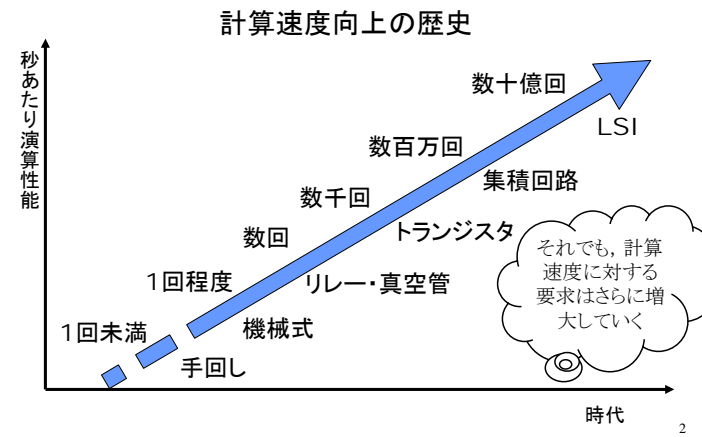


コンピュータシステムII(2)

情報基盤センター
天野 浩文

前回のおさらい(1)



前回のおさらい(2)

- 計算速度に対する要求は増大しつづけているのに...
 - 単体の計算機の性能向上だけで100倍, 1000倍といった高速化を達成するのは非常に困難
- 一方で, 計算機を構成する部品の値段は, 半導体技術の進歩と, 計算機の普及により, 急速に低下しつづける



並列処理

前回のおさらい(3)

- 並列処理における問題の例
 - プログラムを書くのがややこしい
 - 1台のコンピュータ用に書いたプログラムを自動的に並列化できるとよいのだが, 自動並列化の技術は, まだまだ開発途上
 - 並列処理に参加するコンピュータの負荷が均一でないと, 効率が低下する
 - 並列処理に参加するコンピュータの台数が多くなると...
 - 計算している時間よりも, コンピュータどうしが歩調を合わせる(互いの進み具合を確認する)ための処理の時間のほうが長くなる
 - 計算している時間よりも, 計算に必要なデータをやりとりする(データを見せ合う)時間のほうが長くなる

前回のおさらい(4)

- 並列処理 (parallel processing) とは何か
 - 複数のプロセッサを同時に用いてひとつの処理を行うこと.
 - 単一のプロセッサではやりたい仕事をするのに十分な処理性能が得られない場合に, 複数のプロセッサを同時に用いて処理を行うこと.
- 分散処理 (distributed processing) とは何か
 - 異なる場所にある複数の計算機を同時に用いてひとつの処理を行うこと.
 - 何かの理由で一カ所にある計算機だけではやりたい仕事が遂行できない場合に, 複数の場所に分散した計算機を同時に用いてその処理を遂行すること.

5

前回のおさらい(5)

- 並列処理に用いられるハードウェアの例
 - **並列計算機**: 並列処理用に強化された特殊な計算機
 - 複数のプロセッサ
 - メモリは個別に持つ場合と全体で共有する場合がある.
 - 相互結合網
- 分散処理に用いられるハードウェアの例
 - インターネットで接続された, 遠隔地に分散する複数の計算機
 - 計算機の構成 (プロセッサ種別, メモリ量など) は異なることが多い.
 - 遠隔地にあるプロセッサを連携させたり制御したりする特別なハードウェアは持たない.
 - それぞれの計算機のメモリは共有されない.
 - 並列計算機のような特別な通信用ハードウェアは持たない.
- 並列処理に用いられるハードウェアと分散処理に用いられるハードウェアの間に, 厳密な境界が存在するわけではない

