿名メモリとファイルバックメモリは特定の条件下で物理メモリを共有します
	ページサイズ	2MB / 64MB	実行バイナリのセグメントページサイズと共有ライブラリのページサイズはそれらのアライメントと同じです。VEプログラムは、匿名又はファイルバックメモリのページサイズを指定することができます
入力と出力	ファイルシステム、ネットワークなど	サポートされています	Linuxにオフロードされます

機能一覧(プロセス間通信)

機能	(1) VE内部	(2) VE-VE	(3) VE-VH	(4) VE-IB	関数名(例)
System V shared memory	✓	-	-	-	shmget
POSIX shared memory	✓	-	-	-	shm_open
Spin lock	✓	-	-	-	pthread_spin_init
Mutex	✓(*1)	-	-	-	pthread_mutex_init
Read write lock	✓	-	-	-	pthread_rwlock_init
Condition variable	✓	-	-	-	pthread_cond_init
Barrier	✓	-	-	-	pthread_barrier_init
Unnamed semaphore	✓	-	-	-	sem_init
System V semaphore	✓	✓	✓	-	semget
Named semaphore	✓	✓	✓	-	sem_open
System V message queue	✓	✓	✓	-	msgget
POSIX message queue	✓	✓	✓	-	mq_open
UNIX socket	✓	✓	✓	-	socket
Pipe, Fifo	✓	✓	✓	-	pipe, mkfifo
Signal	✓	✓	✓	-	kill
VH-VE SHM *2	-	-	✓	-	vh_shmat
VESHM *3 *5	✓	✓	-	-	-
CR *5	✓	✓	-	-	-
VEMM *4 *5	-	-	-	✓	-

✓ Supported - Not supported



IB HCA: InfiniBand HCA

*1 robust mutexとpi mutexはサポートされていません
*2 VH-VE SHMは、VH側に作成したSystem V shared memoryに、VEプロセスがデータ転送を可能にする機能です
*3 VESHMは、VEプロセス間でデータ転送を可能にする機能です
*4 VEMMはIBドライバによるVEメモリへのアクセスやVEプログラムによるIB HCAへのアクセスを可能にする機能です
*5 これらの機能は、MPIライブラリまたは他のシステムライブラリで使用されます。 ユーザープログラムから直接使用するためのものではありません

諸元一覧(VE3)

項目	条件	VE当たり	VEプロセス当たり	注意
VEプロセス最大値	-	256プロセス	-	-
スレッド最大値	-	1024スレッド	64スレッド	
VEOSデーモンが受け入れる最大接続数	-	1056	-	VEOSデーモンと代理プロセス間の通信はVEスレッドごとに確立されます。 VEOSと移植されたコマンド/gdbとの間の接続は、要求ごとに確立されます
VEプロセスへ割り当てられるメモリ最大値	96GB メモリモデル	98,176MB	98,176MB	64MBページはゼロページのためリザーブされています。 64MBページが同期用にリザーブされています。
アクセス可能なリモートVEメモリの最大値	16 コアモデル	15360GB	15360GB	コア数が変わると、値が変わります。
ユーザーモードDMAのディスクリプタテーブル	-	-	2 ディスクリプタテーブル	-
スレッドのCRページの最大値 *1	-	-	1 ページ	-
MPIのローカルCRページの最大値 *1	16 コアモデル	16 ページ	3 ページ	コア数が変わると、値が変わります。
MPIのリモートCRページの最大値 *1	-	-	32 ページ	-

*1 1CRページは32CRで構成されています。

諸元一覧(VE1)

