コンピュータシステムII（3）

情報基盤センター
天野　浩文

1

来週10月25日の講義

- 休講ではありませんが，下記の講演会の聴講に振り替えます．
  - 「グリッド講演会」
    - 広域分散処理に関する最近の動向に関する講演会です．
  - 日時：10月25日（火）9:00～12:00
    - 1限目の「高年次科目」の講義を受ける人は2限目からの聴講で構いませんが，もし1限目が空いていれば 9:00 からの聴講を強く勧めます．
  - 場所：九州大学国際ホール（中央図書館そば）
  - 参加の事前申込は不要です．
  - 当日の受付での記帳をもって出欠確認に代えます．
  - 講演会題目その他は，このプリントの最後のページを参照してください．

2

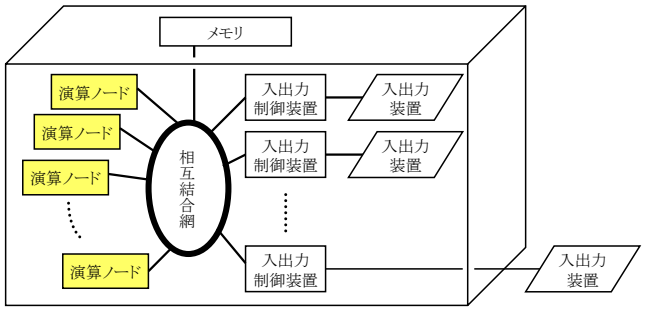
前回のおさらい（1）

- 並列計算機の分類はいろいろあるが
- 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．

3

前回のおさらい（2）

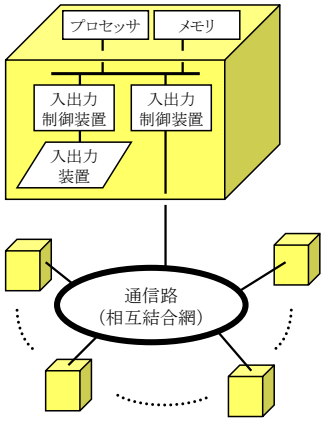
- 並列計算機の構成
  - 多数の演算ノードを高速の通信路（相互結合網）で結ぶ．
  - 全体がひとつの計算機システムとして構築される．
  - 入出力装置は演算ノード間で共有されることもある．



4

前回のおさらい（3）

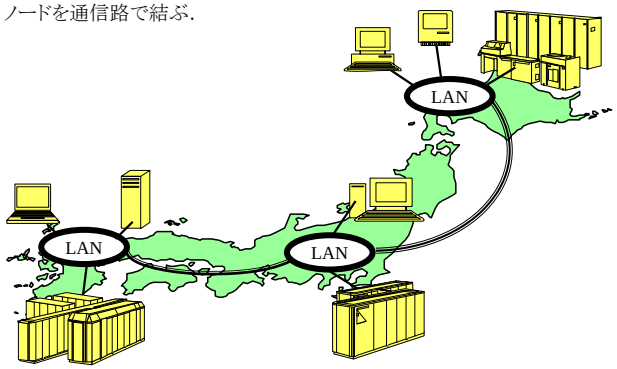
- クラスタシステムの構成
  - 単体で計算機として動作可能な演算ノードを（高速な）通信路で結ぶ．
- 演算ノードになる計算機は小規模なものが多い．
  - PC（パーソナルコンピュータ）
  - WS（ワークステーション）
- 通信路に用いられるハードウェアの例
  - 100Mbpsイーサネット（スイッチ）
  - 1Gbpsイーサネット（スイッチ）
  - その他の高速ネットワーク
    - 例：Myrinet（2Gbps超）



