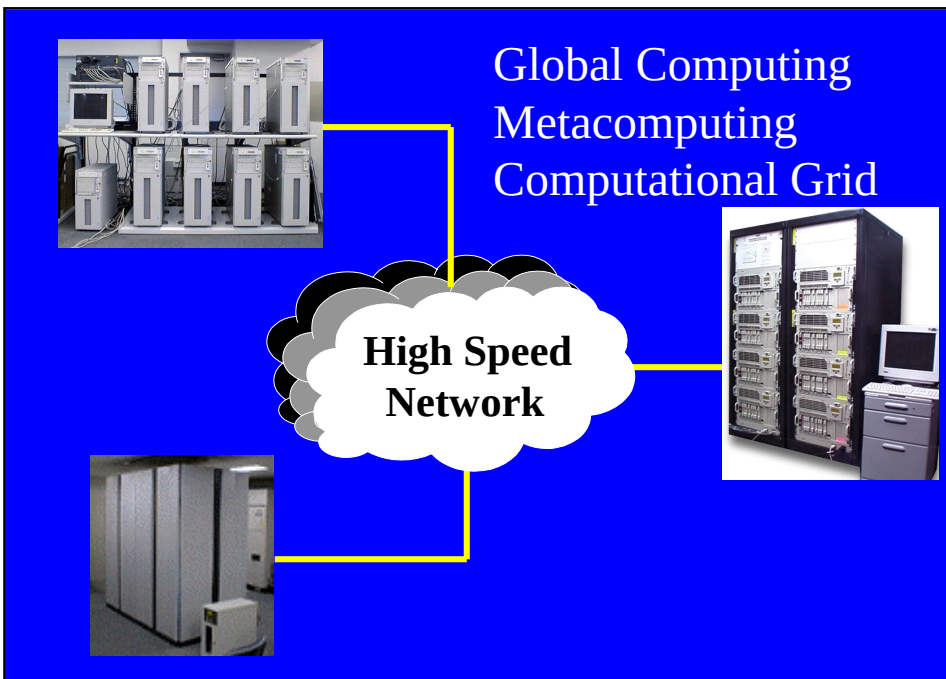
Globusを用いたグローバル コンピューティング環境の構築とその評価

田中良夫，佐藤三久　　平野基孝
新情報処理開発機構　　（株）SRA
中田秀基，関口智嗣
電子技術総合研究所



グローバルコンピューティング（背景）

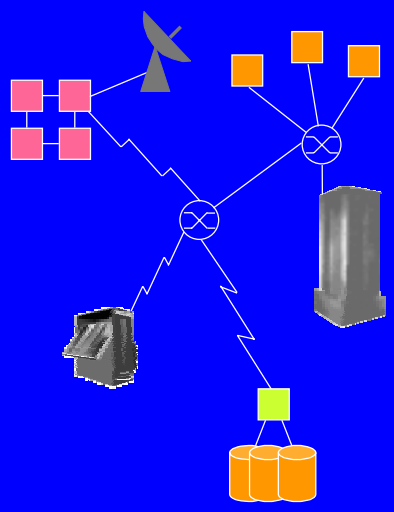
- ネットワークの高度利用
  - インターネットの高速化 kbps → ～100Mbps
  - キャンパスネットの高速化 10Mbps → Gbps
  - 一般家庭，企業への普及
- 仮想的情報資源の共有
  - 計算機資源そのものへの要求
    - 超高速計算機（スーパーコン，超並列機，）
    - 超巨大データベース
    - 高精度ライブラリの利用
    - 観測データへの遠隔アクセス

グローバルコンピューティング（目的）

- ネットワークで接続された世界中/国内/キャンパス内/ラボ内の計算資源にアクセスすることで
  - あたかも1台の超巨大（並列）計算機として
  - 超高速計算機の短期時間借用手段として
  - データベースへの高機能アクセス提供手段として
  - collaboration system として
- （原理的に）すべての許可されたシステムにアクセス
  - 多種多様な計算機・実験システムの仮想複合体
  - デスクトップからシームレスなアクセス

グローバルコンピューティング（特徴）

- 適用可能な問題に期待される共通的性質
  - Ubiquitous -- どこからでも，ネットワーク透過にアクセスできること
  - Resource Aware -- 異機種環境に対応できること
  - Adaptive -- 動的に変化するネットワーク，計算機資源環境に対応して最大性能を得られること