項目	条件	VE当たり	VEプロセス当たり	注意
VEプロセス最大値	-	256プロセス	-	-
スレッド最大値	-	1024スレッド	64スレッド	
VEOSデーモンが受け入れる最大接続数	-	1056	-	VEOSデーモンと代理プロセス間の通信はVEスレッドごとに確立されます。 VEOSと移植されたコマンド/ gdbとの間の接続は、要求ごとに確立されます
VEプロセスへ割り当てられるメモリ最大値	48GBメモリモデル	49,024MB *2	49,024MB *2	64MBページはゼロページのためリザーブされています。 64MBページが同期用にリザーブされています。*4
	24GBメモリモデル	24,448MB *3	24,448MB *3	
アクセス可能なリモートVEメモリの最大値	8コアモデル	894GB	894GB	コア数が変わると、値が変わります。
ユーザーモードDMAのディスクリプタテーブル	-	-	2ディスクリプタテーブル	-
スレッドのCRページの最大値 *1	-	-	1ページ	-
MPIのローカルCRページの最大値 *1	8コアモデル	24ページ	3ページ	コア数が変わると、値が変わります。
MPIのリモートCRページの最大値 *1	-	-	32ページ	-

*1 1CRページは32CRで構成されています。
*2 VEがSX-Aurora TSUBASA A100-1 modelに接続されている場合は、49,086MB
*3 VEがSX-Aurora TSUBASA A100-1 modelに接続されている場合は、24,510MB
*4 VEがSX-Aurora TSUBASA A100-1 modelに接続されている場合は、 2MBページが同期用にリザーブされています