5

前回のおさらい（3）

- 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．



6

プログラム実行制御

教科書とはやや異なる順序で説明する．

- ●プロセスとスレッド
- ●プロセスのスケジューリング
- ●プロセス移動機能
- ●分散共有メモリ

→ 次回以降

- ●キャッシュコヒーレンス → 教科書にはない

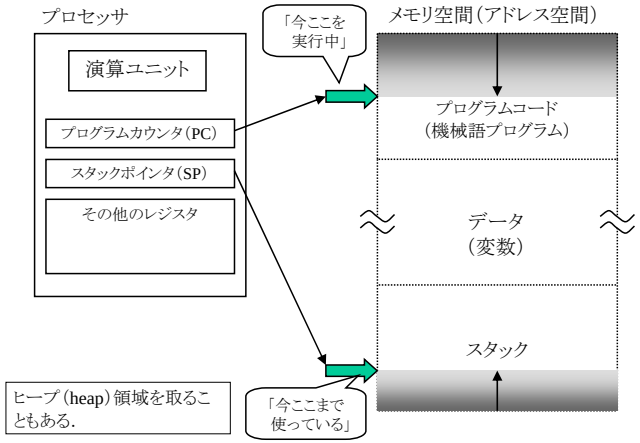
7

プロセス

- プロセス（process）
  - プログラムが計算機上で動作しているときの「実体」
    - すべてのOSに備わっている概念
    - それぞれに，メモリ中に読み込まれた機械語プログラムコード，メモリ上のデータとスタックなどの固有の資源が付随する．
      - 各プロセスのメモリ空間は独立しており，他のプロセスの変数の値を読んだり書いたりすることはできない．
  - OSがプログラムの動作を制御するときの基本単位
    - タスク（task）と呼ぶこともある．
    - 単一のプロセッサの上で，複数のプロセスを切り替えながら実行することもできる．
      ⇒マルチプログラミング，マルチタスキング，タイムシェアリングシステム

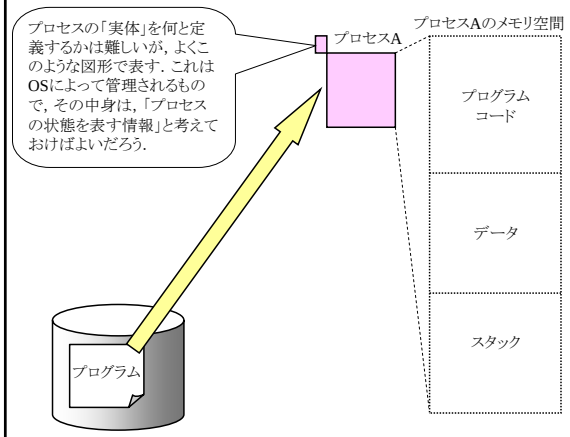
8

プログラムが実行されるときのメモリ空間



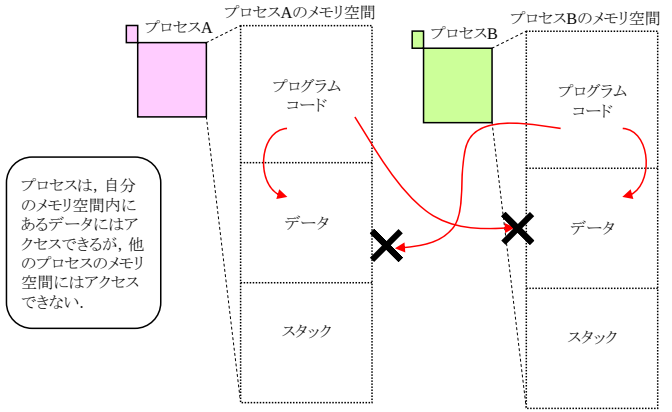
9

プロセスの模式図(1)



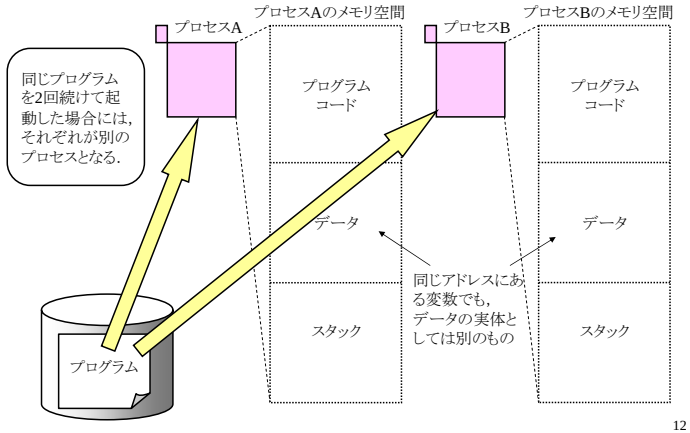
10

プロセスの模式図(2)