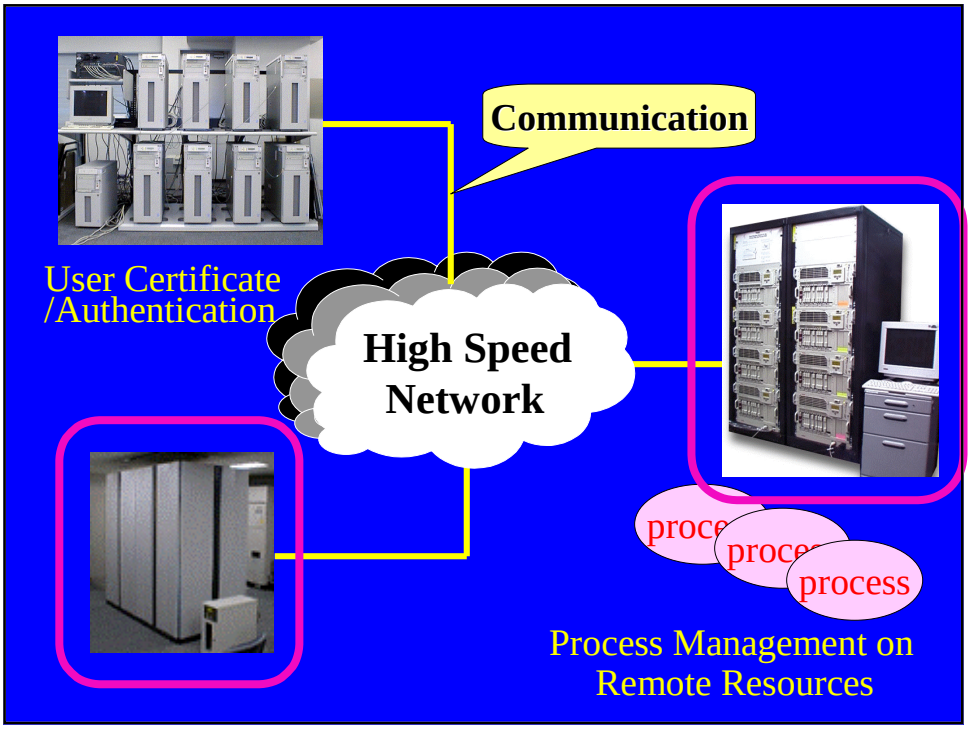


関連研究

- Albatross Project@vrije university, Netherland
  - understand application behavior (programmability and performance) on wide-area cluster system.
- PACX-MPI@hlrs, Germany
  - MPI extention for distributed computing
- Globus Project
  - provides toolkits for building software infrastructure for global computing environment
  - Large testbeds (I-WAY, GUSTO)
- Legion, Netsolve, Ninf…..

The Globus Project

- Basic research in grid-related technologies
  - Resource management, QoS, networking, storage, security, adaptation, policy, etc.
- Development of Globus toolkit
  - Core services for grid-enabled tools & applns
- Construction of large grid testbed: GUSTO
  - Largest grid testbed in terms of sites & apps
- Application experiments
  - Tele-immersion, distributed computing, etc.



目的

- Firewallを越えて複数の並列システム(クラスタ、MPP)を利用するためためのの枠組みを提供
  - 新しい型のGRAMであるRMF(Resource Manager beyond the Firewall)を設計、実装