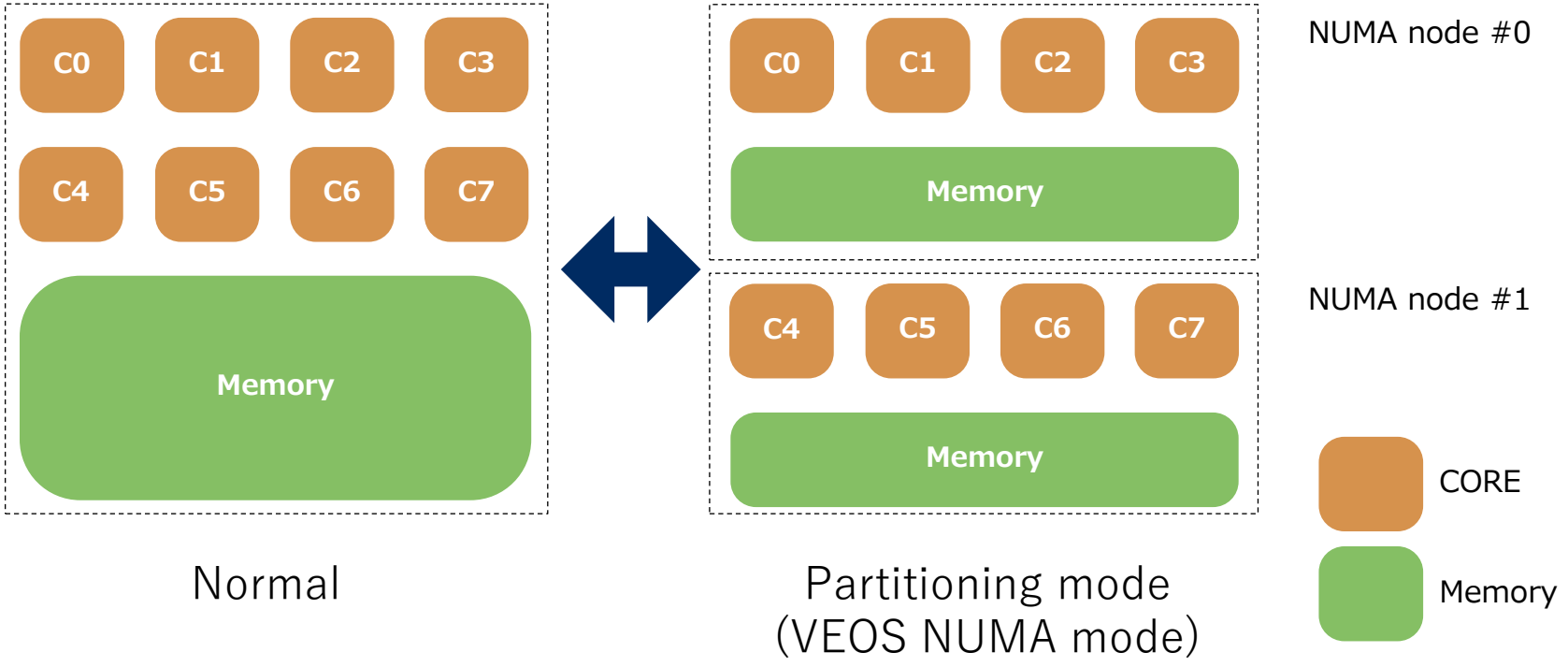
注意事項

1. ベストパフォーマンスを達成するためには、VE上で実行されるVEプロセスのスレッド数が、利用可能なVEコア数以下である必要があります
2. VEプロセスは新しいVEプログラム又はVHプログラムを実行することができます。VEプロセスが新しいVEプログラムを実行するとき、VEプロセスはexecve()システムコールの最初の引数でVEプログラムを指定する必要があります。VEプロセスがVHプログラムを実行すると、VHプログラムがVEプログラムを再度実行しても、リソース制限などのVEプロセスの情報は廃棄されます
3. ほとんど全てのシステムコール、標準的Cライブラリ関数、Linuxコマンド、デバッガーコマンドがサポートされています。しかしいくつかサポートされていないものも存在します
4. VE1との互換性のため、VE3のテキストセグメントとデータセグメントの合計サイズは48GBに制限されます。

VEOSの拡張機能

VEパーティショニングモードのサポート (VEOS NUMAモード)

- VE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
- 既存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、２つのNUMAノードして扱います
- VEOSは、局所性を考慮して、コアとメモリをVEプロセスに割り当てます

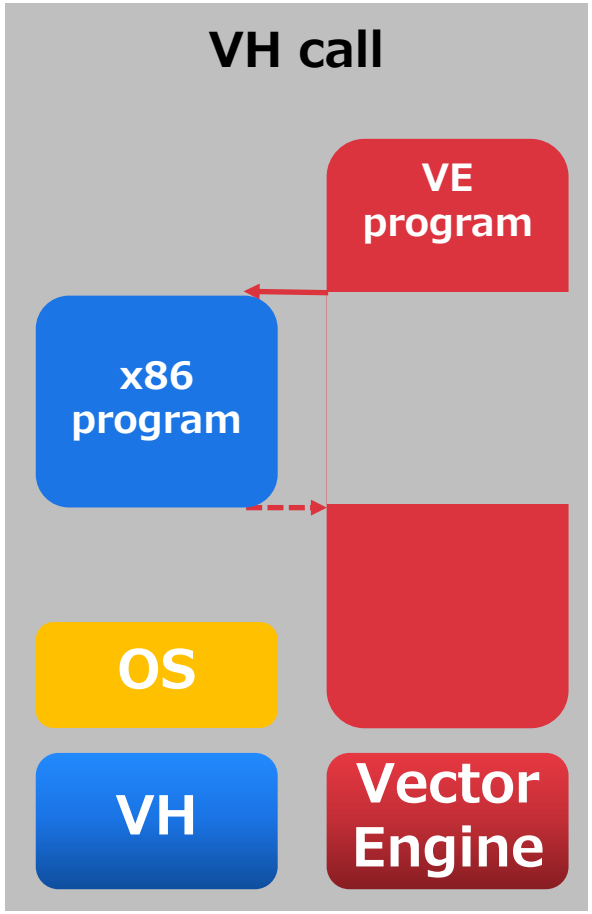


Normal

Partitioning mode
(VEOS NUMA mode)

