2001年3月23日

VPP Fortranの設計思想と特徴

富士通株式会社

基盤テクノロジ開発部

山中 栄次

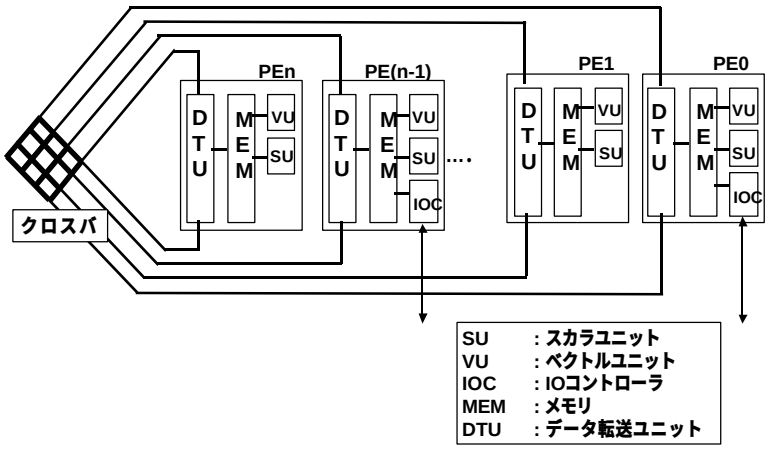
目次

- VPP5000の概要
- 並列プログラミングモデル
- VPP Fortranの設計思想
- VPP Fortranの特徴

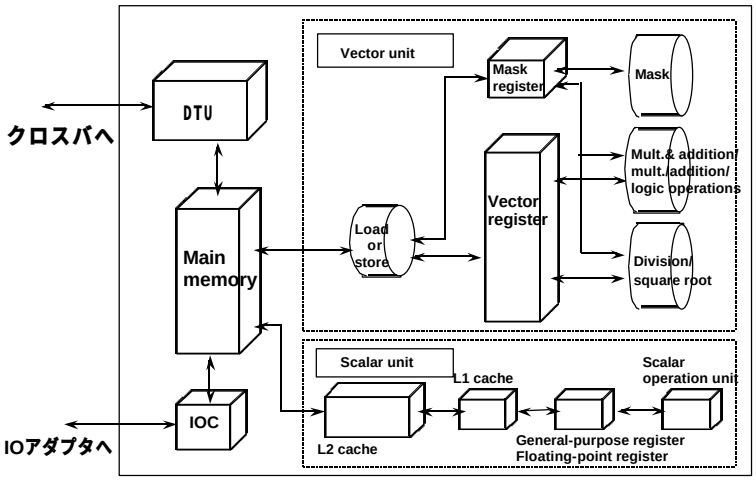
VPP5000の概要

- システムの概要
- PEの概要
- 諸元

VPP5000 (システムの概要)



VPP5000 (PEの概要)



VPP5000 (諸元)

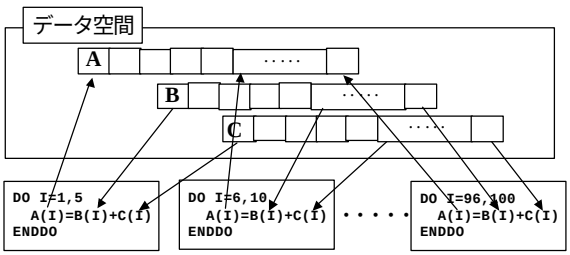
CPU	ピーク性能/PE		9.6 Gflops
	ベクトルパイプライン数		4
	レジスタ	ベクトル	128 KB
		汎用	32 (64 bit)
		浮動小数点	64 (64 bit)
	キャッシュ		1次: 128 KB 2次: 2 MB
	DTU性能		1.6 GB/sec x 2 (Send/Receive)
主記憶	メモリ容量		4, 8, 16 GB
	メモリスループット		76.8 GBps

並列プログラミングモデル

- データパラレルモデル
- メッセージパシングモデル
- 並列プログラミングモデルの対応

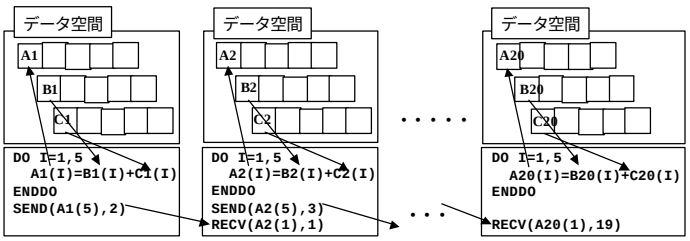
データパラレルモデル

- データ空間を共有して，処理を並列化するモデル
- データ空間を共有するため，共有メモリ型の並列計算機に向いた方式
- VPP Fortran，HPF，OpenMPなど



メッセージパシングモデル

- データ空間と処理を分割し，メッセージの送受信で連携をとるモデル
- データ空間の分割を前提としているため，分散メモリ型の並列計算機に向いた方式
- MPI，PVMなど



並列プログラミングモデルの対応

	データパラレルモデル	メッセージパシングモデル
共有メモリ	◎	○
分散メモリ	△	◎

◎：自然な対応
○：動作可能
△：仮想グローバル空間を実現することで可能

VPP Fortranの設計思想

- VPP Fortranの設計思想
- プログラミング方法の比較

VPP Fortranの設計思想（1）

- プログラミングの容易さ
  ➔データパラレルモデルの堅持
- 高性能
  ➔ユーザと処理系の役割の明確化
    ➔シンプルな仕様

結果として，Unix/Cのように，
New Jerseyアプローチ“Worse is Better”
を実践.