6

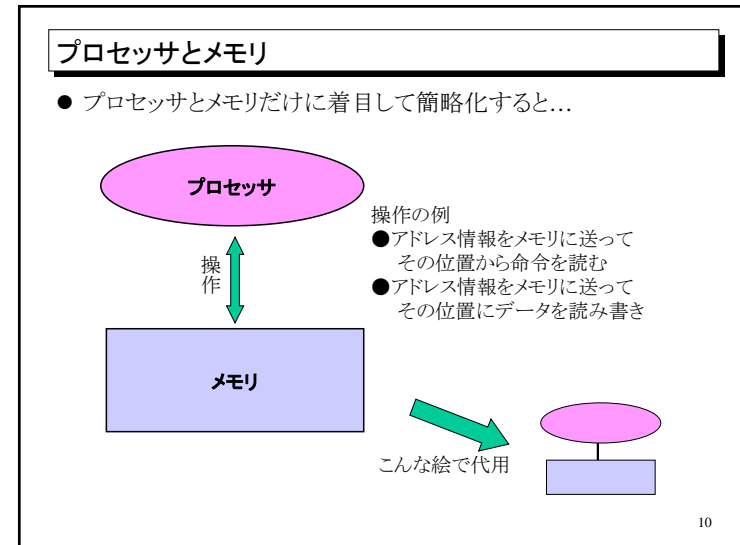
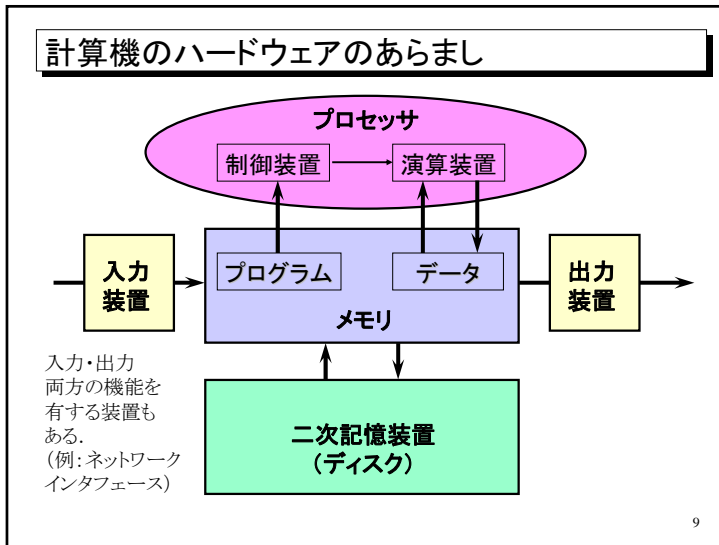
2. システム構成

7

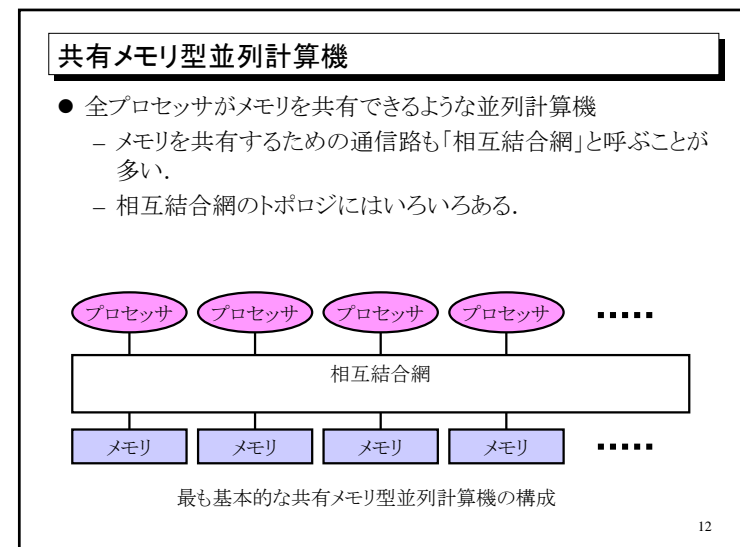
2.1 プロセッサとメモリ (並列計算機の構成)

広域分散処理システムについては, 2.2節で説明する.