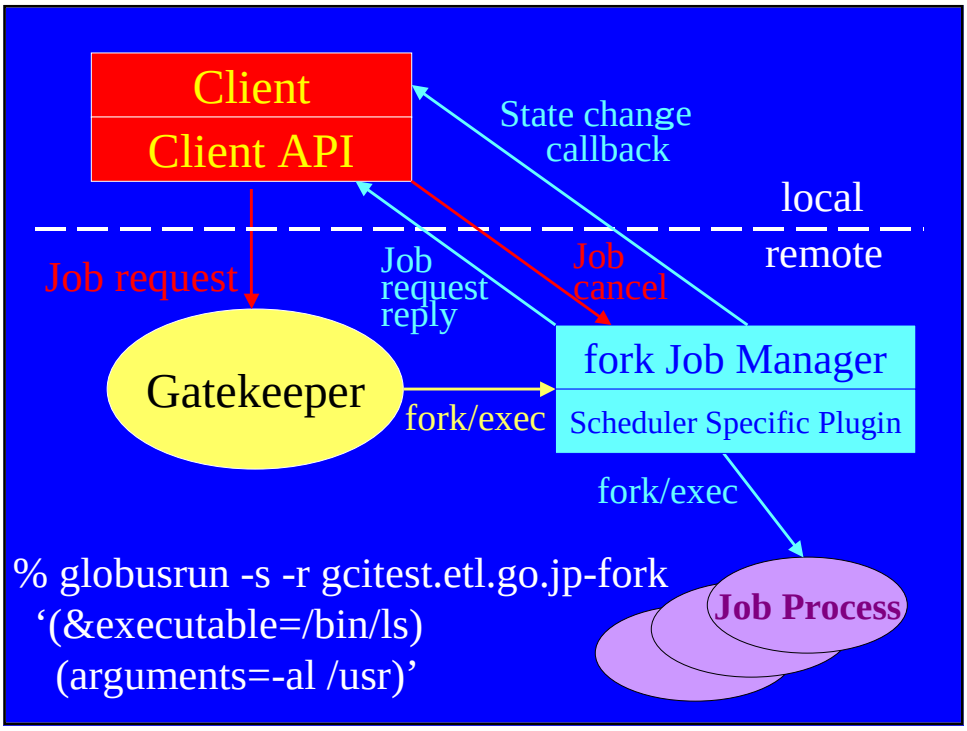
発表概要

- Globusの概要 (ver. 1.0)
- RMFの設計と実装
- 簡単な性能評価
- Summary and Future Work

Core Globus Service

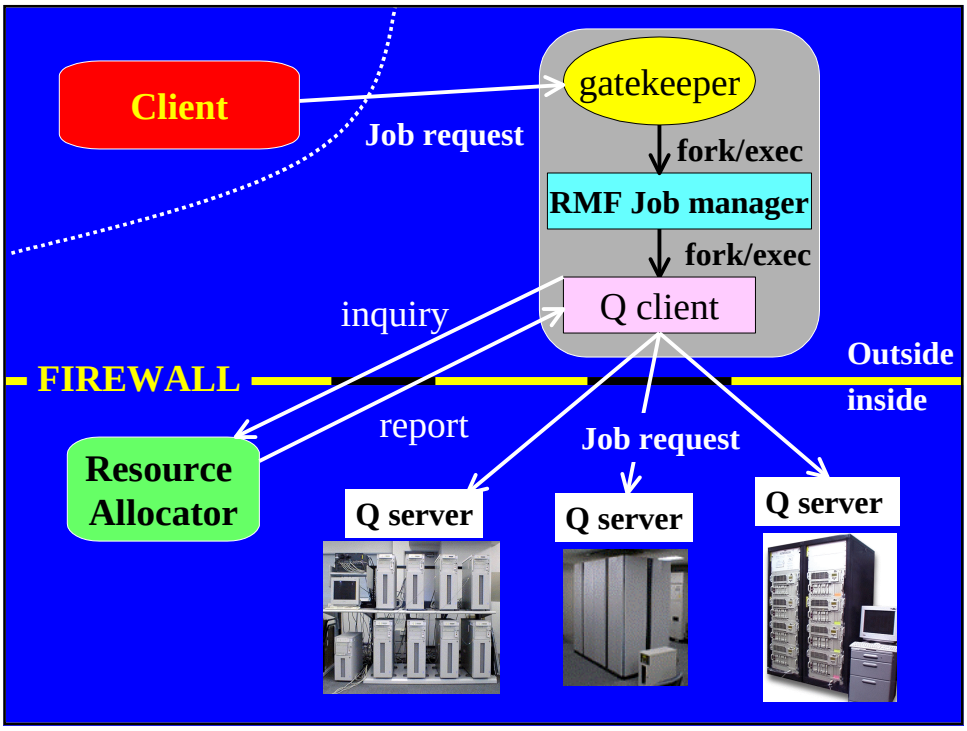
- Resource Management (Globus Resource Allocation Manager, GRAM)
- Communication (Nexus)
- Information (Metacomputing Directory Service, MDS)
- Security(Globus Security Infrastructure, GSI)
- Health and Status (Heat Beat Monitor, HBM)
- Remote Data Access (Globus Access to Secondary Storage, GASS)
- Executable Management (Globus Executable Management, GEM)

