

---

# 計算資源とネットワーク資源を同時確保する 予約ベースグリッドスケジューリングシステム

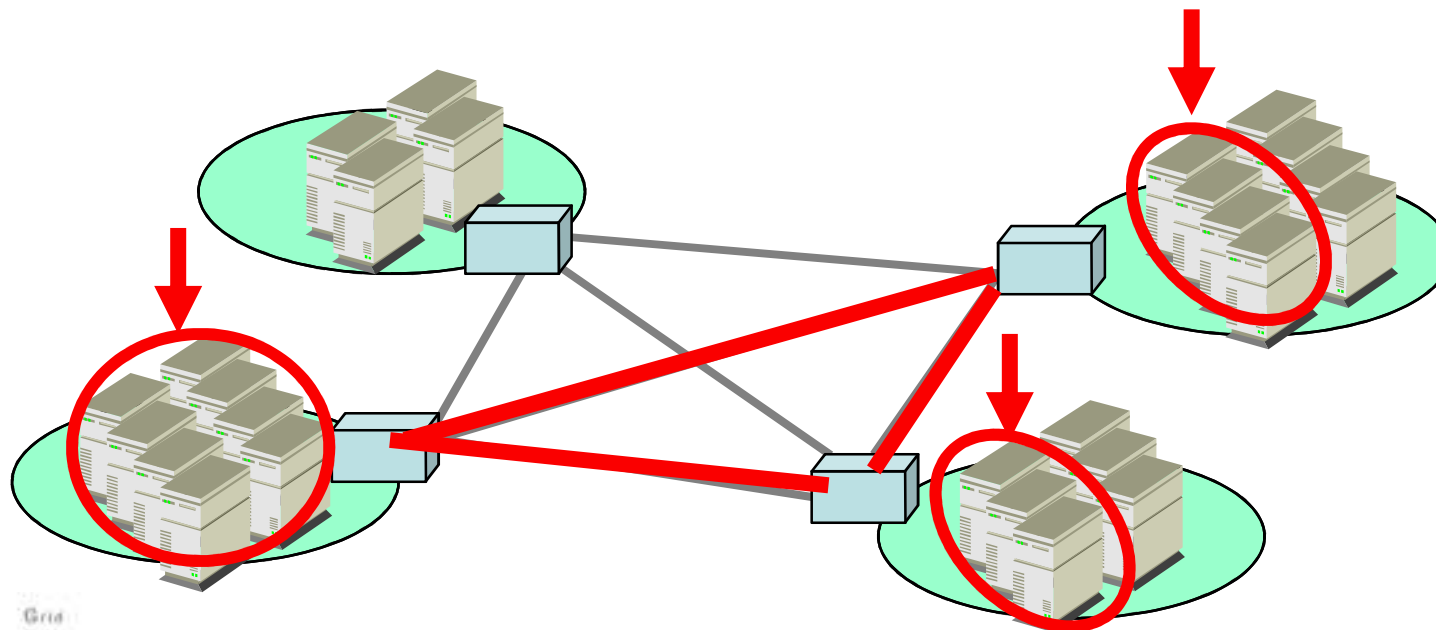
竹房あつ子, 中田秀基, 工藤知宏,  
田中良夫, 関口智嗣

産業技術総合研究所 グリッド研究センター



# グリッドにおけるメタコンピューティング

- 異なる組織から提供される分散した計算資源を同時に利用
- 計算機の性能・負荷とクラスタ間のネットワークの通信遅延・帯域がアプリケーションの実効性能に影響



クラスタ計算機

# グリッド資源のコアロケーション

- グリッド上の異なる組織から提供されるネットワーク資源と計算資源の**コアロケーション(同時確保)**が重要
    - ▶ インターネットではベストエフォート型の通信を行うため、通信性能を保証できない
    - ▶ 光ネットワークでは高品質通信が提供可能だが、必要なネットワークの構成・提供は自動化されていない
    - ▶ スーパースケジューラはコアロケーションを実現するものであるが、実用段階に至っていない
- ユーザアプリケーションの**性能保証**が困難
- 資源を利用する際に**人為的手続き**が必要