11

プロセスの模式図(3)



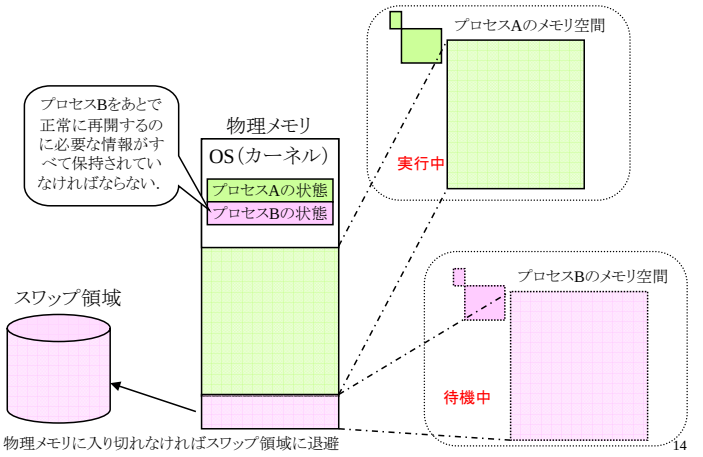
12

複数のプロセスを切り替えるということ

- 各プロセスが中断されたとき，その「状態」をOSが保存しておかなければならない．
  - プロセスAからプロセスBに切り替え，またあとでAに戻る，というような処理をしても，Aの実行がうまく継続できなければならない．
  - 物理メモリの上に載りきれないものは，ディスク上のスワップ領域に退避させる．
    - スワップに退避されていたプロセスが再開されるときには，スワップからメモリに戻さなければならない．
- プロセスの切り替えは結構大変．

13

切り替えながら実行されるプロセス



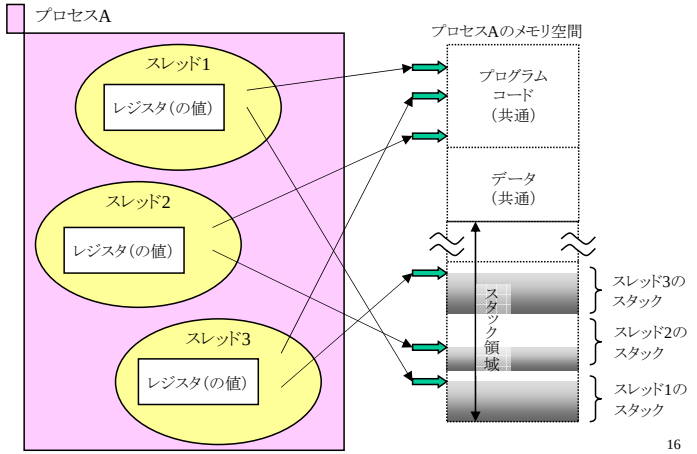
14

スレッド

- スレッド（thread）
  - プロセスよりもさらに小さな単位
    - 1つのプロセスは単一または複数のスレッドからなる．
      - 各スレッドには，固有のプログラムカウンタ，スタックとスタックポインタ，その他のレジスタ（の値）が付随する．
      - 同一のプロセス内のスレッドは，そのプロセスに割り当てられたメモリ空間を共有する．
      - 同一のプロセス内のスレッドは，同じプログラムの中の違う部分を実行できる．
    - OSがプロセッサを割り当てるときの最小単位
      - 単一のプロセスの実行中に，その中のスレッドを切り替えながら実行することもできる．
  - プロセスよりは小さいがプロセスとよく似ている．
    - このため，スレッドのことを軽量プロセス（lightweight process）と呼ぶこともある．