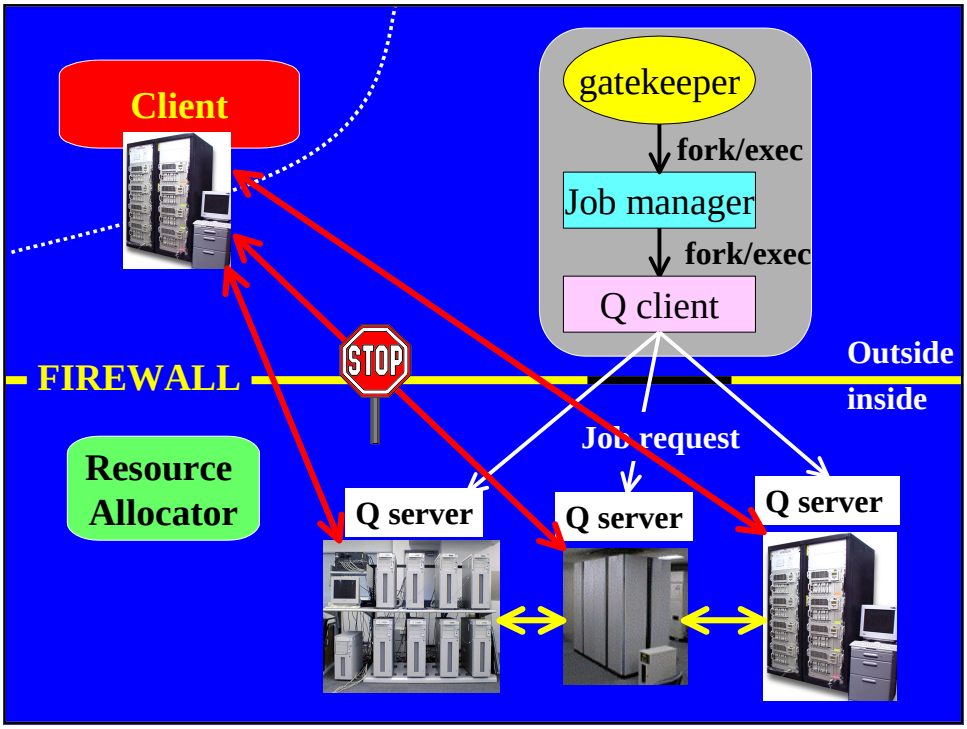
API + User Command



RMF: Resource Manager beyond the Firewall

- 新しい型のGRAM
- クラスタやMPPを管理
- Firewallを越えて計算資源の利用を可能とする
- 基本モジュール:
  - Q-system (クライアント/サーバモデルに基づく遠隔ジョブキューイングシステム)
  - Resource Allocator

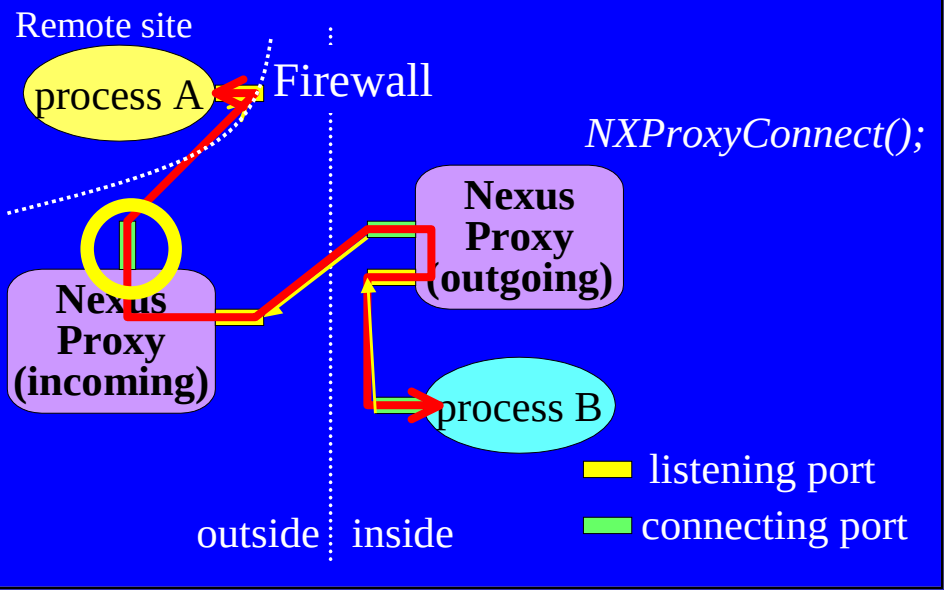

Client
Job request
gatekeeper
fork/exec
RMF Job manager
fork/exec
Q client
inquiry
report
FIREWALL
Outside
inside
Resource Allocator
Job request
Q server
Q server
Q server