8



- ### 並列計算機ハードウェアの分類
- 分類のしかたはいろいろあるが...
 - メモリの形態で分類すると...
 - 共有メモリ型並列計算機
 - UMA型
 - NUMA型
 - 分散メモリ型並列計算機(NORA/NORMA型)
 - 広域分散処理システムやクラスタシステムは、概念的には、分散メモリ型並列計算機と同じ構成をしている。
- 分散メモリの上でソフトウェア的に共有メモリをエミュレートしたものをNUMAの仲間に入れることもあるようだが、本来NUMAはハードウェアの分類に用いられる用語である。
- エミュレート:一部他の方法(部品)を使って元の方式と同じ効果を得ること。
- 11



UMA (uniform memory access) 型システム

- メモリのどの位置にアクセスする場合でも同じ時間で処理できるような共有メモリ型並列計算機
 - 下の図の赤い矢印に相当するアクセスがいずれも同じ時間で処理されるなら, UMA型である.

相互結合網

13

最も単純なUMA型システム

- ただ1つのメモリをバスで共有
 - 同時に1つのプロセッサしかメモリにアクセスできない.
 - メモリの容量は小さい.
 - あまり多数のプロセッサは搭載できない.

バス

メモリ

14

もう少し複雑なUMA型システム

- プロセッサと同数のメモリをバスで共有
 - 同時にメモリアクセスできるプロセッサは1つのままだが, メモリの総容量は大きくできる.

バス

15

さらに複雑なUMA型システム

- クロスバススイッチを用いるもの
 - ちょっと高価

↑ ↓ どちらか

交点にある各スイッチは独立に制御できる

16

NUMA (non-uniform memory access) 型システム

- メモリの位置によってアクセス時間に差が生じるような共有メモリ型並列計算機
 - 下の図の赤い矢印に相当するアクセスに要する時間が場所によって異なるなら、NUMA型である。

この図は、4つのプロセッサ（ピンクの楕円）と4つのメモリ（青い長方形）が「相互結合網」を通じて接続されている様子を示しています。赤い矢印は、各プロセッサから遠く離れたメモリへのアクセス経路を示しており、これがアクセス時間の差を生じる原因となります。

17

NUMA型システムの例

- 近い場所のメモリと遠い場所のメモリでアクセス時間に差が出るような相互結合網の例

この例では、プロセッサが2個ずつペアを組み、バスによってメモリを共有している。「近い」アドレス（他のペア配下のメモリ）にアクセスする場合でも、上位の相互結合網を通じて同じ機械語命令でアクセスできる。アクセス先のプロセッサが仕事を代行する必要はない。ただし、遠いアドレスへのアクセスに要する時間は長くなる。

18

NUMA型並列計算機のもうひとつの構成例

- 各プロセッサが固有に持つローカルメモリと全プロセッサで共有されるメモリ（コモンメモリ）の両方がある例
 - ローカルメモリのアクセスは早い
 - コモンメモリのアクセスは遅い
 - メモリへのアクセスは、いずれもハードウェアによってサポートされる

この構成では、各プロセッサ（ピンクの楕円）が自身の「メモリ」（青い長方形）を持ち、さらにシステム全体で共有される「コモンメモリ」も存在します。これらは「相互結合網」を介して接続されています。

19

分散メモリ型並列計算機(1)

- 各プロセッサが固有のメモリを持ち、他のプロセッサのメモリには直接アクセスできないような並列計算機
 - アクセスできないところは、他のプロセッサにメッセージを送って処理（アクセスの代行）を依頼する。
 - ・ 太い赤線のような経路
- NORA/NORMA (no remote memory access) 型と呼ぶこともある。

この図は、各プロセッサ（ピンクの楕円）が独自の「メモリ」（青い長方形）を持つ分散メモリ型システムを示しています。太い赤線は、メモリを共有できない場合、他のプロセッサにアクセスを依頼するためのメッセージ経路を示しています。

20

演算ノード

- ソフトウェアを実行できる最小の単位
 - 最低限以下の2つの要素を含む
 - プロセッサ
 - メモリ
 - ただし、共有メモリ型並列計算機の場合は定義が難しい
 - プロセッサとローカルメモリの組を演算ノードと考える
 - 共有メモリ型並列計算機が複数つながれる場合にのみ、それぞれの並列計算機を演算ノードと考える
 - 通常の計算機は、それ単体で演算ノードになりうる
- ここまでの講義で使った絵との対応:

プロセッサ

↓

メモリ

→

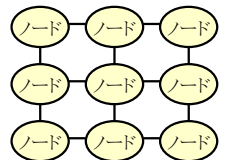
演算ノード

メーカーは、特にこれらの立場を採るものが多い

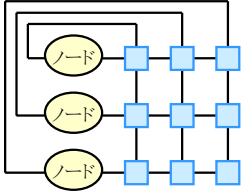
21

分散メモリ型並列計算機(2)

- 分散メモリ型並列計算機の相互結合網の分類
 - 静的ネットワーク
 - 演算ノードの位相幾何学的な構成(トポロジ)が変わらない
 - 動的ネットワーク
 - スイッチによって演算ノードの位相幾何学的構成を動的に変更可能



静的ネットワークの例(メッシュ)

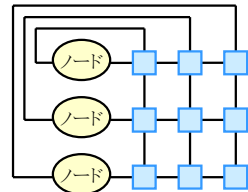


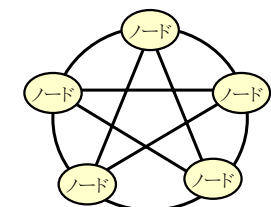
動的ネットワークの例(クロスバー)

22

動的な相互結合網の例

- クロスバーによる接続
 - 任意の演算ノード間の接続経路を動的に作れる
 - 完全結合に近い構成だが...
 - 大規模なシステムでは、コストが高い。
- 完全結合:すべてのノード間に直接の接続経路があること。
 - 通信のしやすさでは理想的だが、非常にコストが高い。
 - ノード数 n に対して
 - 各ノードから $(n - 1)$ 本
 - 全体で $n(n - 1)$ 本





23

共有メモリ型と分散メモリ型の比較(1)

- プログラミングのしやすさ
 - ⇒ 共有メモリ型に軍配
 - 共有メモリ型
 - メモリのどの位置にアクセスする場合でも、同じ命令でアクセス可能
 - 分散メモリ型
 - 位置に応じて、メモリアクセス命令と通信命令を使い分ける必要がある

24

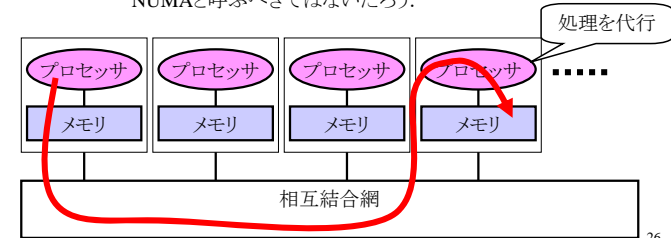
共有メモリ型と分散メモリ型の比較(2)

- ハードウェアのコスト
⇒分散メモリ型に軍配
- メモリ共有用の相互結合網は
 - 安価な手法では、大規模なものが作りにくい
 - 大規模にできるものは、一般に高価
- 分散メモリ型のほうが安価に大規模なシステムを作りやすい。

25

分散メモリ型の欠点を補うために

- 分散共有メモリ(distributed shared memory, DSM)
 - あるプロセッサが「遠い」メモリにアクセスしようとする時、自動的に他のプロセッサにアクセスを代行させるようなしきりを作る。
 - ハードウェアだけで作れば、NUMAと呼べるかも。
 - 例外処理ルーチンなど、ソフトウェアの力を使えば、それはもうNUMAと呼ぶべきではないだろう。



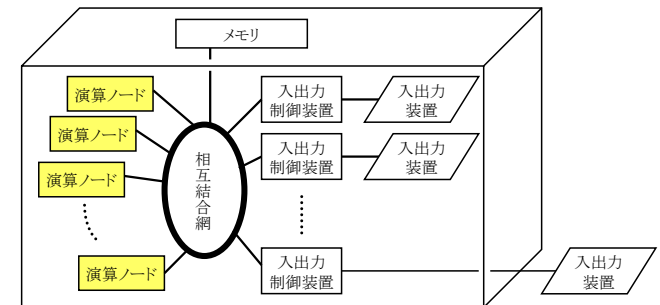
26

2.2 演算ノードと通信路

27

並列計算機の構成

- 多数の演算ノードを高速の通信路(相互結合網)で結ぶ。
- 全体がひとつの計算機システムとして構築される。
- 入出力装置は演算ノード間で共有されることもある。



