

# グリッドプログラミング環境(2)

建部修見

筑波大学システム情報工学研究科

コンピュータサイエンス専攻

# 概要

## ● グリッド技術とは？

- ▶ 計算グリッド
- ▶ データグリッド
- ▶ アクセスグリッド

## ● グリッド技術とその要素技術

- ▶ 単一認証技術
- ▶ 情報サービス
- ▶ ファイル管理
- ▶ 高速広域データ転送技術
- ▶ 資源管理

## ● オープングリッドフォーラム (OGF)

## ● グリッド基本ソフトウェア

- ▶ Globus

## ● グリッドプログラミング

- ▶ コンポーネントモデル
- ▶ MPI
- ▶ GridRPC
- ▶ Grid Datafarm

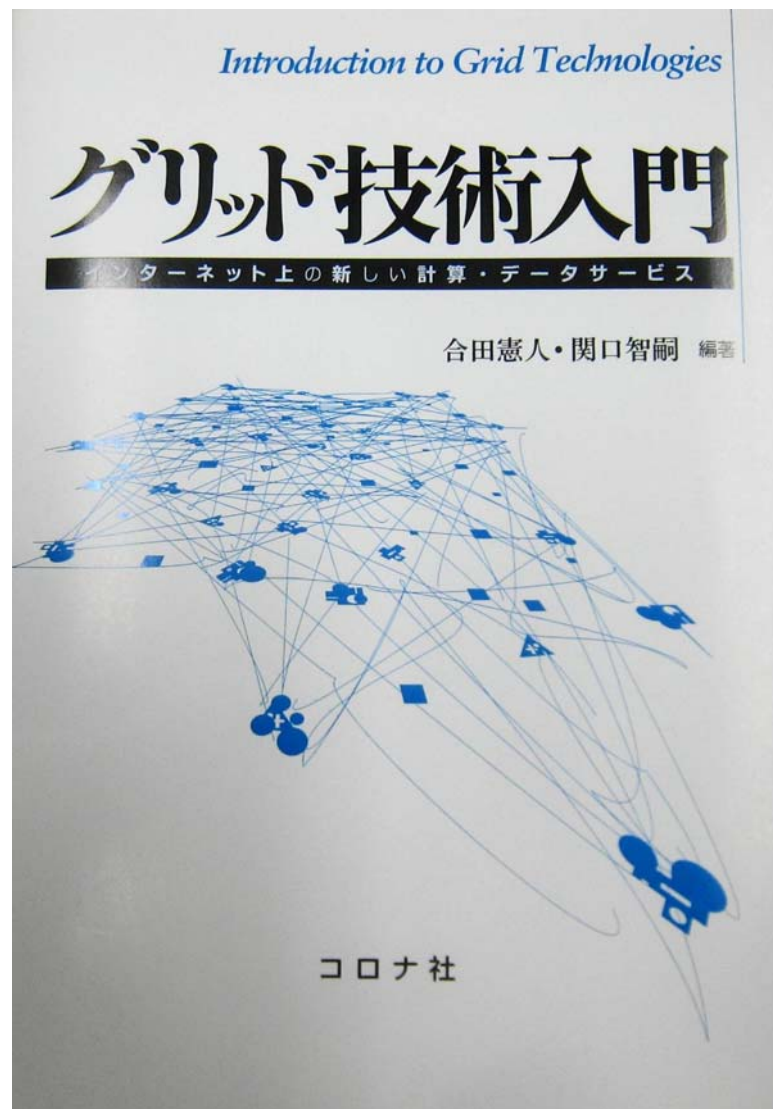
# グリッドの本

## ● グリッド技術入門

▶ インターネット上の新しい  
計算・データサービス

● 合田憲人, 関口智嗣編著

● コロナ社



# グリッドの要素技術

アプリケーション

アプリケーション実行支援

プログラミング  
モデル

情報サービス

データベース管理

スケジューリング

データ管理

ジョブ実行管理

セキュリティ

インフラ(ネットワーク, 計算機, 実験装置, 他)

# グリッドにおけるデータ管理

## ● ネットワーク上のストレージにアクセス

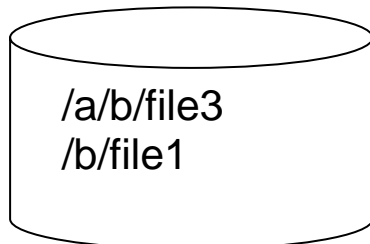
- ▶ サーバの名前
- ▶ プロトコル
- ▶ サーバ内のパス名

## ● 何らかの理由で、ファイルが移動することも

## ● ファイル複製を作成していたつもりが、一貫性がなくなることも

## ● 巨大ファイルに対する高速なアクセスも

gsiftp://tsukuba.ac.jp/



http://u-tokyo.ac.jp/



ftp://mext.go.jp/



# ファイル管理の役割(1)

- グリッド上で必要なファイル, 検索したいデータに対し, 簡単, 高速なアクセス, 安定したアクセスを提供すること
- 「簡単」
  - ▶ ファイルのパス名, 検索式, 検索条件などを指定し, 欲しいデータへのアクセスを可能にする
    - ◎ サーバ名, プロトコルを指定しない
  - ▶ ファイルのパス名, 検索式などから所在, プロトコルに変換する機構が必要
    - ◎ パス名の場合, ディレクトリ管理サービス
    - ◎ 検索式の場合, メタデータ管理サービス
  - ▶ 間接的な管理により, サーバ, プロトコルの動的決定が可能となり, **柔軟性**が向上する
    - ◎ ファイルの移動, 利用可能なファイル複製の選択など

# ファイル管理の役割(2)

## ●「高速」なアクセス

### ▶ アクセス帯域を大きく

- ◎ 高速転送技術

- ◎ 必要数の複製を適切な場所に配置し、アクセスを分散

### ▶ アクセス遅延を短く

- ◎ ネットワーク的に近い複製を選択

- ◎ よく利用される場所の近くに複製を作成

- ◎ よく利用されるデータはアクセス集中を避けるため、複製を作成

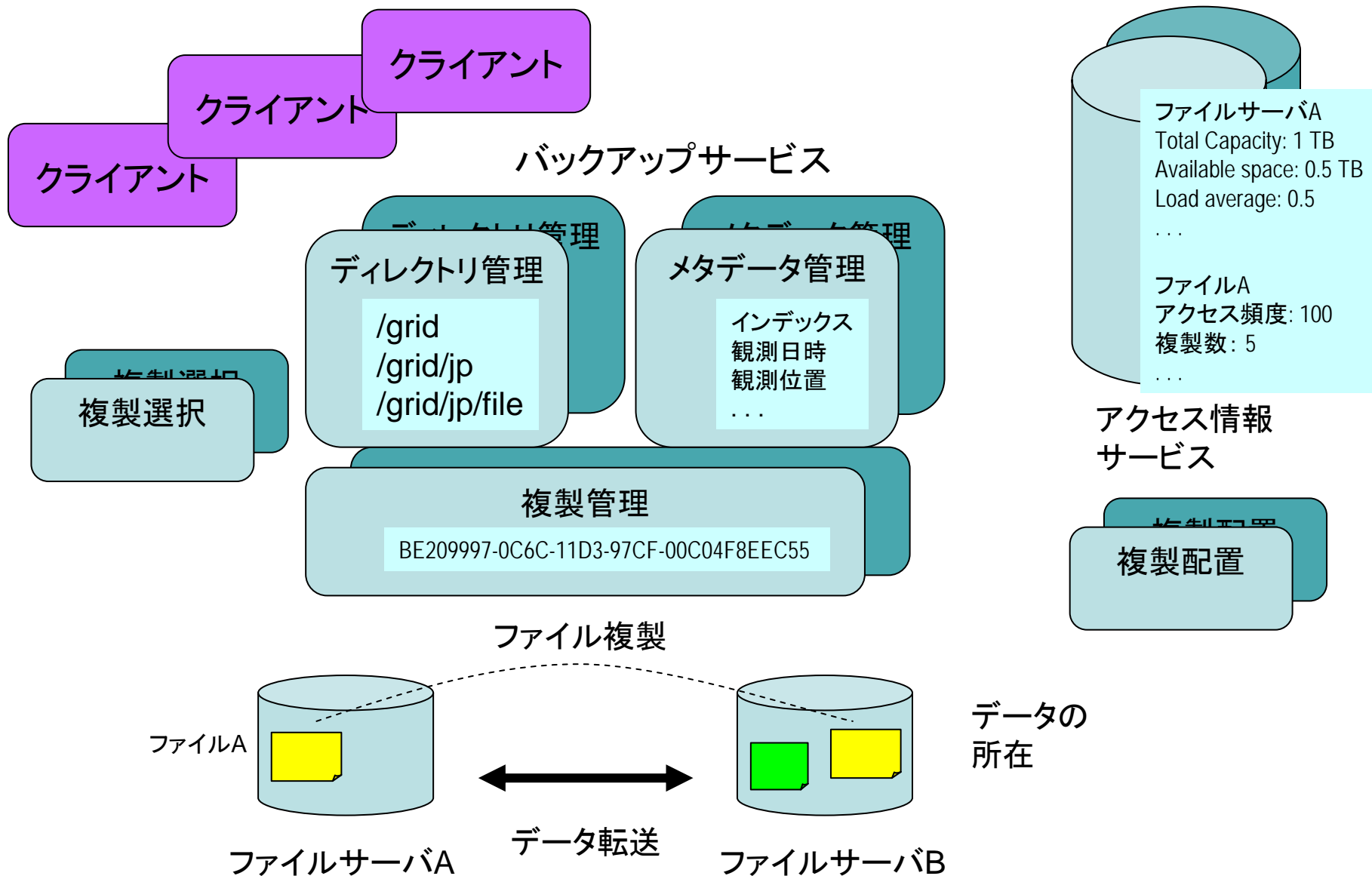
# ファイル管理の役割(3)