VPP Fortranの設計思想 (2)

- ユーザと処理系の役割の明確化
  - ユーザの分担
    - リモートメモリアクセスのコントロール
  - 処理系の分担
    - 演算の高速化
      - ➔自動ベクトル化，最適化など
    - 通信の高速化
      - ➔通信パケットスケジューリングなど

プログラミング方法の比較

	HPF	VPP Fortran	MPI
パラダイム	データパラレル	データパラレル	メッセージパシング
プログラミングの難易度	容易？	容易	難しい
性能	高速化に課題が多い	高速	高速
流通性	スタンダードな仕様	独自仕様	スタンダードな仕様
処理系開発の難易度	難しい	普通	普通

VPP Fortranは，以下の場合に適した言語
- 簡単に高性能なプログラムを書きたい.
- ソースの流通性は気にしない.

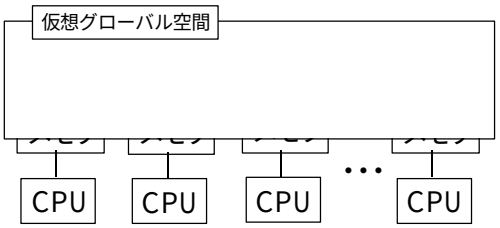
VPP Fortranの特徴

VPP Fortranのキーコンセプト

- 仮想グローバル空間
- インデックスによる分割
- 明示的通信

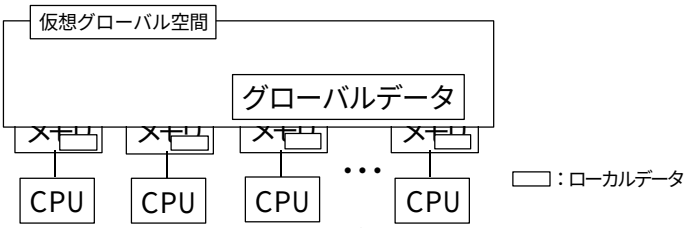
仮想グローバル空間

データパラレルモデルの実現のために，複数のPE上のメモリを用いて，ソフト的に実現した共有メモリ空間



データ配置

- 仮想グローバル空間上のデータ（グローバルデータ）
  - 全てのPEからグローバルデータの全範囲をアクセス可能
  - グローバルデータへのアクセスは低速
    - ➔自PE上のデータを高速にアクセスする方法を用意
- 通常空間（ローカル空間）上のデータ（ローカルデータ）
  - 自PE上のデータのみアクセス可能
  - ローカルデータへのアクセスは高速

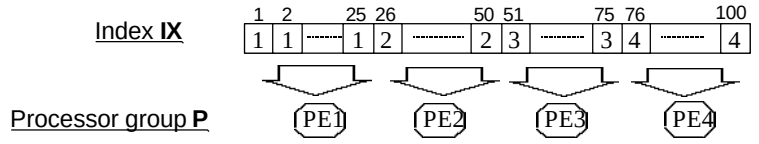


インデックスによる分割

- インデックスとは?
  - 配列の添字と，ループのイタレーションの両者を扱うための抽象化インデックス.
- インデックスの目的は？
  - データと演算の分割を一致させて，リモートアクセスを減らす.

インデックス分割（1）

- インデックス分割は，プロセッサグループ上における，データと処理の分割を対応させる．
- INDEX PARTITION ディレクティブを用いて，インデックス分割（名前，形状など）を宣言する．




```
!XOCL PROCESSOR P(4)
!XOCL INDEX PARTITION IX
       = ( PROC = P, INDEX = 1:100, PART = BAND )
```

19

インデックス分割（2）

INDEX PARTITIONディレクティブによる指定項目

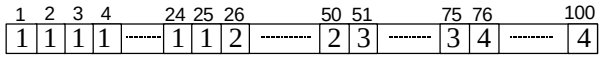
- PROC
  - プロセッサグループの名前を指定
- INDEX
  - インデックスの上下限を指定
- PART
  - PEへのマッピング方式を指定
- OVERLAP
  - 隣接するPEに重複してマッピングされる範囲を指定

20

インデックス分割 (PART節)

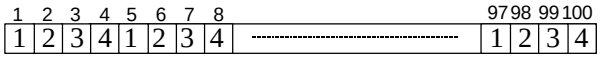
- BAND

  インデックスを等分割して配置する.



- CYCLIC

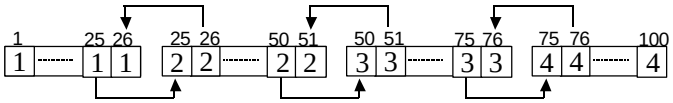
  インデックスをサイクリックに配置する.