15

プロセスとスレッドの模式図



16

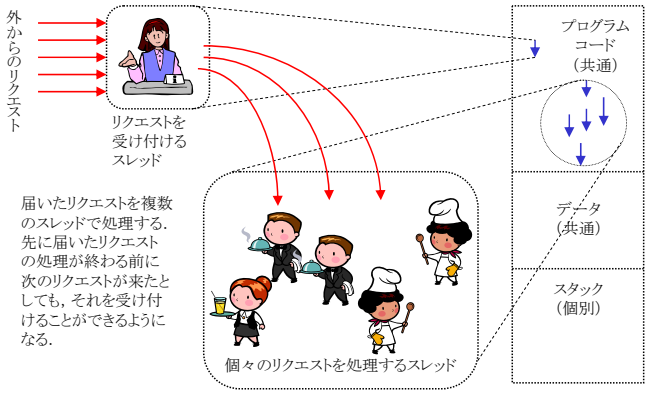
スレッドの例

- スレッドの例
  単一プロセスの内部で:
  - クライアントからのリクエストに応答するスレッドを5個起動して，同時に5つまでのリクエストに対応できるようにする．
  - **i=1** から **1000** までのループを，100ずつ10個のスレッドに分割して実行する．
- ただし:
  - どんなプロセスも複数のスレッドに分割して動作できるわけではない．
    - 同一プロセス内のスレッドが相互に悪影響を及ぼさないように注意深くプログラミングしなければならない．

17

スレッドの利用例（1）

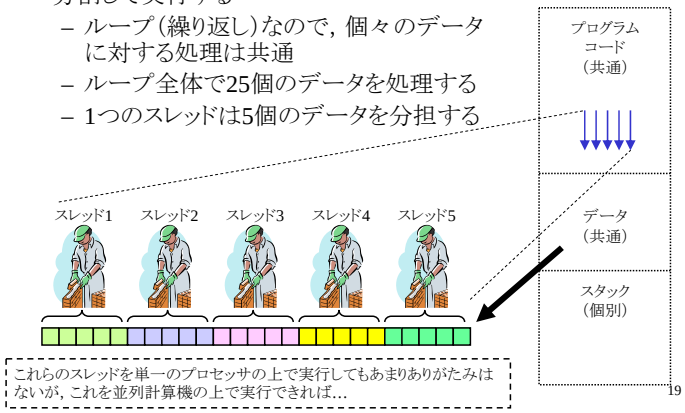
- Webサーバを複数のスレッドで構成する



18

スレッドの利用例（2）

- **i=1** から **25** までのループを **5** のスレッドに分割して実行する
  - ループ（繰り返し）なので，個々のデータに対する処理は共通
  - ループ全体で25個のデータを処理する
  - 1つのスレッドは5個のデータを分担する



19

プロセスとスレッドの比較（1）

- 「大きさ」の比較
  - プロセスは，プログラムが動作しているときの実体
  - スレッドはプロセスよりもさらに小さく，単一のプロセスが複数のスレッドから構成されることがある．
- メモリ空間の比較
  - 各プロセスは固有のメモリ空間を持つ．
    - 異なるプロセスがメモリアクセスによって相互に影響を及ぼすことはない（相互の保護可能）．
    - スレッドに比べ，実行するプロセスを切り替えるのに時間がかかる．
  - 同一プロセス内のスレッドは，そのプロセスのメモリ空間を共有する．個別に持つのはレジスタとスタックのみ．
    - 実行するスレッドを切り替える時間が短くてすむ．
    - 共有されるデータへ不適切なアクセスを行うと，相互に悪影響を及ぼすことがある（「相互の保護ができない」）

20

プロセスとスレッドの比較（2）

- 教科書の p.39 の議論へのコメント
  - プロセスとスレッドのファイル操作
    - 教科書では「同一のプロセス内のスレッド間では，ファイル操作は区別なく制御される」と説明しているが…
    - これは，オープンされているファイルの情報（メモリ空間の中に記録されている）が共有されるか，共有されないかによる．
      - 同一プロセス内のスレッドであっても，どのファイルにアクセスするか，または，ファイル内の何バイト目を操作しているかをスレッド間で区別するようにプログラミングすれば，区別して制御することも可能