Client
gatekeeper
fork/exec
Job manager
fork/exec
Q client
STOP
FIREWALL
Outside
inside
Job request
Resource Allocator
Q server
Q server
Q server

Nexus Proxy

- TCP 通信をリレー
- Nexus Proxy incoming server はデーモンプロセスとしてfirewallの外側で稼動
- Nexus Proxy outgoing server はデーモンプロセスとしてfirewallの内側で稼動
- *NXProxyBind()*, *NXProxyConnect()*, *NXProxyAccept()*などのライブラリ関数を提供

Nexus Proxy (inside → outside)



Globusソースコードの修正

- 通信モジュールのいくつかの関数(約10万行のソース中、10個程度)を修正
- NXPROXY_SERVER環境変数がセットされていれば、Nexus Proxyを介した通信を行う。そうでなければ通常どおりの通信を行う。

広域並列システム上での実験

- これまでの実験例は主に分散計算
  - 大規模分散シミュレーション
  - Virtual Realityソフトウェア
- 広域並列システムに適したアプリケーション
  - **各プロセッサが非同期に計算を進められる**
  - **データの独立性が高い**
- Knapsack問題
  - **分枝限定法による0-1 Knapsack問題の並列解法**
  - **これまでに算出された下界値の最大のもの(Global Low)を用いて枝刈りを行なう**

MPICH-G

- Globus deviceによるMPICHの実装
  - 利用する計算機の情報をMDSにより獲得
  - ユーザ認証にGSIを利用
  - プロセス生成にGRAMを利用
  - 通信にはNexusを利用
  - ファイルアクセスにはGASSを使う
  - アプリケーションの監視、停止処理などにGlobusのプロセス監視機構を利用

Machineファイルの記述例

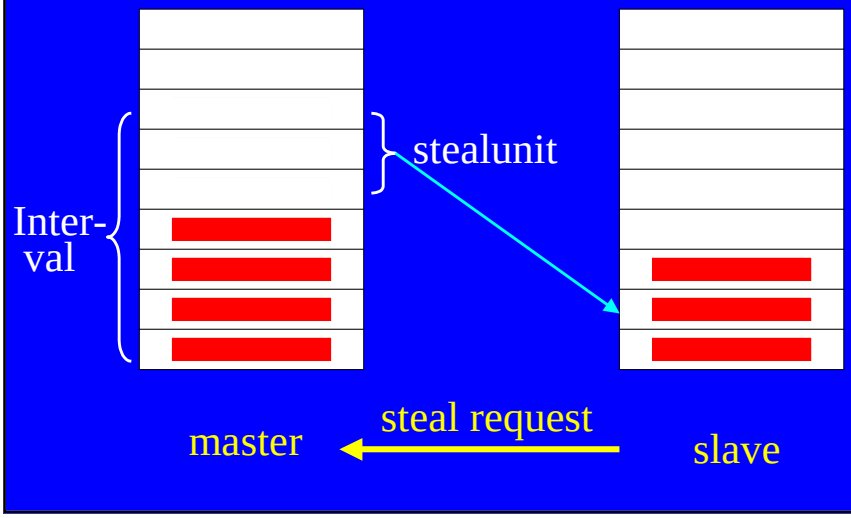
```
%cat machines
donner.mcs.anl.gov-fork 8
bonny.isi.edu-fork 8
moti4.ncsa.uiuc.edu-lsf 64

% mpirun -np 80 -machinefile machines a.out
```

並列化のポイント

- 枝刈りにより木構造が一様にならないことが予想されるため、動的にジョブを分散
- 負荷分散のためのオーバーヘッド（通信回数、通信量）をなるべく抑える
- Global Lowの通知方法を工夫する