インデックス分割 (OVERLAP節)

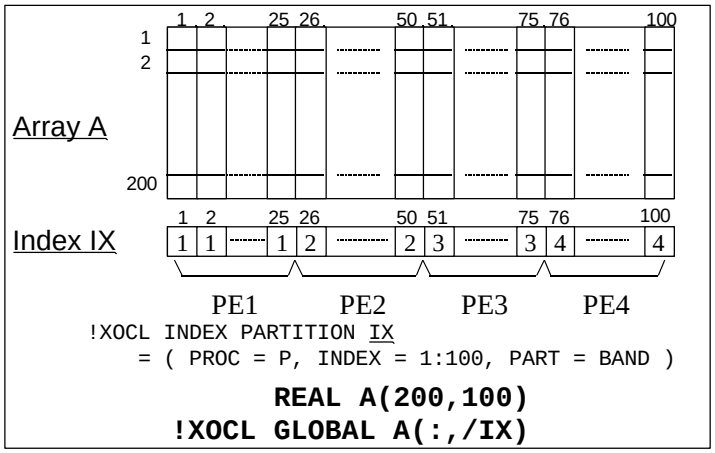
- OVERLAP

  分割の境界部分を重複して配置する.



グローバル配列の分割

インデックス分割によりグローバル配列の分割形状を記述する．



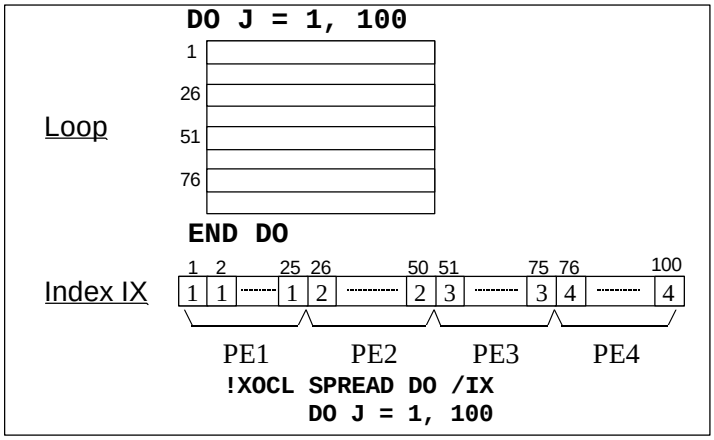

```
!XOCL INDEX PARTITION IX
    = ( PROC = P, INDEX = 1:100, PART = BAND )
```

```
REAL A(200,100)
!XOCL GLOBAL A(:,/IX)
```

23

ループ分割

- SPREAD DOディレクティブを用いて，ループの分割を記述する．
- インデックス分割を付加することにより，インデックス分割にしたがってループを分割する．




```
!XOCL SPREAD DO /IX
      DO J = 1, 100
```

24

ループ分割とデータ分割の統合

- ループとデータの分割に同じインデックス分割を適用することにより，リモートアクセスを排除する．
- RESIDENT 節の導入
  - グローバルデータが，ローカルデータと同様に高速にアクセス可能なことを指示する．
  - グローバルデータを，自プロセッサに配置された範囲に限定してアクセスする場合に指定する．

```
      REAL A(200,100)
!XOCL GLOBAL A(:,/IX)
        :
!XOCL SPREAD DO /IX RESIDENT(A)
      DO J=1,100
        DO I=1,200
          A(I,J)=…
        ENDDO
      ENDDO
!XOCL END SPREAD DO
        :
```

25

明示的通信（一覧）

- 必要なリモートアクセスを最小限にコントロールできるように，明示的な通信ディレクティブを導入する．
  - SPREAD MOVEディレクティブ
    - 多対多の一括通信
    - 分割データのトランスポーズなど
  - OVERLAPFIXディレクティブ
    - OVERLAP領域を，マスターの値で更新する．
  - BROADCASTディレクティブ
    - 1対多の通信
  - UNIFYディレクティブ
    - BROADCASTの一括化

26

明示的通信（SPREAD MOVE）

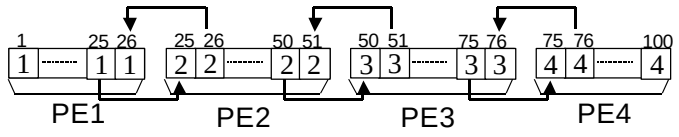
- 配列代入の形式で記述する.

```
      REAL A(200,100),B(100,200)
!XOCL GLOBAL A(:,/IX),B(/IX,:)
        :
!XOCL SPREAD MOVE /IX RESIDENT(A) ID(WID)
      DO J=1,100
        DO I=1,200
          A(I,J)=B(J,I)
        ENDDO
      ENDDO
!XOCL END SPREAD MOVE
        :
!XOCL MOVEWAIT(ID)
```

27

明示的通信（OVERLAPFIX）

- OVERLAP領域を，隣接するPE上のマスター領域の値で更新する.



28

Fujitsu
Core Technology