# 事前予約に基づくグリッドスケジューリングシステム

- 計算・ネットワーク資源を事前予約により同時確保
- グリッド資源スケジューラ(GRS)と計算資源マネージャ(CRM)からなる
  - ▶ WSRFベースインタフェースを提供
  - ▶ CRMでは既存ローカルスケジューラ(TORQUE)に事前予約機能を提供するPluSプラグインを開発
- ネットワークキャリアの提供するネットワーク資源管理システム(NRM)と連携
  - ▶ G-lambdaプロジェクトで提案するGNS-WSI (Grid Network Service - Web Services Interface) を提供



Grid  
Technology  
Research  
Center  
AIST

④ 産総研, KDDI研究所, NTTの共同プロジェクト

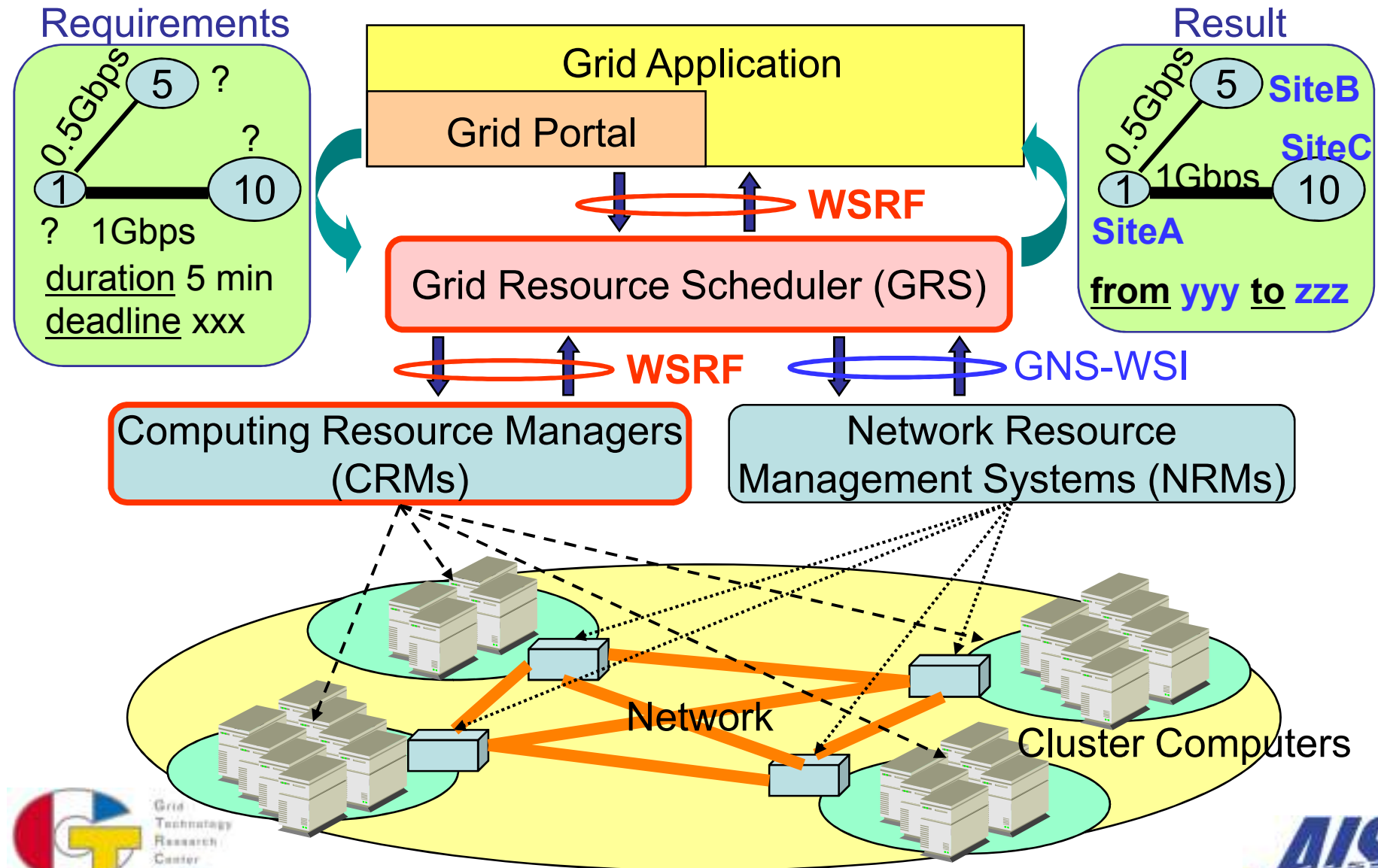


# 発表内容

---

- 事前予約に基づくグリッドスケジューリングシステムの概要
- WSRFによるシステムアーキテクチャ
  - ▶ リソースステータス
  - ▶ 事前予約・解放・監視プロトコル
- GRSとCRMの実装
  - ▶ CRMではPluSローカルスケジューラpluginを開発
- 予備実験
- 関連研究
- まとめと今後の課題