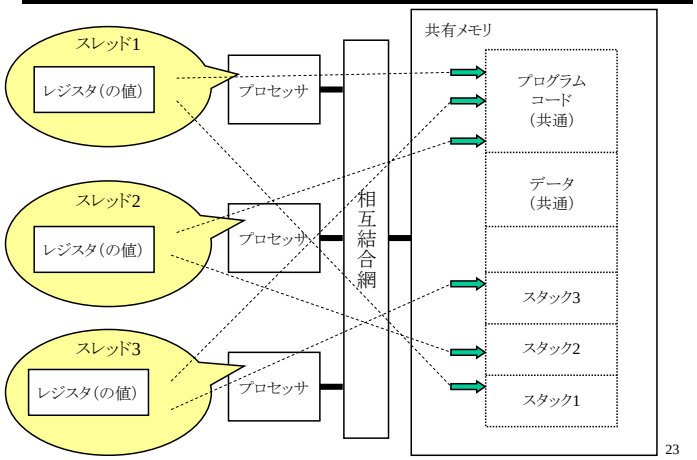
21

プロセスとスレッドの比較（3）

- 並列処理・分散処理との親和性
  - 共有メモリ型並列計算機では
    - **同一プロセス内の複数のスレッドを別々のプロセッサ上で同時に実行可能**
    - 各プロセッサに割り当てる処理の単位をスレッドにすると効率的に実行できることがある．
  - 分散メモリ型並列計算機・クラスタシステム・広域分散システムには共有メモリがない．
    - 単一のプロセスを複数のスレッドに分割しても，それらのスレッドを別のプロセッサ上で動作させることは難しい．
    - このようなシステムでは，**各プロセッサに割り当てる処理の単位をプロセスとせざるを得ない**．

22

共有メモリ型並列計算機とスレッドの親和性



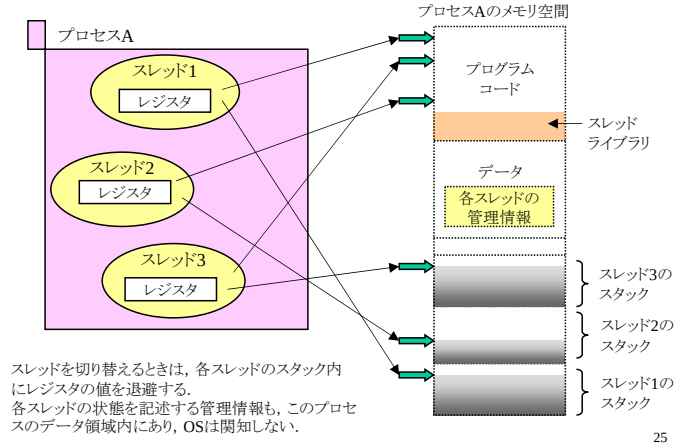
23

スレッドの実現法

- ユーザ空間で実装する方式
  - スレッド固有のデータ(レジスタの値を待避する領域，および，スタック)と，スレッドの操作(生成・切り替え・終了)に必要なライブラリを，ユーザプロセスのメモリ空間に配置する．
  - ユーザプロセスの実行中に，スレッドライブラリコールによって自分で勝手にスレッドを切り替える．
    - OS(カーネル)はスレッド操作にほとんど関与しないので，スレッド機能のないOSでも，少し変更すれば利用可能．
    - スレッドの実行中に，OSによってプロセス切り替えが発生することがある．
- 一部はカーネル空間を使って実装する方式
  - 各スレッドのスタックはユーザ空間内に，レジスタの値を退避する領域はカーネル空間内に，それぞれ配置する．
    - スレッド操作に必要な機能はカーネルに実装しておく．
  - スレッドがシステムコールによりプロセッサの利用権を放棄する(カーネルに返す)か，カーネルがスレッドからプロセッサの利用権を(一時的に)奪うと，スレッド切り替えが発生する．

24

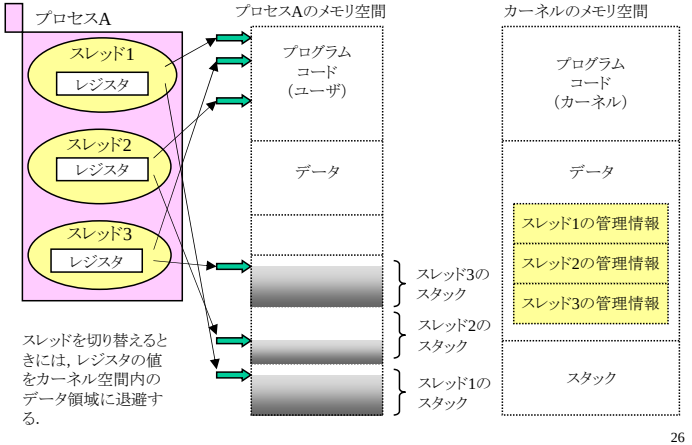
ユーザ空間のスレッドの模式図



スレッドを切り替えるときは，各スレッドのスタック内にレジスタの値を退避する．
各スレッドの状態を記述する管理情報も，このプロセスのデータ領域内にあり，OSは関知しない．