28

クラスタシステムの構成

- 単体で計算機として動作可能な演算ノードを(高速な)通信路で結ぶ。
 - この通信網も相互結合網と呼ぶことがある。
 - 演算ノードは均一であることが多い。
 - 演算ノードは通常は非常に近い位置に配置される(室内・館内)。
- 演算ノードになる計算機は小規模なものが多い。
 - PC(パーソナルコンピュータ)
 - WS(ワークステーション)
- 通信路に用いられるハードウェアの例
 - 100Mbpsイーサネット(スイッチ)
 - 1Gbpsイーサネット(スイッチ)
 - その他の高速ネットワーク
 - 例: Myrinet(2Gbps超)

29

広域分散処理システムの構成

- 単体で計算機として動作可能な演算ノードを通信路で結ぶ。
 - 通信路も単一とは限らない。
 - 各演算ノードはLANを経由してさらに広域の通信路につながる。
 - 演算ノードもさまざま。
 - 演算ノードは広域に分散している。

30

通信制御処理に関連するハードウェア

- 構成要素
 - プロセッサ(CPU)
 - メモリ(memory module, MM)
 - 通信インタフェース制御部(network interface controller, NIC)
- 通信処理
 - メモリの上にあるデータを送信
 - メモリ上にデータを受信
- 通信制御のやり方(機能分担)は2通り
 - CPUが通信制御処理の大半を行い、NICは送信のみを担当
 - CPUは通信をNICに依頼するのみで、通信制御処理の大半をNICが実行

どちらを選ぶかは、CPUと通信路の性能の兼ね合いで決める

31

DMA (direct memory access)

- MMとNIC内の通信データ格納域との間のデータのコピーを、CPUに頼ることなく行う機能
 - CPUやNICの処理負荷を軽減
 - DMAC(DMA controller)という部品を追加することで実現

注意: DMAは、通信以外の入出力装置でも用いられることがある

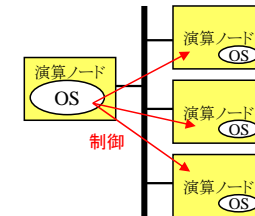
32

2.3 演算ノードとオペレーティングシステム

33

演算ノードとオペレーティングシステム(1)

- 1つの演算ノード上で動作するOSが全演算ノードを制御する方式 (並列計算機に向く)
 - システム全体で効率的にソフトウェアを実行することができる
 - 通信路は高速でなければならない(広域分散では使えない)
 - 全体を制御するノードに障害が発生するとシステム全体がダウンする
- 残りの演算ノードは以下のいずれか
 1. OSを持たない方式(マスタ・スレーブ方式)
 - 共有メモリ型でのみ可能
 - 上記のノードが「マスタ」、残りのOSなしのノードが「スレーブ」となる。
 - 小規模なシステムでのみ有効
 2. 同種のOSが残りの全ノードでも動くが、上記の制御担当ノードのOSが主導的な役割を担う方式

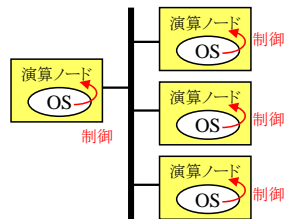


教科書p22の図では、制御される側のノードにまったくOSがない。左の分類の2の場合、上の図のようになる。

34

演算ノードとオペレーティングシステム(2)

- 各演算ノードのOSがそれぞれを制御する方式(分散システムに向く)
 - 演算ノードが広域に分散しても動作可
 - 通信に要する時間が長くなると、遠くの演算ノードの制御は困難
 - いくつかの演算ノードが停止してもシステムは動作可能
 - 他のノードに依存することなく動作可能
 - システム全体で効率的にソフトウェアを実行するのは困難