## ●「安定」して提供する

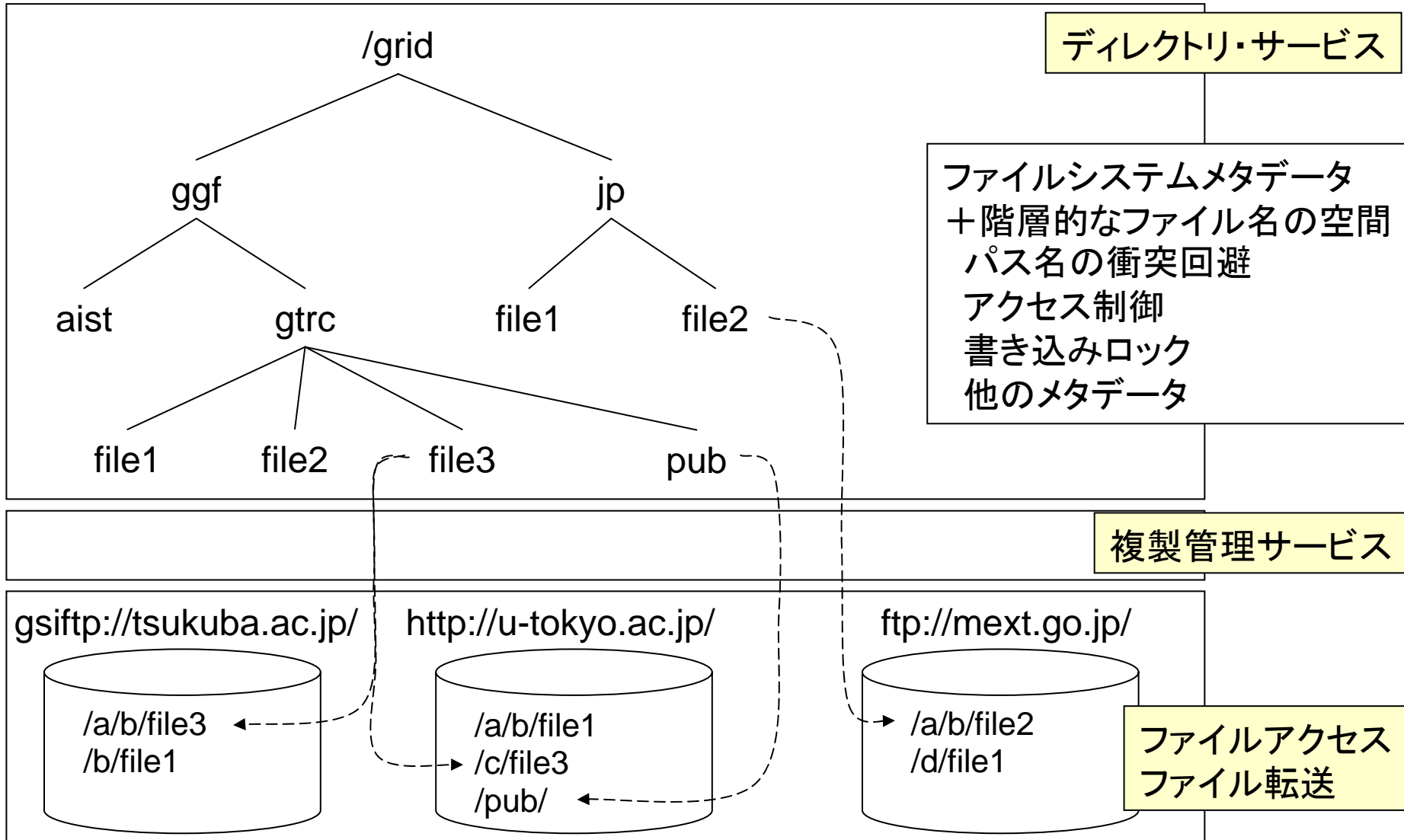
- ▶ ストレージ, ネットワークの障害時でも, 障害を意識させず, アクセスを可能にする
- ▶ SPOF (Single Point of Failure)をなくす
  - ◎ ある特定部分の故障によりシステム全体が停止することを避ける
- ▶ データの消滅を防ぐため, データ複製を異なる組織に配置し, 二重化, 三重化を図る
- ▶ データへアクセスするためのディレクトリ管理サービスなどの二重化, 三重化を図る