25

カーネル空間を使ったスレッドの模式図



スレッドを切り替えるときには，レジスタの値をカーネル空間内のデータ領域に退避する．

26

まとめ

- **プロセスとスレッド**
  - プロセスは，プログラムが動作しているときの実体
  - スレッドはプロセスよりもさらに小さく，単一のプロセスが複数のスレッドから構成されることがある．
- 同一のプロセスに含まれるスレッドはメモリ空間を共有する．
  - プロセスを切り替えるのに比べ，スレッドを切り替えるのにかかる時間は短い．
  - ただし，相互に悪影響を及ぼす可能性がある．
- メモリ空間を共有するスレッドは，共有メモリ型並列計算機との親和性がよい．

27

グリッドコンピューティング入門

来週の講演会に向けての予告編

28

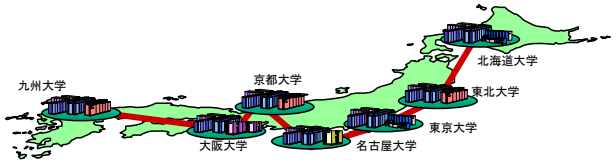
グリッドコンピューティング（1）

- 電化製品とPCの比較
  - 電化製品を使うとき
    - 国内各地の発電所が電力網につながっている.
    - 利用者は，それらが国内のどこにあるか意識せずに，電源プラグを壁のコンセントにつなぐだけで電力を使える.
  - PCでインターネットにアクセスするとき
    - 世界各地のサーバがインターネットにつながっている.
    - ユーザは，サーバが世界のどこにあるか意識せずに，自分のPCをLANにつなぐだけでサービスを受けられる

29

グリッドコンピューティング（2）

- 電力網（Electric Power Grid）
  - 各地の発電所で生産された電力が電力網の上を流通している
- 計算グリッド（Computational Grid）
  - 『各地の計算サーバが提供する計算パワーがインターネット上を流通している』
    と見なすこともできる



30

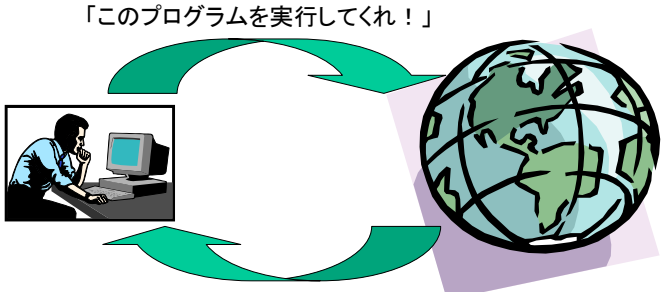
グリッドコンピューティング（3）

- 計算だけでなく，各種のサービスが流通するコンピュータネットワーク
  ⇒「**グリッド**」と呼ぶことにする
- そのような環境でサービスを利用する・提供することを「**グリッドコンピューティング**」と呼ぶ

- もう少し堅苦しい表現をするなら・・・
  「インターネット上に広域分散する計算機資源を統合運用することにより，それらの資源全体の効率的な運用や，単一の資源では達成できないような大規模計算サービス等を目指すこと」

31

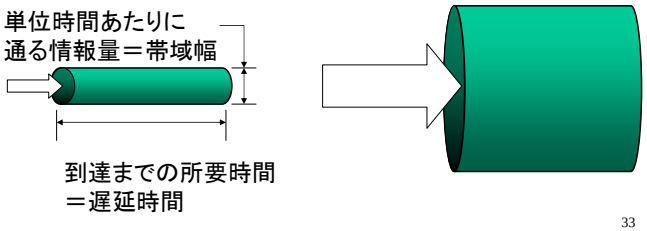
グリッドの理想的な姿



32

グリッド実現に必要なもの（1）

- 高速のネットワーク
  - 急速に整備が進みつつある
    例：SuperSINET (10Gbps)
  - 遅延時間についてはあまり短縮できないが，帯域幅については大変な勢いで改善されつつある