35

並列OSの構築方式

- 既存の単一プロセッサ用のOSを拡張する方式
 - 複数の演算ノードを制御するための機能を追加、など
 - 既存のアプリケーション(application program, AP)がそのまま動作する。
 - 拡張に困難が伴ったり、実現できる機能・達成できる性能に制約があったりする。
- 並列計算機のハードウェアに合わせて新規に作成する方式
 - 新規に作成する手間が大変
 - 既存のAPIは動作しない可能性が高い。

15年くらい前までは、並列計算機(主に研究用)のメーカーが独自のOSを提供することも多かったが、最近の商用システムでは、既存のソフトウェア資産を保護することが重視されるため、単一プロセッサ用の既存OS(のうち、UNIXを選択して)を拡張する方式が主流となっている。

36

最近の商用並列計算機のOSの例

- SGI Altix 3000シリーズ
 - 256プロセッサ用LINUX(拡張版)
- 富士通 PRIMEPOWER HPC2500
 - 128プロセッサ用Solaris
- IBM eServer pSeries
 - 64プロセッサ用AIX
- ヒューレット・パッカード Integrity Superdome
 - 64プロセッサ用LINUX(拡張版)
 - HP-UX, Windows Server 2003も動作

37

分散OSの構築方式

- 分散処理システムの特徴
 - 分散処理を行わないときには, 各演算ノードが単体でも計算機として動作しなくてはならない
 - 多種多様なハードウェアが混在することが多い
 - 同種のハードウェアが広域に分散するのでない限り, それぞれの上で共通に動くOSを新規開発するのは絶望的
 - 研究目的をのぞくと, 新規開発はあまり行われない
- 典型的な分散OSの構築法
 - 既存のOSを拡張する
 - 既存のOSの上に, 広域分散処理を行うための基本ソフトウェアを載せる
 - ハードウェアとアプリケーションの中間という意味で, ミドルウェアと呼ぶことがある.

38

実際の並列計算機・クラスタシステム

九州大学情報基盤センターに設置されている並列計算機やクラスタの一部を紹介する

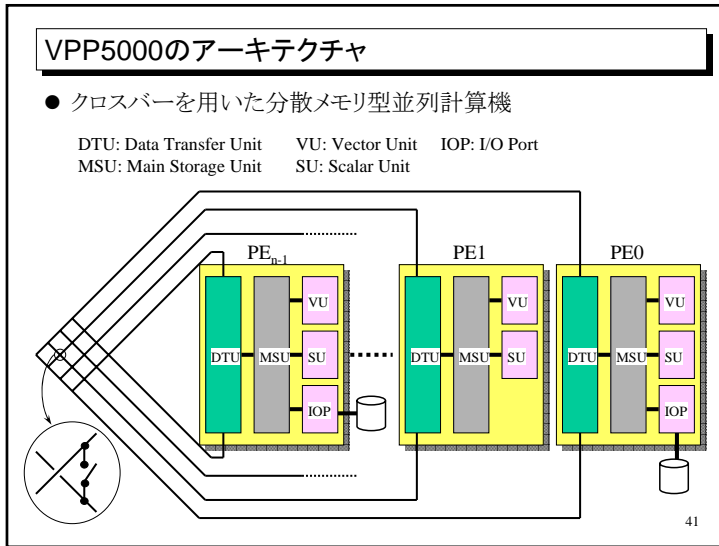
39

富士通VPP5000/64

- 9.6GFLOPSのベクトル演算ノードを64台搭載したベクトル並列型スーパーコンピュータ
 - 各ノードでOS: UXP/V(富士通の開発したUNIX)が動作
 - ノード(PE)の0番がシステムの全体を制御