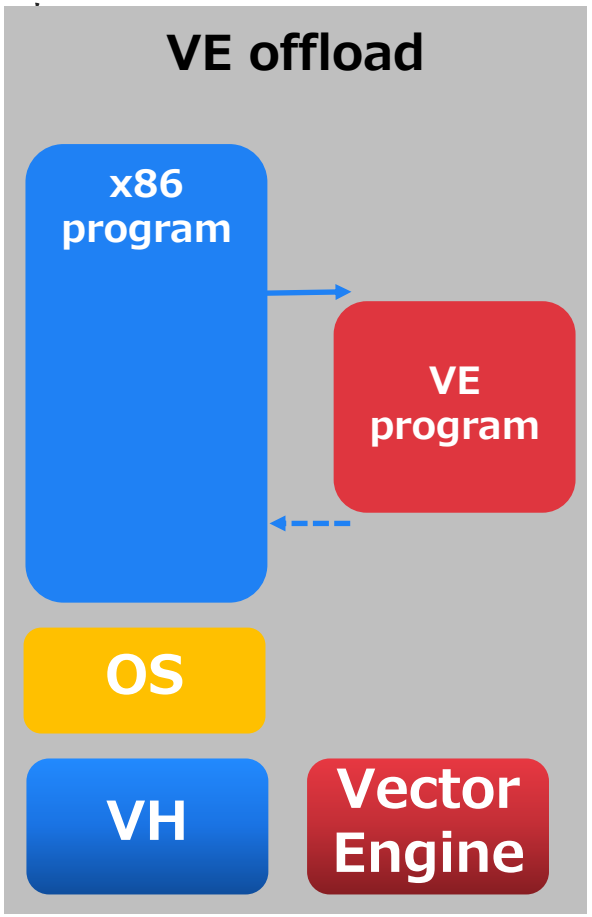
オフローディングプログラミングモデルのサポート

- 「VH call」はスカラ処理をVEからx86へオフロードします



VH callは同期APIです

- 「VE offload」は計算カーネルをx86からVEへオフロードしま

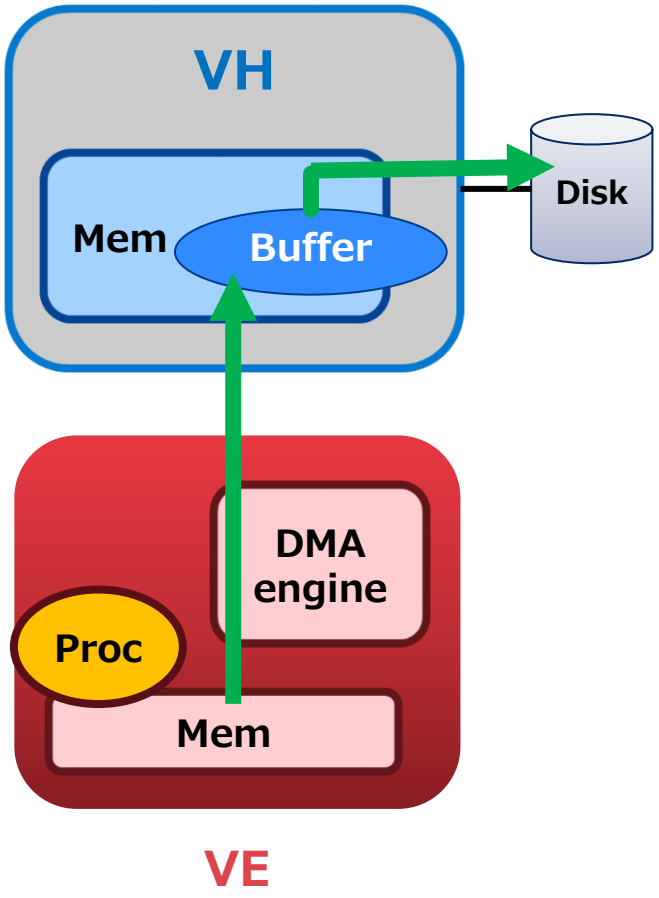


VE offloadは非同期APIです

高速I/O

- 以下のread/write系のシステムコールのスループットとレイテンシが改善します
  - read, pread, readv, preadv
  - write, pwrite, writev, pwritev
- 高速I/Oは、VE DMAエンジンを使って、VEとVH間でデータを転送します
  - VE DMAエンジンは直接VH側のバッファへアクセスします
- 高速I/Oを無効化するためには、以下の環境変数を設定する必要があります