# ファイル管理におけるサービスの連携



# ファイル管理の例

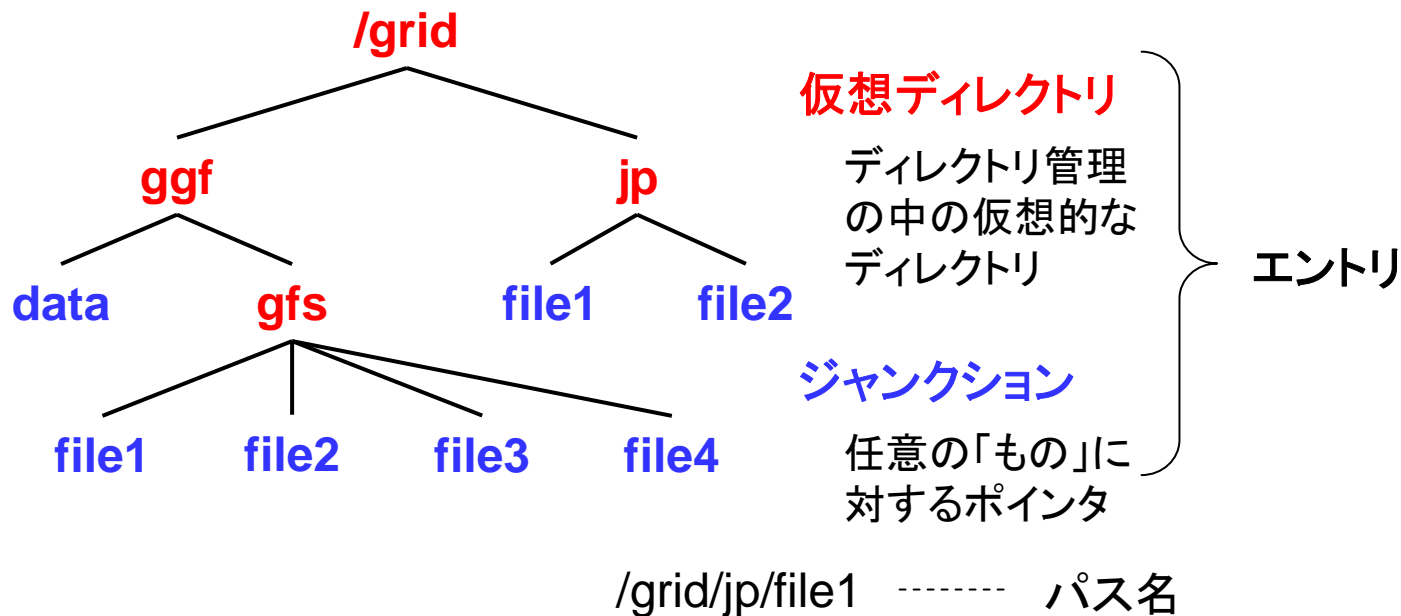


# ディレクトリ管理サービス

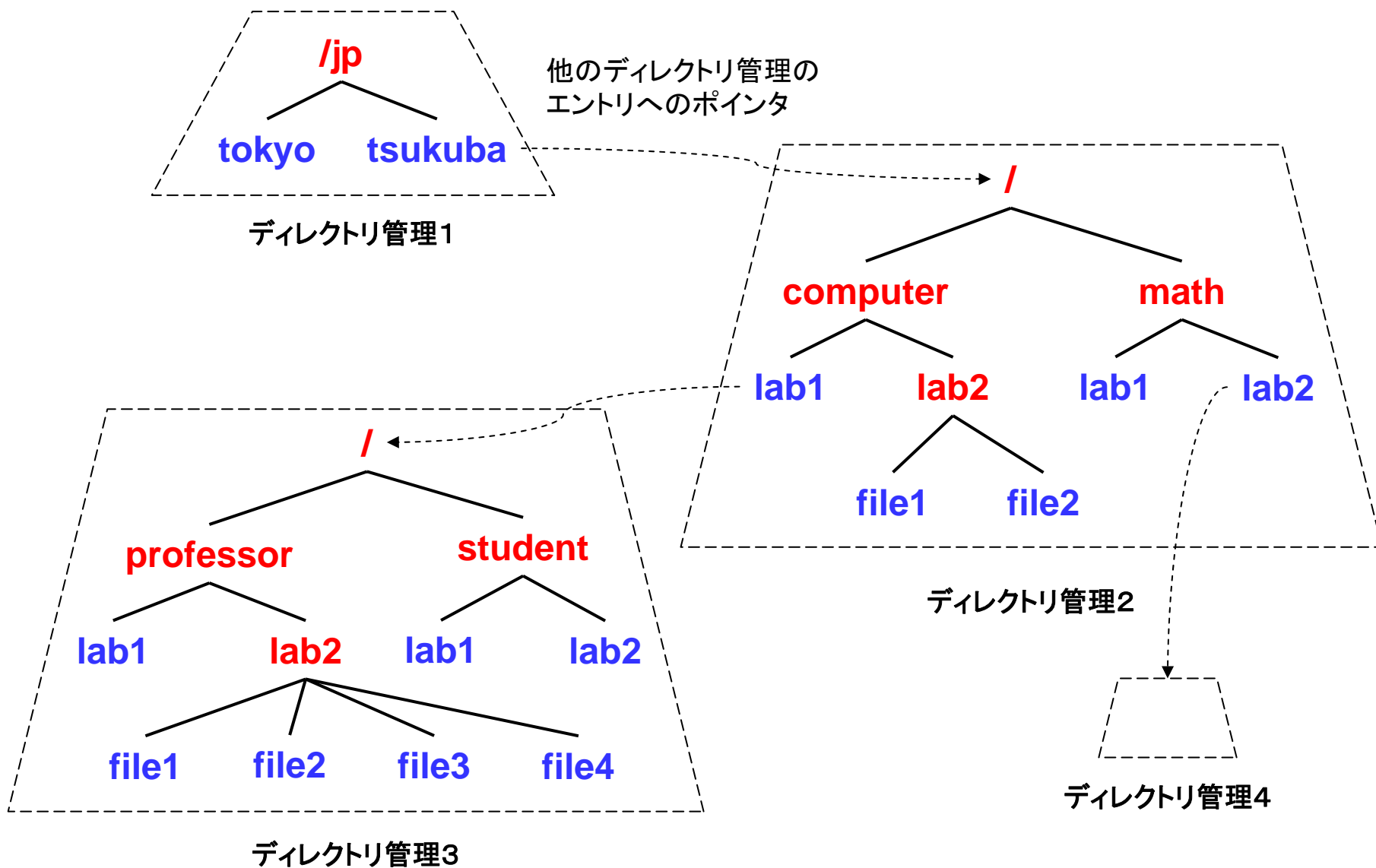
## ● 仮想的な階層名前空間の管理

▶ ファイルシステムのディレクトリツリーのような

## ● ファイル, データ, ファイルシステム, データベースなどを参照するポインタを保持することにより, 分散配置されたデータを統一的な階層名前空間で管理



# 階層化によるディレクトリの分散管理



# ディレクトリ管理の標準化 - RNS

## 🌐 Open Grid Forum

▶ <http://www.ogf.org>

## 🌐 M. Pereira, O. Tatebe, L. Luan, T. Anderson, “Resource Namespace Service Specification”, GFD.101, 2007

## 🌐 5つのインターフェース

▶ Add, list, update, query, remove

## 🌐 エントリに自由にメタデータを追加可能

▶ グリッドファイルシステムのメタデータ管理

▶ バイオ, ナノなどのアプリケーション分野のメタデータ

# RNSの操作

**add ( string: entry\_name, EPR:  
entry\_reference, . . . )**

**list ( string: entry\_name\_regexp )**

**update ( EPR: parent, string: entry\_name, EPR:  
entry\_reference, . . . )**

**query ()**

**remove ( [ string: entry\_name | string:  
regexp ] )**

# Add (1)

## add request message

```
<rns:add>
  <rns:entry_name>
    xsd:string
  </rns:entry_name>
  {any}*
  <rns:entry_reference>
    wsa:EndpointReferenceType
  </rns:entry_reference>
</rns:add>
```

# Add (2)

## addResponse response message

```
<rns:addResponse>  
  <rns:entry_reference>  
    wsa:EndpointReferenceType  
  </rns:entry_reference>  
</rns:addResponse>
```



# ファイルの複製(1)

- アクセス性能の向上, 安定性の向上のため, ファイル複製を作成することは有効
- 複製管理サービスは
  - ▶ 「論理名」を管理し,
  - ▶ 「論理名」から同一な「もの」へのポインター一覧へ変換する
- 「論理名」はファイルやデータを一意に表すID
- UUIDのような人にとって意味をなさないものとなるため, ディレクトリ管理などと併用される
  - ▶ パス名 → 論理名 → ポインタ

# ファイルの複製(2)

## ● 複製選択

- ▶ もっとも「適切な」ファイル複製を選択する
- ▶ データアクセス時間最短など
  - ◎ データサイズ小の時, ネットワーク的に近い場所
  - ◎ データサイズ大の時, バンド幅大の場所

## ● 複製配置

- ▶ 複製を作成する場所を決定する
  - ◎ ファイル複製作成時間を短くしたい
  - ◎ ディザスタリカバリを想定して, 地理的に離れた場所に複製を作成したい
  - ◎ ホットスポットの回避
    - ✦ アクセスが集中しているファイルの特定
    - ✦ アクセス集中をもっとも適切に防ぐように