実装方法

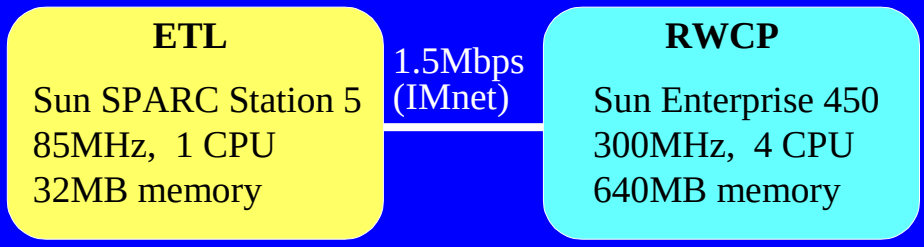


実装の特徴

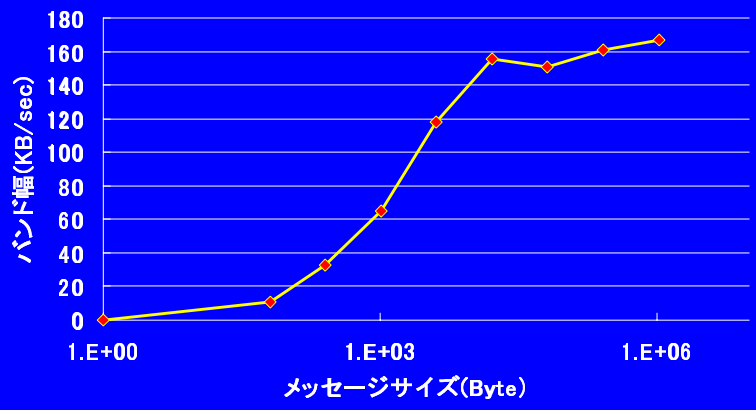
- スレーブは自分のスタックが空になった時にのみマスタのジョブを盗みに行く。
  負荷分散のためのオーバーヘッドが少なくなり、負荷が均一になりやすい
- Global Lowを更新するための通信はジョブのやりとりに含めてしまう

実験環境

ETLのSS5とRWCPのEnterprise450



Point-to-pointバンド幅



レイテンシ：約 5 msec

実験結果（基礎データ）

PCクラスタ上での並列化効率
(interval = 1000, stealunit = 10)

ノード数	1	2	4	6	8
PSC6	1.00	0.93	0.90	0.90	0.89
PSC7	1.00	1.41	1.26	1.14	1.00

逐次実行時間

	SS5	Enterprise
PSC6	333.4秒	70.8秒
PSC7	125.7秒	26.0秒

実験方法

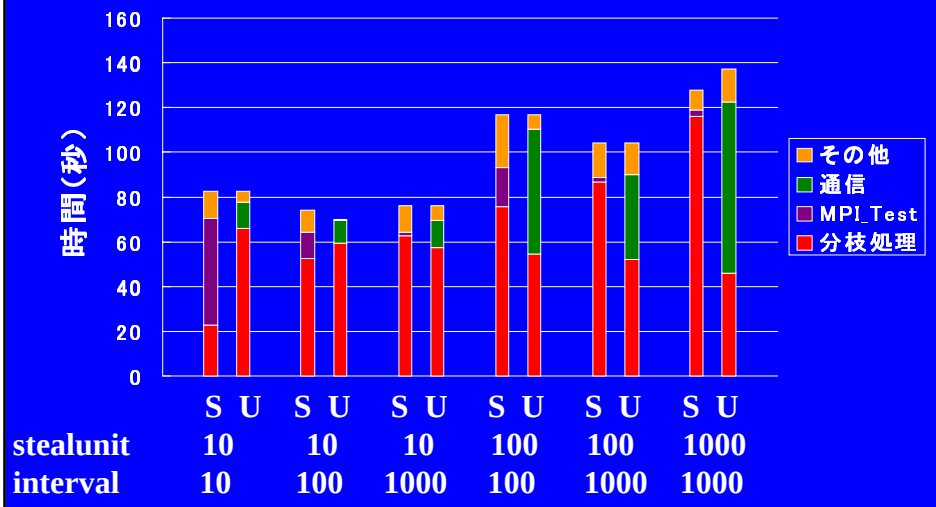
- ETL側の計算機をマスタ、RWC側の計算機をスレーブとする
- intervalとstealunitを変化させる
  - **intervalが小・・・マスタのオーバーヘッド大**
  - **intervalが大・・・スレーブがアイドル状態になりやすい**
  - **stealunitが小・・・ジョブ要求が頻繁に発生する可能性**
  - **stealunitが大・・・転送オーバヘッド大、負荷の不均一**