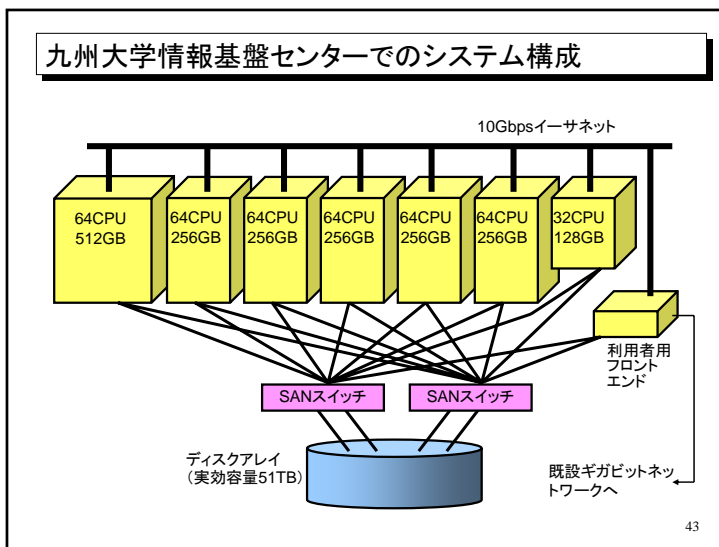
40



IBM eServer p595

- IBM POWER5プロセッサ(1.9GHz)を16・32・48・64台搭載可能な共有メモリ型並列計算機
- OSはAIX (IBMの開発したUNIX)

42

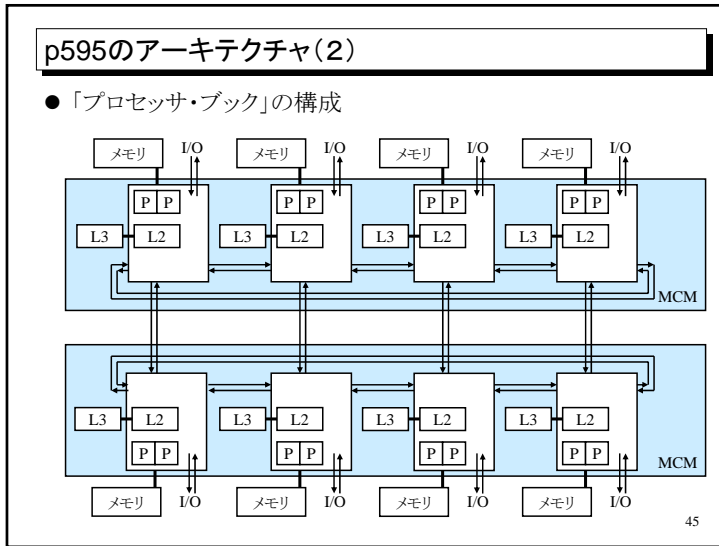


p595のアーキテクチャ(1)

- NUMA型
- 1つのチップに
 - 2つのPOWER5
 - L2キャッシュ
 - スイッチ
 - メモリコントローラ
- 95mm×95mmのマルチチップモジュール (MCM) に、4個のチップ
- MCM 2 個=「ブック」=16プロセッサ
- 「ブック」単位で増設可能