# 複製管理に関する標準化 – WS-Naming

- 「論理名」からポインタへの変換(リゾルバ)部分
- A. Grimshaw, D. Snelling, “WS-Naming Specification”, GFD.109, 2007
- ポインタはEPR (WS-Addressing Endpoint Reference) で表される
- EPRを拡張, 「論理名」のエンドポイント識別子を埋込
- エンドポイント識別子により, 同一性の判定
- EPRのアドレスを更新するリゾルバのアドレスも埋込まれ, アドレスの更新が可能

# WS-Namingの例

```
<wsa:EndpointReference xmlns:wsa="..." xmlns:naming="...">
  <wsa:Address>http://tempuri.org/application</wsa:Address>
  <wsa:Metadata>
    <naming:EndpointIdentifier>
      urn:guid:B94C4186-0923-4dbb-AD9C-39DFB8B54388
    </naming:EndpointIdentifier>
    <naming:ReferenceResolver>
      <wsa:Address>
        http://tempuri.org/resolver
      </wsa:Address>
    </naming:ReferenceResolver>
  </wsa:Metadata>
</wsa:EndpointReference>
```

エンドポイント識別子

EPR更新のための  
リゾルバ

# ファイル管理に関するサービスの連携

## ● ホットスポットとなるファイルの複製作成

- ▶ アクセス情報サービスによりアクセスが頻繁なファイルリストを取得
- ▶ アクセス頻度から必要な複製数を決定
- ▶ 複製配置により作成先の決定
- ▶ それぞれのファイル複製作成に対し、複製作成元となるファイル複製を複製選択により決定
- ▶ ファイル複製生成のスケジュールの決定
- ▶ スケジュールに従い、データ転送を行う
- ▶ データ転送完了後、複製管理に作成した複製を登録

# サービス連携の問題

## ● エラーや障害発生時の処理

## ● データ転送の途中でネットワーク障害が発生

- ▶ 転送途中のゴミファイルが残ってしまう

→ 一連のサービス連携に対し、トランザクションを保証する

- ▶ 処理を見張るサービスを導入
- ▶ 処理途中のチェックポイントを保存
- ▶ エラー発生時には再実行を試みる
- ▶ 失敗した場合は、ロールバックを行う
- ▶ 処理を見張るサービスが(障害により)中断することにも配慮が必要