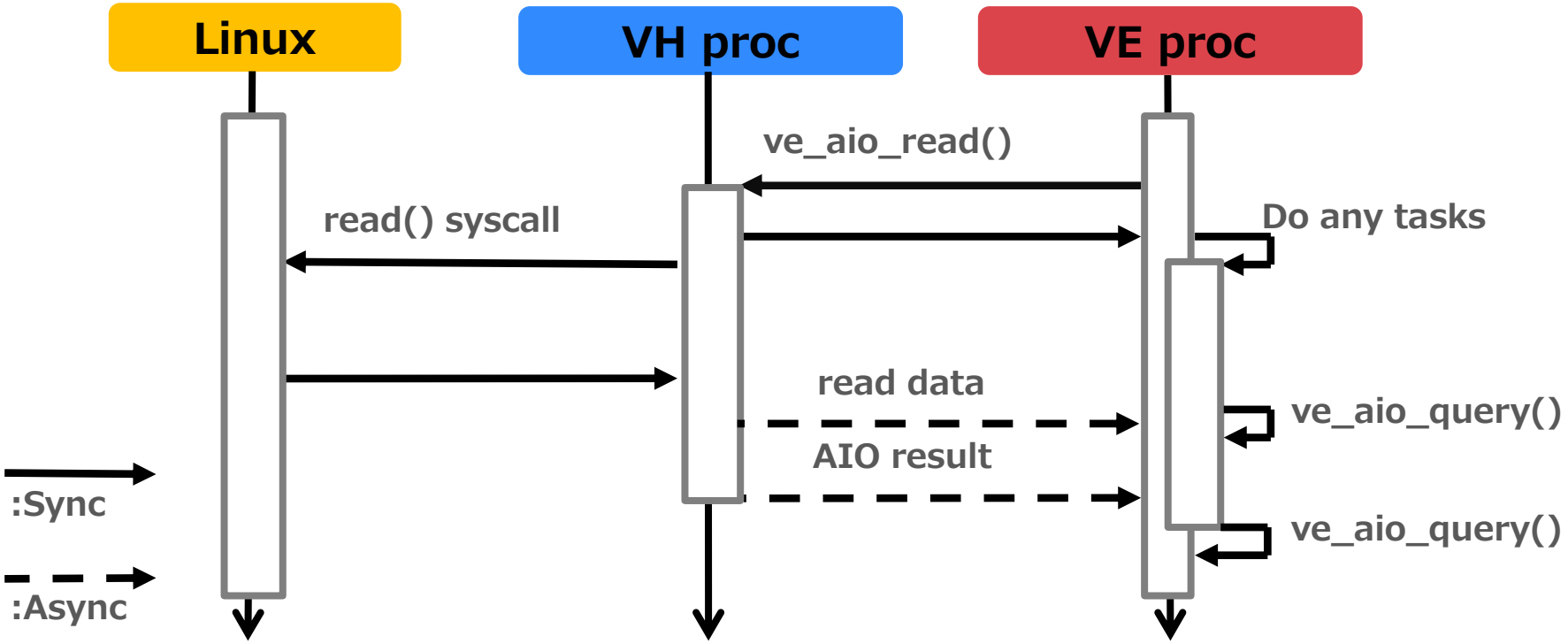
```
(In case of bash)
export VE_ACC_IO=0
```