アクセス時間異なる

44

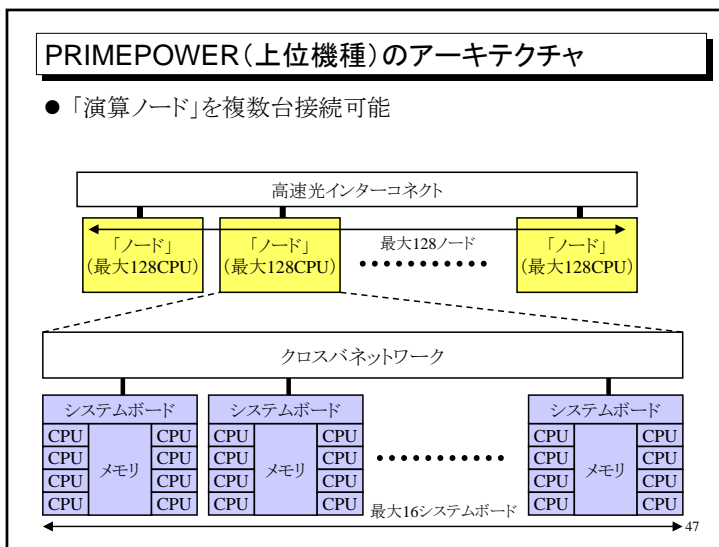


富士通PRIMEPOWER850

- 富士通SPARC64Vプロセッサ (1.3GHz)をクロスバススイッチによって16台接続したUMA
 - OS: Solaris8
 - メモリ: 48GB

この部分がPRIMEPOWER850本体、上部に置かれているPCは、制御用のコンソール。

46



Hewlett-Packard Itanium2 クラスタ

- 演算ノード: Hewlett Packard hpserver rx2600: 16台
 - CPU: Itanium2
 - メモリ: 512MB
 - OS: Red Hat Linux Advanced Workstation 2.1
 - 相互結合網: Myricom社 Myrinet2000 M3F-PCI64B
- 各ノードが独立の計算機としても動作する

48

Hewlett-Packard Itanium2 クラスターの背面

- デスクトップサイズのPC 16台をラックに収めて相互接続するため...
 - 電源ケーブルやネットワークケーブルが川や滝のように流れている



49

こんな形のクラスタもあります(1)

- 普通のデスクトップPCを棚に並べる形



50

こんな形のクラスタもあります(2)

- 普通のデスクトップPCを床の上に並べたもの



51

まとめ

- 並列計算機の構成
 - 共有メモリ型(UMA/NUMA)
 - 分散メモリ型
- 演算ノードと通信路
 - 並列計算機
 - クラスタシステム
 - 広域分散処理システム
- 演算ノードとOS
- 実際の並列計算機・クラスタシステム

52

コンピュータシステムII(2) (訂正と補足)

情報基盤センター
天野 浩文

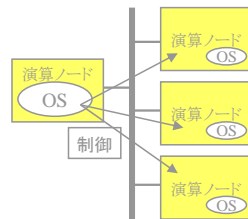
amano@cc.kyushu-u.ac.jp

この科目のwebサイト
<http://isabelle.cc.kyushu-u.ac.jp/~amano/compsys2/>

2.3 演算ノードとオペレーティングシステム (一部訂正)

演算ノードとオペレーティングシステム(1) 訂正前

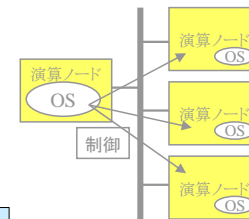
- 1つの演算ノード上で動作するOSが全演算ノードを制御する方式 (並列計算機に向く)
 - システム全体で効率的にソフトウェアを実行することができる
 - 通信路は高速でなければならない (広域分散では使えない)
 - 全体を制御するノードに障害が発生するとシステム全体がダウンする
- ただし、残りの演算ノードでまったくOSが動いていないというわけではない。
 - 特に、分散メモリ型の場合
 - 同種のOSが各ノードで動いているが、ある1つのノード上のOSが主導的な役割を担う、と考えるべき



教科書p22の図では、制御される側のノードにまったくOSがないように見えてしまうが、実際には、上記のように考えるのが妥当であろう。

演算ノードとオペレーティングシステム(1) 訂正後

- 1つの演算ノード上で動作するOSが全演算ノードを制御する方式 (並列計算機に向く)
 - システム全体で効率的にソフトウェアを実行することができる
 - 通信路は高速でなければならない (広域分散では使えない)
 - 全体を制御するノードに障害が発生するとシステム全体がダウンする
- 残りの演算ノードは以下のいずれか
 1. OSを持たない方式 (マスタ・スレーブ方式)
 - 共有メモリ型でのみ可能
 - 上記のノードが「マスタ」、残りのOSなしのノードが「スレーブ」となる。
 - 小規模なシステムでのみ有効
 2. 同種のOSが残りの全ノードでも動くが、上記のノードが主導的な役割を担う方式



教科書p22の図では、制御される側のノードにまったくOSがない。左の分類の2の場合、上の図のようになる。

訂正箇所

実際の並列計算機・クラスタシステム (補足)

九州大学情報基盤センターに設置されている並列計算機やクラスタの一部を紹介する

5

こんな形のクラスタもあります(1)

- 普通のデスクトップPCを棚に並べる形



6

こんな形のクラスタもあります(2)

- 普通のデスクトップPCを床の上に並べたもの



7