# グリッドファイルシステムの例

## ● Gfarmファイルシステム@筑波大

▶ <http://sf.net/projects/gfarm>

● ストレージを束ねた**分散ファイルシステム**

● **マウント**して世界中から利用可能。アプリケーションを選ばない

● 利用者、プログラムの並列性に応じた**スケーラブルなI/O性能**

● 自動ファイル複製選択による**高速データアクセス、耐故障性の実現、アクセス集中の回避**

**Global namespace**

/gfarm

ggf

jp

aist

gtrc

file1

file2

file1

file2

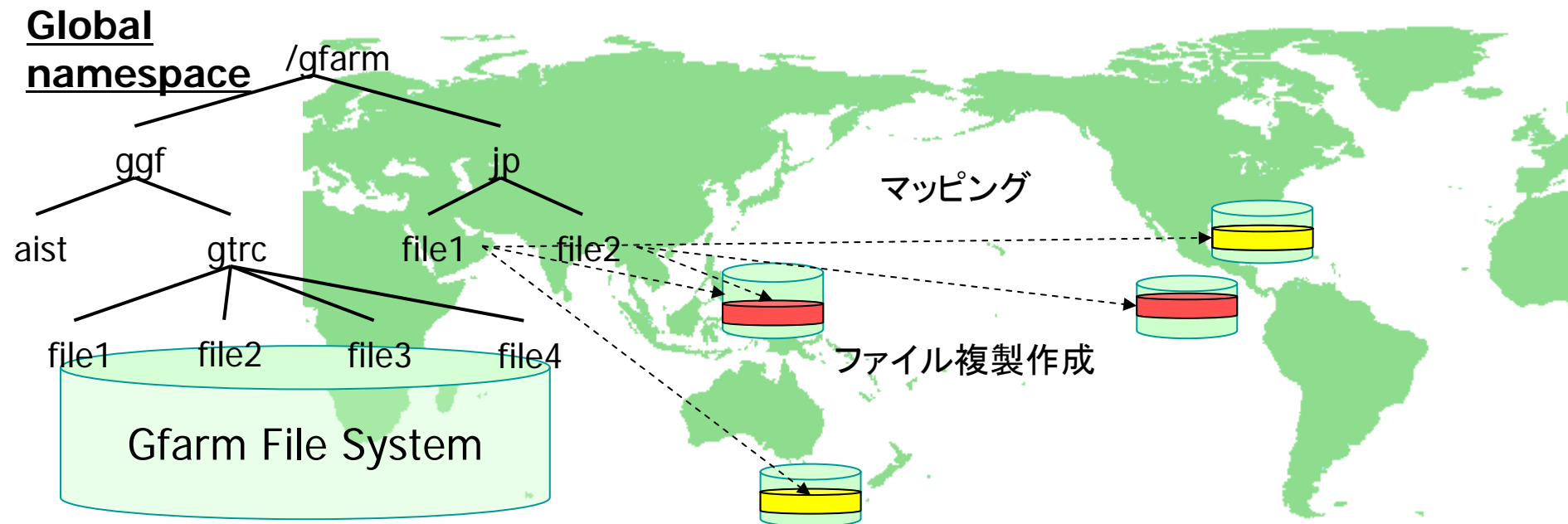
file3

file4

Gfarm File System

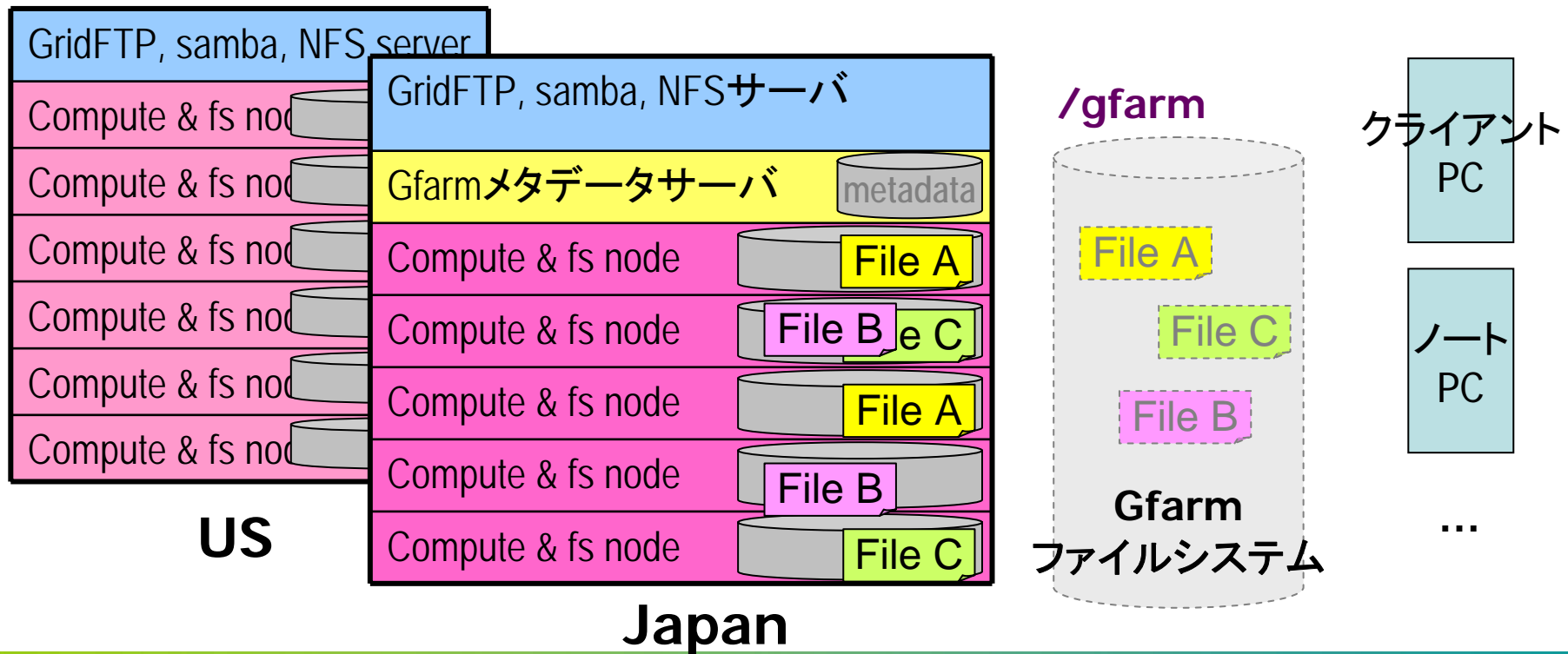
マッピング

ファイル複製作成



# Gfarmグリッドファイルシステム

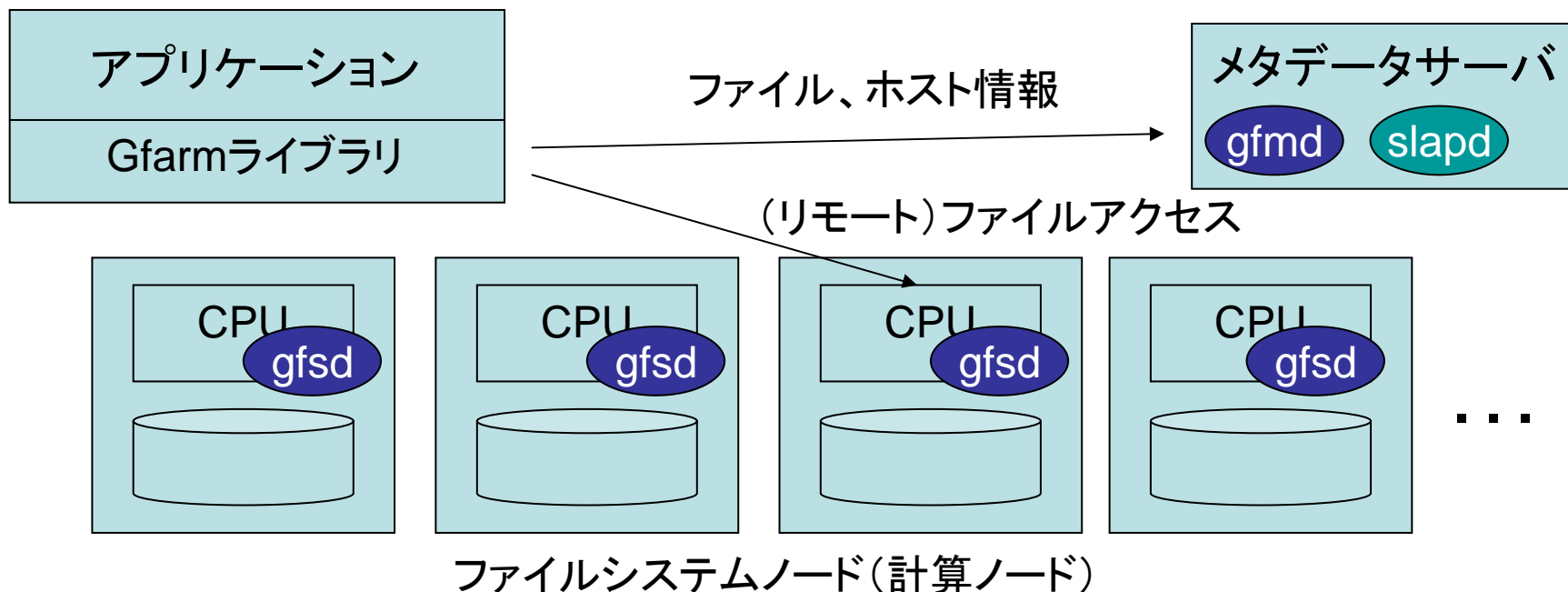
- ファイルはどこからでも共有可能
- 実際には、ファイルは複製が作成され、どこかのファイルシステムノードに格納される
- アプリケーションは格納場所を意識することなくアクセス可能





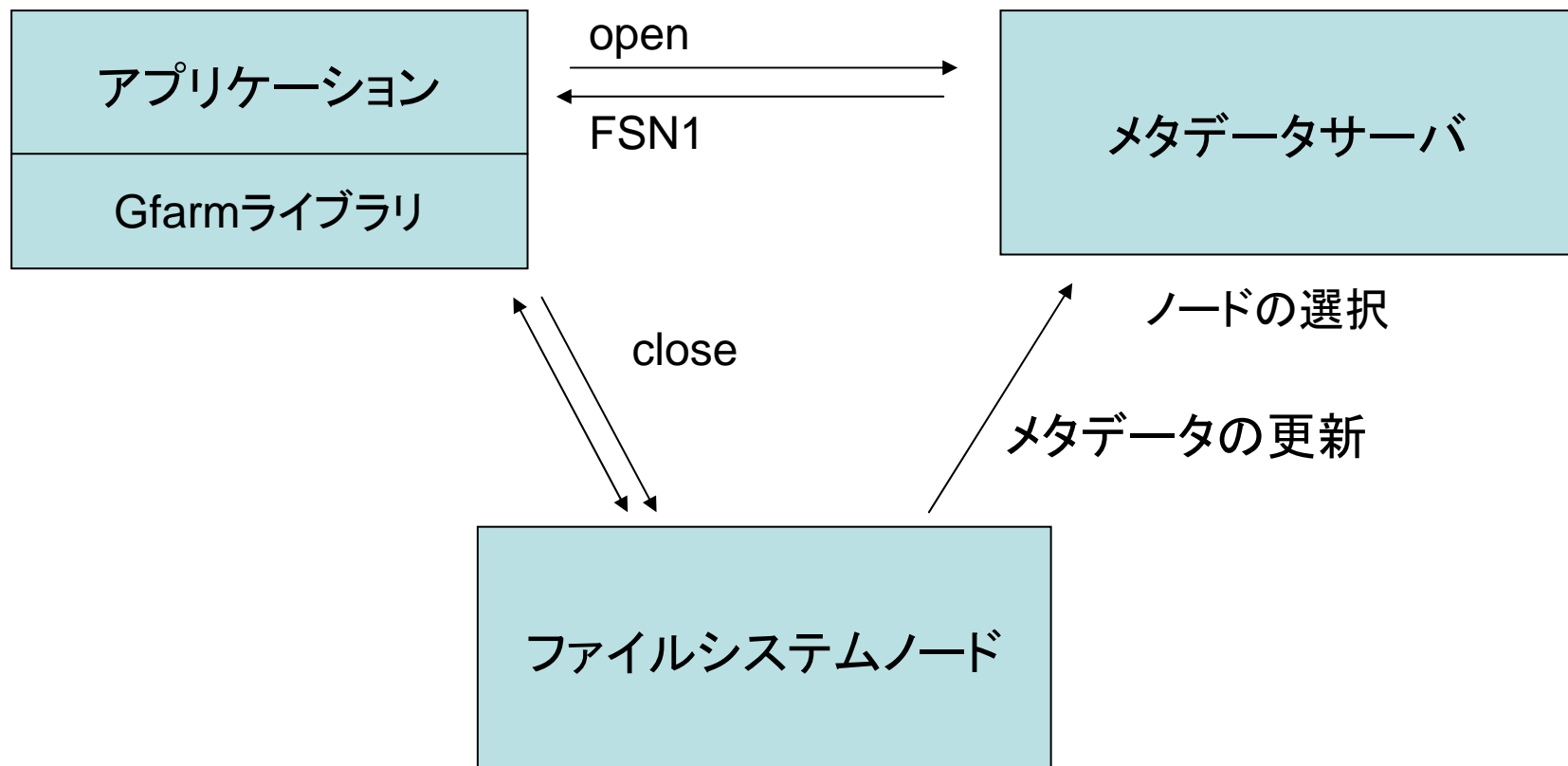
# Gfarmファイルシステムのソフトウェア

- **gfarm2fs, libgfarm** – Gfarmライブラリ
  - ▶ Gfarm API
- **gfmd** – メタデータサーバ
  - ▶ ディレクトリ情報、複製カタログ、ホスト情報、プロセス情報
- **gfsd** – I/Oサーバ
  - ▶ (リモート)ファイルアクセス