実験結果（実行時間）

stealunit	PC6			PSC7		
	interval			interval		
	10	100	1000	10	100	1000
10	82.6	73.9	76.0	51.9	42.9	39.5
100	×	116.7	104.1	×	209.9	177.5
1000	×	×	137.0	×	×	160.9
1 node	333.4			125.7		

単位：秒

ブレークダウン(PSC6)



まとめ（その1）

- 新しい型のResource ManagerであるRMF (Resource Manager beyond the Firewall)の設計と実装
  - Q-System
  - Resource Allocator
  - Nexus Proxy
- Firewallを越えてクラスタやMPPなどの並列資源を利用することができる。

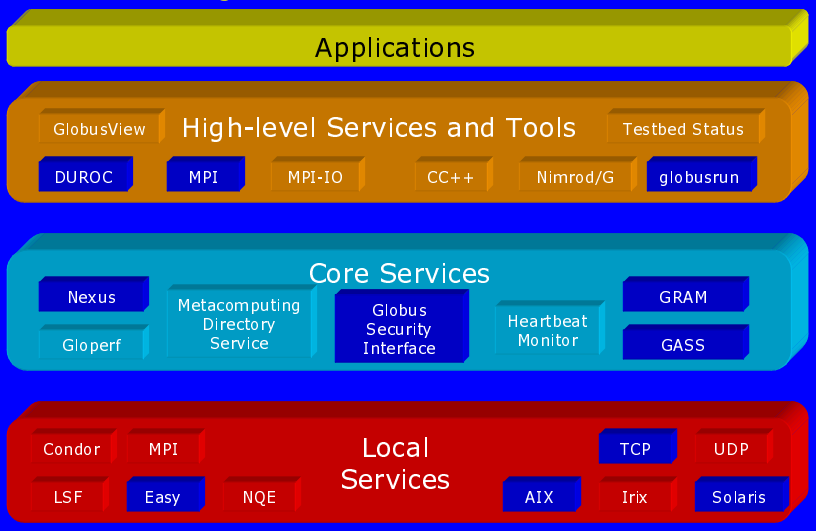
まとめ（その2）

- 2台のワークステーションを使った広域並列システム上で並列ﾌﾟﾛｸﾞﾗﾑを実行
  - **動的負荷分散を行う**
  - **負荷分散のオーバーヘッドを抑える**
  - **通信量を抑える**
  - **通信と計算をオーバーラップさせる**
- 探索問題のような問題は広域システム上でも十分高い効率を得られそう

Future Work

- セキュリティのチェック
- スケジューリング（アロケーション）ポリシーの研究
- グローバルコンピューティング 環境に適したアプリケーションの探索
- ワイドエリアクラスタシステムの性能評価
- インフラ整備、拡大

Layered Architecture



Globus Resource Allocation Manager (GRAM)

- 遠隔資源の割り当て、管理 (Ex: 遠隔資源上でのプロセス生成)
- ジョブ送信や管理のためのAPIを提供
- いくつかの型のGRAMが存在:
  - fork type

    Globusシステムの管理の負担が大きくなる
  - LSF type

    クラスタシステムは1台の仮想的な並列計算機ではなく、多数のワークステーション、PCとみなされる
    Firewallを越えて計算資源を利用することができない

Resource Allocator

- 起動時に設定ファイルを読み込む
- 設定ファイルの例:

#Name	type	procs	nnodes	clock	prefix
COMPaS	c	4	8	200	pdpsmp
COMPaS-II	c	4	8	450	pdsmpc
SR2201	p	1	256	150	

- 資源選定の基準:
  - 要求されたプロセッサ数
  - 計算資源の
    - プロセッサ数、走っているジョブの数、プロセッサクロック

資源上でのプロセス生成 (jobtype != MPI の場合)

- ユーザは1台の並列システム上でのジョブの実行を意図する



- 選ばれた資源のマスターノード上で、その資源固有の mpirun コマンドによってジョブプロセスが生成される。

資源上でのプロセス生成 (jobtype == MPI の場合)

- ユーザは複数の並列システム上でジョブの実行を意図。ジョブは MPICH-G で記述されたプログラムであることが前提。

- 選ばれたすべての資源上で、execv() システムコールによってジョブプロセスが生成される。