非同期I/O

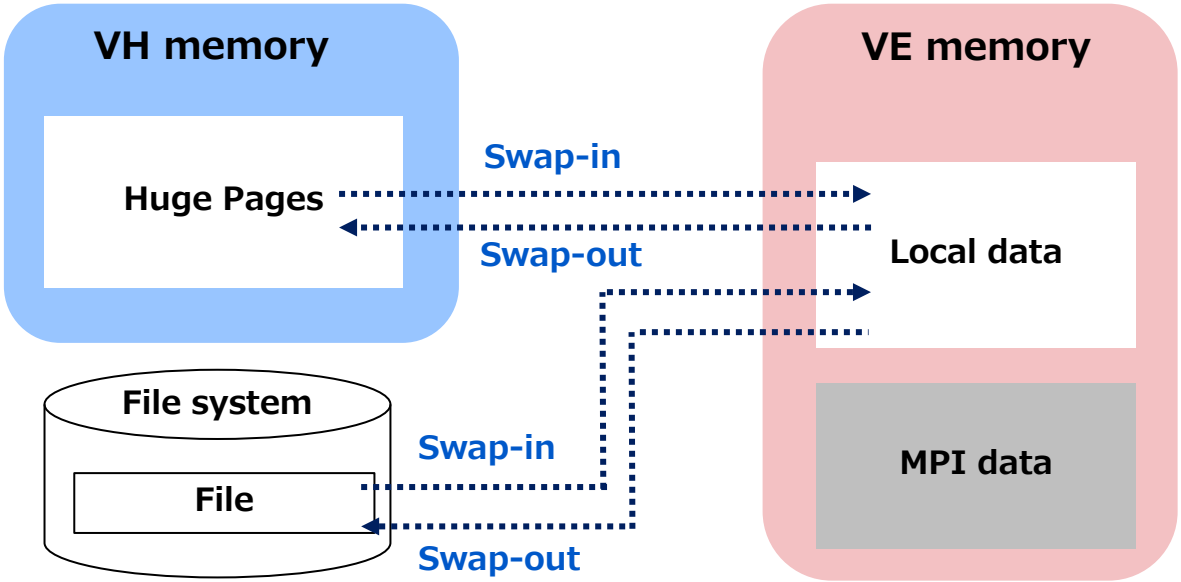
- 非同期I/O(VE AIO)は、I/Oシステムコールを VH側で処理している間も、VEプログラムが処理を進められるようにします

- 非同期I/Oを使ってVH側でI/Oを処理している間、VE側のリソースを必要としません



Partial process swapping

- 停止中のVEプロセスのローカルデータを含むVEメモリの一部を、 VH側のメモリまたは、ストレージ上のファイルに、スワップアウトします
  - 解放されたVEメモリは他のプログラムを実行するために利用することができます
- MPIデータを含むVEメモリの一部は、VEメモリに残り続けます
  - MPIプロセス間の通信は、エラーが発生することなく完了します

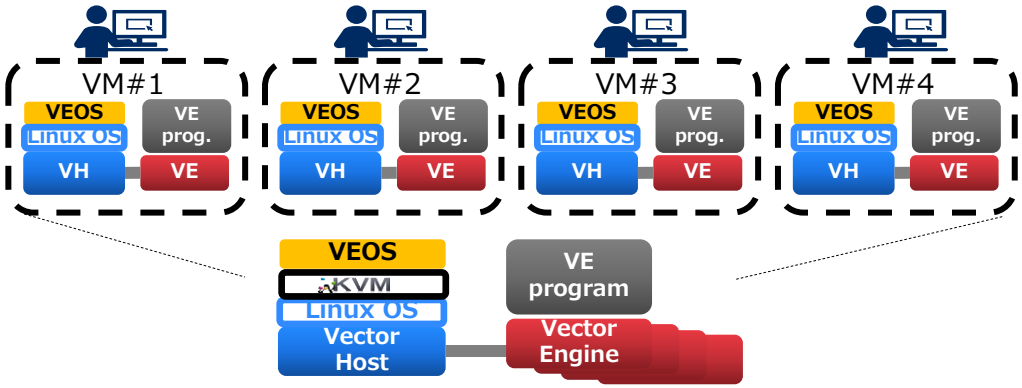


- VHメモリにスワップする場合、VEOSはVHメモリから割り当てられたhuge pagesにVEメモリをスワップアウトします。
- ファイルにスワップする場合、VEOSは分散ファイルシステムまたはローカルファイルシステム内のファイルにVEメモリをスワップアウトします。Huge pagesも、メモリの内容を転送するバッファとして使用されます。