# メタデータの一貫した更新

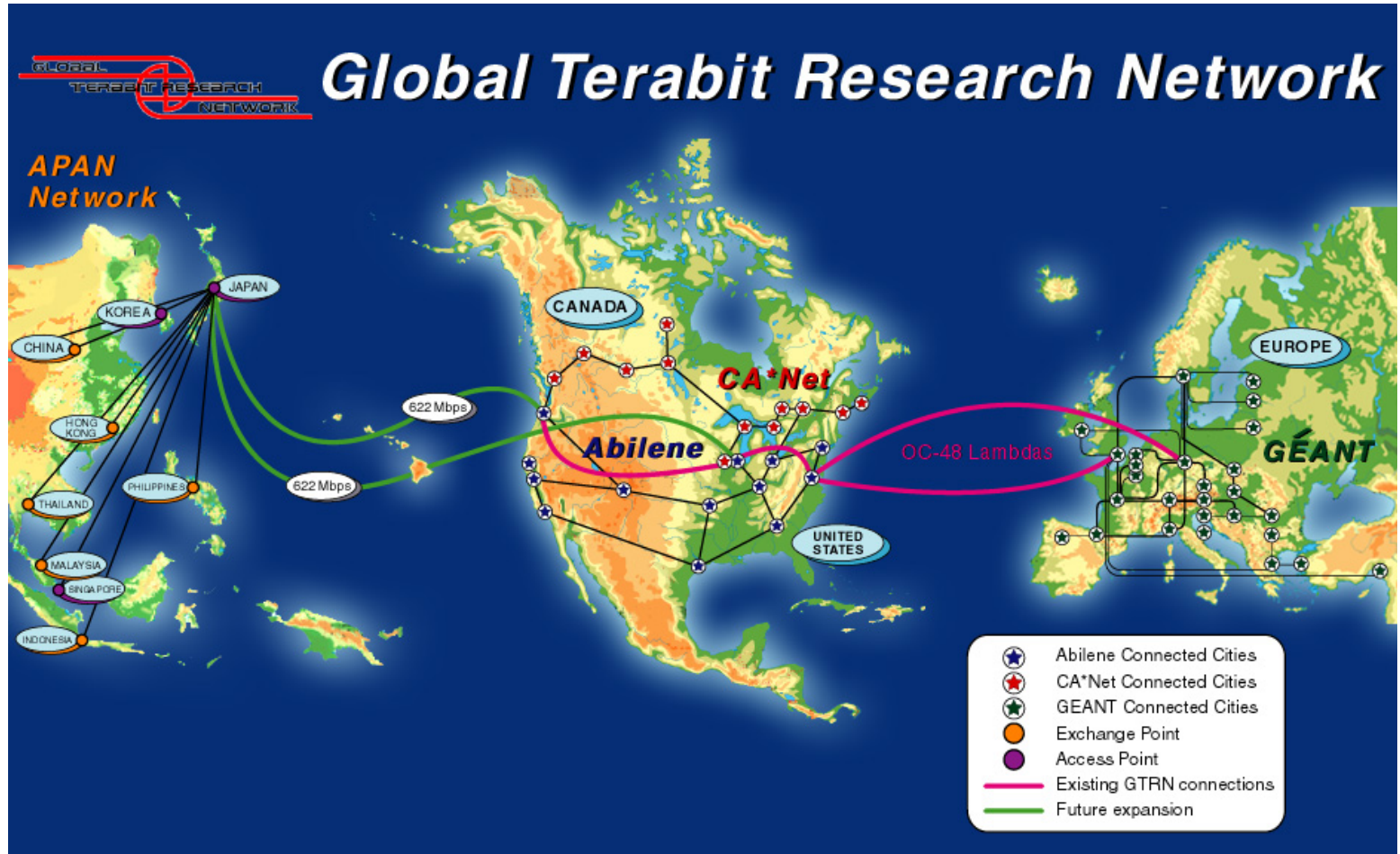
## Gfarm v2 – ファイルシステムノードによるメタデータの更新



**FSN1**

アプリケーション異常終了時にも  
ファイルシステムノードが  
メタデータを更新

# 広域高速データ転送



# IP (RFC791, 1981)

- Internet Protocol
- Internet address で指定される source から destination へ データグラムと呼ばれるデータを転送
- 必要であれば長いデータグラムの分割、再構成も行う
  - ▶ MTU – Maximum Transmission Unit
  - ▶ DFフラグにより断片化禁止
- ネットワーク上のデータグラムの転送のみを提供し、信頼性、フロー制御などは行わない
- 生存期間、チェックサム

## ● Internet address (IP address)

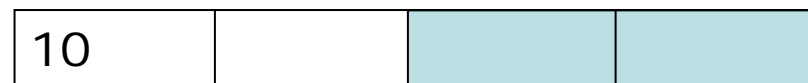
- ▶ Version 4
- ▶ 32 bits

Class A



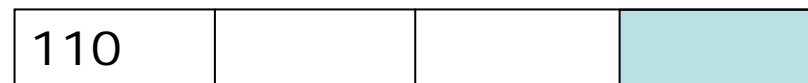
local (host) address

Class B



local address

Class C



network address

Escape to extended addressing mode



# TCP (RFC793, 1981)

## ● Transmission Control Protocol ● プロセス間の信頼性のある通信サービス

### ● 基本データ転送

- ▶ 双方向のバイトストリームの転送
- ▶ 転送の確認のためのプッシュファンクション。プッシュは直ちにデータを送信する

### ● 信頼性

- ▶ TCPはデータ化け、損失、重複、順序の入れ替えなどを復元
- ▶ Sequence numberと acknowledgment (ACK)による
- ▶ ACKがタイムアウト (Retransmission timeout; RTO) したら再送する
- ▶ 受信側はsequence numbersにより重複、順序の入れ替えを検出
- ▶ それぞれのセグメントにチェックサムを付加することによりデータ化けを検出、検出時は捨てる

### ● フロー制御

- ▶ 受信側が送信データ量を制御
- ▶ それぞれのACKと同時に "window" を返す
- ▶ "window" は、さらなる許可なしに送信してもいいデータ量を示す
- ▶ 輻輳制御
  - 送信側、受信側、その間のネットワークのうち送信側以外にボトルネックがあった場合、ボトルネックを超えて送信し続けることを防ぐ

### ● 多重送信 (multiplexing)

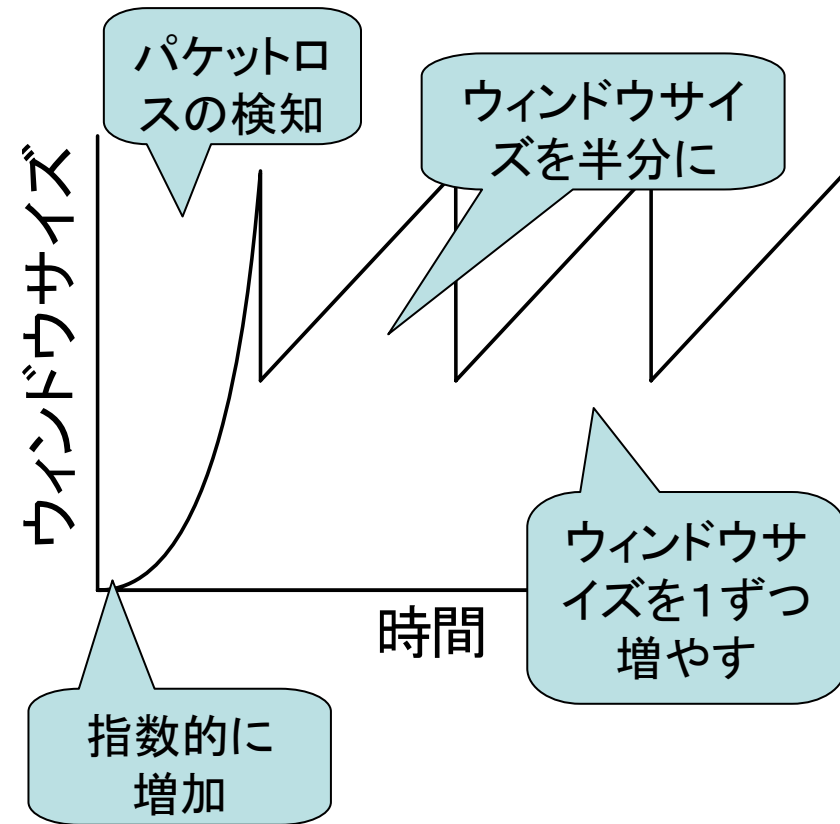
- ▶ 単一ホストの多くのプロセスがTCPを利用するためにいくつかのポートがある
- ▶ ポートとホストアドレスでソケット、コネクションはソケットのペア
- ▶ Well-known ポート番号 (cf. /etc/services)