33

グリッド実現に必要なもの（2）

- ネットワークに接続された計算機資源
  - これもまぁ，「はいて捨てるほどある」と言えなくもない・・・
  - 各家庭のPCのような小さな計算機資源を多数集めることによってグリッドを構築することを目指す向きもあるが・・・
    - これは，ごく限られた種類の問題についてではあるが，すでにある程度実用化されている．
      例：setiathome
      `http://setiathome.ssl.berkeley.edu/`

34

グリッド実現に必要なもの（3）

- アーキテクチャ，OS，性能・規模，運用方針（利用資格・課金・利用制限値，など）が大きく異なる計算機群を統合運用する技術
  - 複数の計算機の能力を合わせて巨大な仮想スーパーコンピュータを実現する技術
  - 多数の計算機の中から空いているものを見つけて効率的に活用する技術
  - 多数の計算機を多数の利用者が安全に使えることを保証する技術
- これらの部分については，まだ課題が山積み！

35

グリッドの現状（1）

- 必要とされる機能を提供する各種ミドルウェアの開発が進行中
  - Globus（米）
  - Unicore（欧）
  - Ninf（日）
  - STA（日）
  - ITBL（日）
  - NAREGIプロジェクト（日）
- 来週の講演会は
  - NAREGIプロジェクトの主要メンバから2名の講師が参加

36

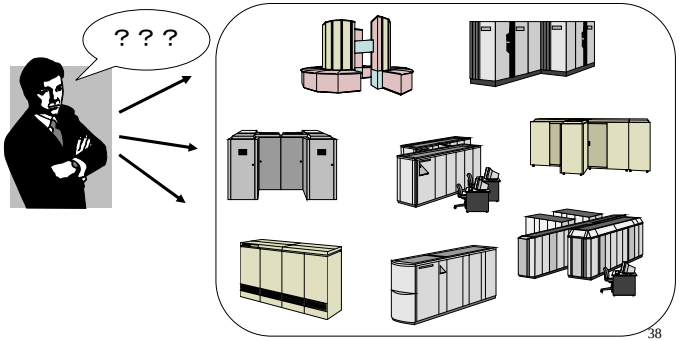
グリッドの現状（2）

- ミドルウェアは続々と作られているものの・・・
  - 大規模計算機資源が多数統合された実用レベルのグリッドはまだまだ遠い夢.
    - 多数のPCに「宿題」をばらまいて「答案」を回収するだけの自由参加型（setiathome型）の応用はある.
    - おもちゃのような例題を解くことはできても，実用規模の問題を解けるような大規模なグリッドの実現はこれから
  - 利用資格の制限や利用代金の負担など，技術的な問題以外の障害を乗り越えるためには，具体的な成功例を見せて多くの利害関係者を説得する必要あり.

37

グリッドの課題（1）

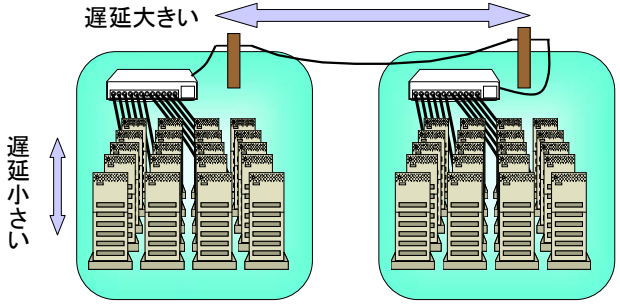
- アーキテクチャ・OS・性能・規模・運用方針の差異を意識せずに利用できるためには・・・



38

グリッドの課題（2）

- 通信遅延の大きく異なる環境で，本当に複数の計算機が力を合わせるためには・・・



39

グリッドの課題（3）

- その他の課題（まだまだあるかも）
  - 多数の利用者の登録情報の効率的管理法，一括認証方式
  - 計算リクエスト・データ・実行結果をインターネット上で安全に配送するための暗号方式
  - インターネット上に分散する計算機資源の稼働状況・負荷状況の実時間把握
  - 多数のジョブを広域分散環境下で実行するときのスケジューリング方式

40