仮想マシン対応

ユーザがVEを含む仮想マシン環境を排他的に利用することが可能です。

- VEノードの全て、或いは一部分を仮想マシン環境向けに設定できます。
- 各仮想マシンは１つのVEを占有します(※)。
  - 利用技術: KVM PCI passthrough
- 仮想マシンの管理にAurora固有の変更は無く、OSSの使い勝手のまま操作することができます。
  - 管理ソフト/仮想化プログラム: libvirt / qemu-kvm



(※) VEカードでは ATS (Address Translation Services) がサポートされないため、仮想マシンにおける　VEノード間の性能が十分に出ません。そのため、仮想マシン1台につき1つのVEでの運用を推奨します。

改訂履歴

改訂履歴

Rev.	日付	改訂内容
1.8	2018年2月28日	初版
1.9	2018年5月14日	• VEOSがLinux/VH上で動作することを明確にするため、VEOSの紹介を更新した • VEOS 1.1の新機能であるVH-VE SHMを追加した
1.10	2018年7月11日	• VEOS 1.2.1以降では、VEコア間でのスレッドの移行が起きることを説明した • 最大メモリ量の記述を更新した • VEOS 1.2で行った、DMAディスクリプタテーブルの数の変更を反映した
1.11	2018年9月12日	• User mode DMAのための低レベルAPIが実験的機能として提供されるため、user mode DMAの説明を更新した • VEプログラムがVH-VE SHMを利用可能なため、機能一覧を更新した • Fifoを機能一覧に追加した • 図を改善した
2.0	2018年12月5日	• VEOS 2.0以降では、VEOSの一部としてカーネルヘッダを提供するため、カーネルヘッダに関する注意事項を削除した
2.1	2019年2月13日	• この版はVEOS v2.0.1 以降に対応します • 表紙の書式を変更 • このドキュメントが対象とする VEOS バージョンに関する情報を追加した • 用語を訂正した 擬似プロセス→代理プロセス
3.0	2019年4月22日	• この版はVEOS v2.1以降に対応します • 「VEOSとは」を更新 • 代理プロセスがve_execコマンドであることを記載 • User mode DMAの低レベルAPIは、実験的機能ではなくなったことを反映 • VEOSの拡張機能を記載
3.1	2019年6月19日	• VEOS NUMA モードについての説明を記載

改訂履歴(続き)

Rev.	Date	Update
3.2	2019年9月	• この版はVEOS v2.2以降に対応します • メモリアドレスマップを変更
3.3	2020年1月	• この版はVEOS v2.3以降に対応します • VE_ACC_IO=1を設定することで高速I/Oを有効にできることを記載 • 新機能であるpartial process swappingの説明を追加
3.4	2020年2月	• 誤記を修正
3.5	2020年7月	• この版はVEOS v2.6.2以降に対応します • Partial process swapping to fileの説明を追記 (Page 21)
3.6	2020年9月	• この版はVEOS v2.6.2以降に対応します • 体裁を改善
3.7	2021年3月	• この版はVEOS v2.11以降に対応します • RDMAサポートのためアドレスマッピングを変更 (Page 8)
4.0	2023年3月	• この版はVEOS v3.0.2以降に対応します • VE3向けのアドレスマッピングに更新(Page 9) • VE3向けの利用可能なリソースを追加(Page 14) • 仮想マシン対応の説明を追加(Page 22)
4.1	2024年2月	• 見出しを修正(Page 14,15)