### ● コネクション

- ▶ ソケット、sequence numbers、ウィンドウサイズなど通信の情報をコネクションと呼ぶ
- ▶ プロセス間通信の前にコネクションを確立する必要がある
- ▶ 信頼性のないホストと信頼性のないインターネットにおけるコネクションの確立のため、clock-based sequence numbers が用いられる

# ウィンドウ（輻輳）制御アルゴリズム

- Reno, Newreno, Vegas, Sack, Fack などが知られる
- Renoが現状で最もよく使われ、かつほかのアルゴリズムより強い
- Renoでの輻輳ウィンドウの変化
  - ▶ スロースタートフェイズ
    - ◎ 輻輳ウィンドウが最小の場合
    - ◎ ウィンドウサイズを指数的に増加
  - ▶ 輻輳回避フェイズ
    - ◎ ウィンドウサイズを1ずつ増加
    - ◎ パケットロスが起きた時点で輻輳ウィンドウを半分に
- ⇒バンド幅を75%程度しか使用できない



# 高速広域ネットワークにおけるデータ転送

- LFN (elephan(t), Long, Fat Network)におけるTCPの性能低下
- RFC1323 TCP Extensions for High Performance (1992)
  - ▶ TCPの転送速度はネットワーク速度ではなく、バンド幅 × 遅延による
  - ▶ バンド幅 × 遅延はネットワーク上で転送されるデータ量であり、TCPで最大バンド幅を出すためには、このバッファ量が送信側、受信側に要求される

# TCP over LFN における性能の問題

## ● ウィンドウサイズの制限

- ▶ 受信ウィンドウサイズはTCPヘッダの16bitのフィールドで送信側に送られる
  - ◎ 最大ウィンドウサイズ=64KB
  - ◎ バンド幅の上限=64KB/RTT
- ▶ TCPオプションWindow Scaleの導入 (RFC1323)
  - ◎ 16bit -> 30bit = 1GB (31bit sequence number の制限)

## ● パケットロスからの復帰

- ▶ LFNにおけるパケットロスは悲劇的
- ▶ パケットロスにより、データパイプラインのフラッシュとスロースタートによる復帰
- ▶ ウィンドウサイズを広げるによりより可能性が大

## ● 往復の遅延測定

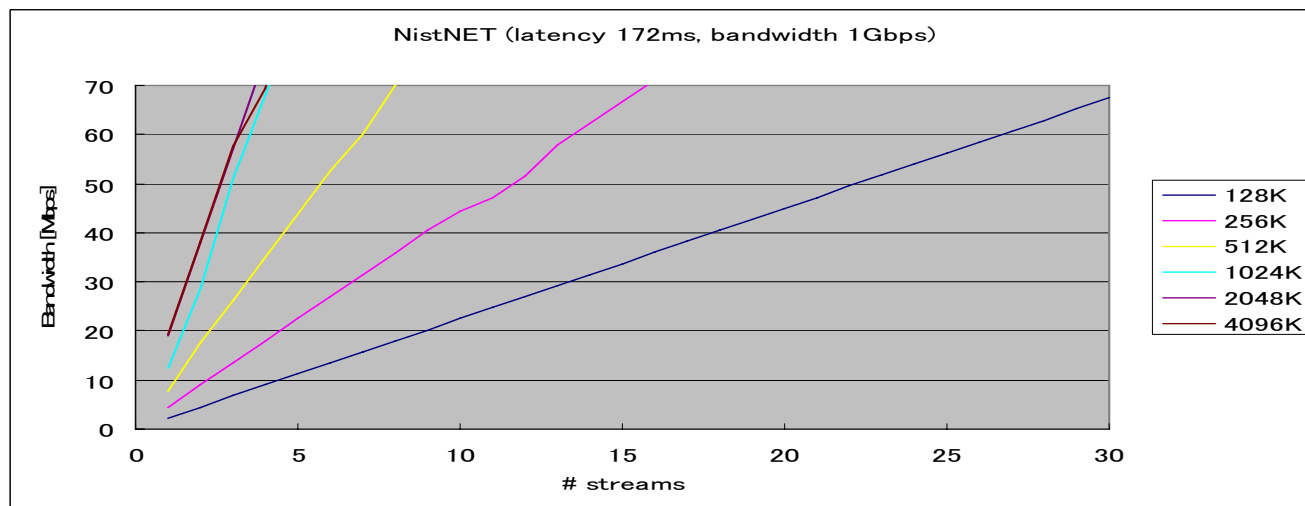
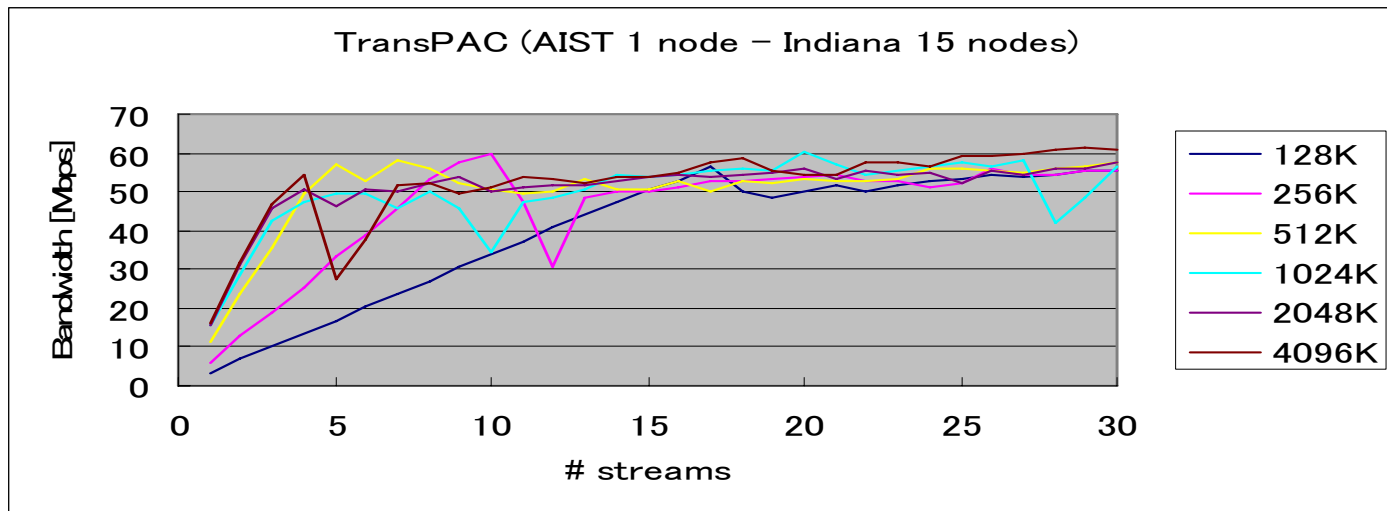
- ▶ 適切なRTOの動的決定がTCP性能において重要
- ▶ RTOはRTT (round-trip time) の平均と分散により決定
- ▶ TCPオプションTimestampsの導入 (RFC1323)



# ネットワークストライピング

- 大容量ソケットバッファサイズの指定はroot権限が必要
- アプリケーションレベルの複数ストリームによる転送
- デフォルトのバッファサイズ(=64KB) × ストリーム数を指定する効果
- 実験では上記をしのぐ効果

# TransPAC (Tokyo – Seattle)とNistNETによる並列ストリームのバンド幅の評価



# GridFTP (GFD20, 2003)

- **GridFTP: extended version of popular FTP protocol for Grid data access and transfer**
- **Secure, efficient, reliable, flexible, extensible, parallel, concurrent, e.g.:**
  - ▶ Third-party data transfers, partial file transfers
  - ▶ Parallelism, network striping, striping server (e.g., on PVFS)
  - ▶ Automatic and manual TCP tuning
  - ▶ Reliable, recoverable data transfers, data channel authentication
- **Reference implementations**
  - ▶ gridftp-server, globus-url-copy, uberftp
  - ▶ Flexible, extensible libraries in Globus Toolkit

# GridFTPによる拡張

## ● プロトコル拡張

SPAS	Striped Passive	Host/portの配列を返す
SPOR	Striped Port	Host/portの配列を返す
ERET	Extended Retrieve	ファイルを一部分を転送する
ESTO	Extended Store	ファイルの一部分を格納する
SBUF	Set TCP Buffer Size	TCPのバッファサイズを設定する
ABUF	Auto-negotiate TCP Buffer Size	自動的にTCPのバッファサイズを決定する
DCAU	Data Channel Authentication	制御チャンネルはRFC2228でGSS認証が導入されたが、データチャンネルにも導入

## ● モード拡張

- ▶ EBLOCK (Extended block)モード
- ▶ データをブロックに分割して送信可能とするため
- ▶ 8bitのフラグ, 64bitのサイズ, 64bitのオフセット, データ

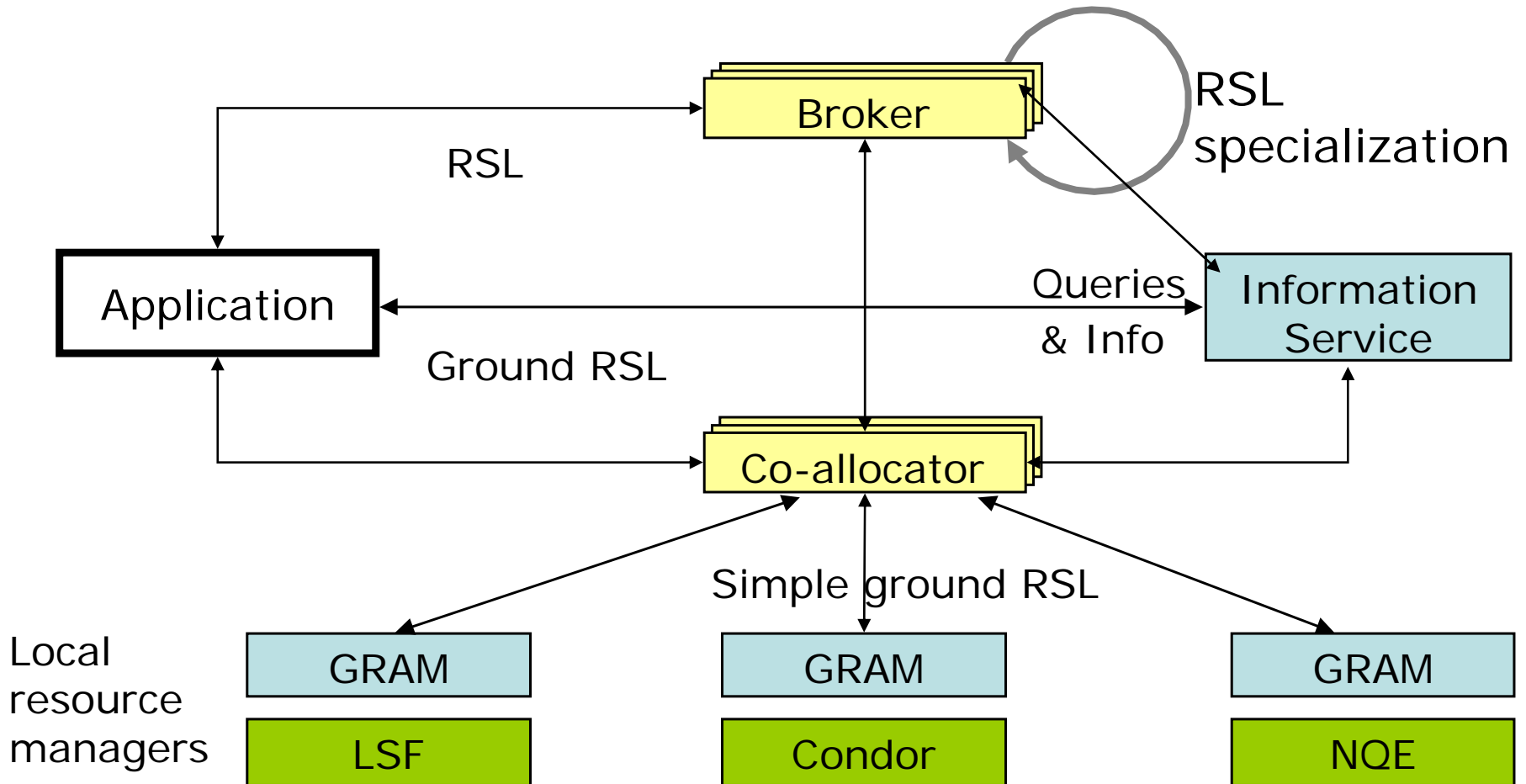
# 参考文献：広域高速データ転送

- H. Sivakumar, S. Bailey, R. L. Grossman. Pockets: The Case for Application-level Network Striping for Data Intensive Applications using High Speed Wide Area Networks, Proc. SC2000  
<http://www.sc2000.org/techpaper/papers/pap.pap240.pdf>
- Thomas Dunigan, Matt Mathis, Brian Tierney. A TCP Tuning Daemon, Proc. SC2002  
<http://www.sc2002.org/paperpdfs/pap.pap151.pdf>
- Thomas J. Hacker, Brian D. Noble, Brian D. Athey. The Effects of Systemic Packet Loss on Aggregate TCP Flows, Proc. SC2002  
<http://www.sc2002.org/paperpdfs/pap.pap270.pdf>
- B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, S. Tuecke. Data Management and Transfer in High Performance Computational Grid Environments. Parallel Computing Journal, Vol. 28 (5), May 2002, pp. 749-771. <http://www.globus.org/research/papers/dataMgmt.pdf>
- W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, S. Meder, S. Tuecke. GridFTP Protocol Specification. GGF GridFTP Working Group Document, September 2002.  
<http://www.globus.org/research/papers/GridftpSpec02.doc>

# 参考文献：複製管理

- A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunst, M. Ripenu, B. Schwartzkopf, H. Stockinger, K. Stockinger, B. Tierney. Giggle: A Framework for Constructing Scalable Replica Location Services. Proc. SC2002  
<http://www.sc2002.org/paperpdfs/pap.pap239.pdf>

# Resource Management



# 参考文献：資源管理、資源割当

- Rajesh Raman, Miron Livny, and Marvin Solomon, Resource Management through Multilateral Matchmaking, Proc. Ninth IEEE Symposium on High Performance Distributed Computing (HPDC9), August 2000, pp 290-291.

<http://www.cs.wisc.edu/condor/doc/gangmatching.ps>

- Fabio Kon, Roy Campbell, M. Dennis Mickunas, Klara Nahrstedt, and Francisco J. Ballesteros. 2K: A Distributed Operating System for Dynamic Heterogeneous Environments. Proc. Ninth IEEE Symposium on High Performance Distributed Computing (HPDC9), August 2000.

<http://choices.cs.uiuc.edu/2k/papers/hpdc2000.pdf>