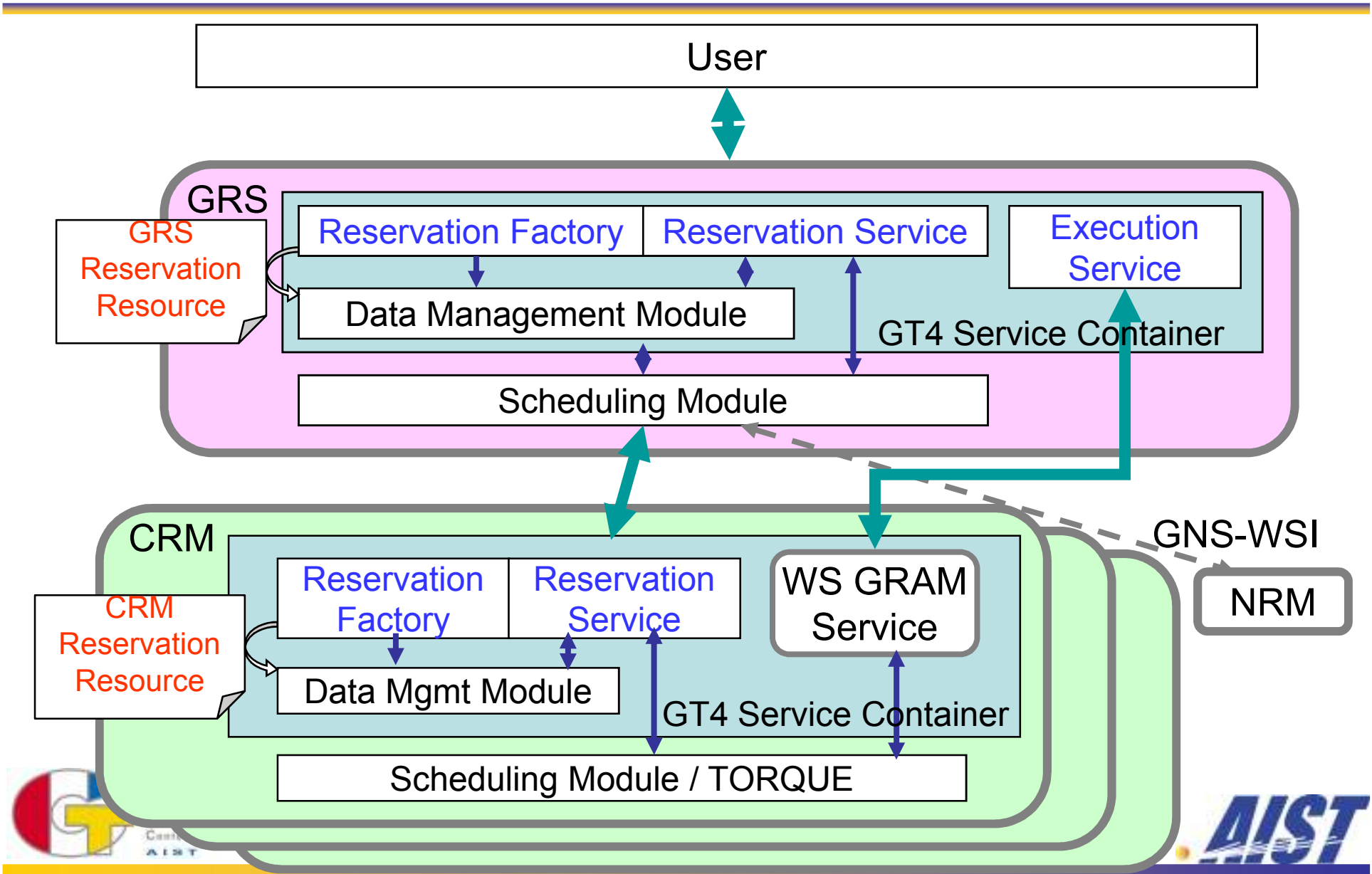
# 事前予約に基づくグリッドスケジューリング システムの概要



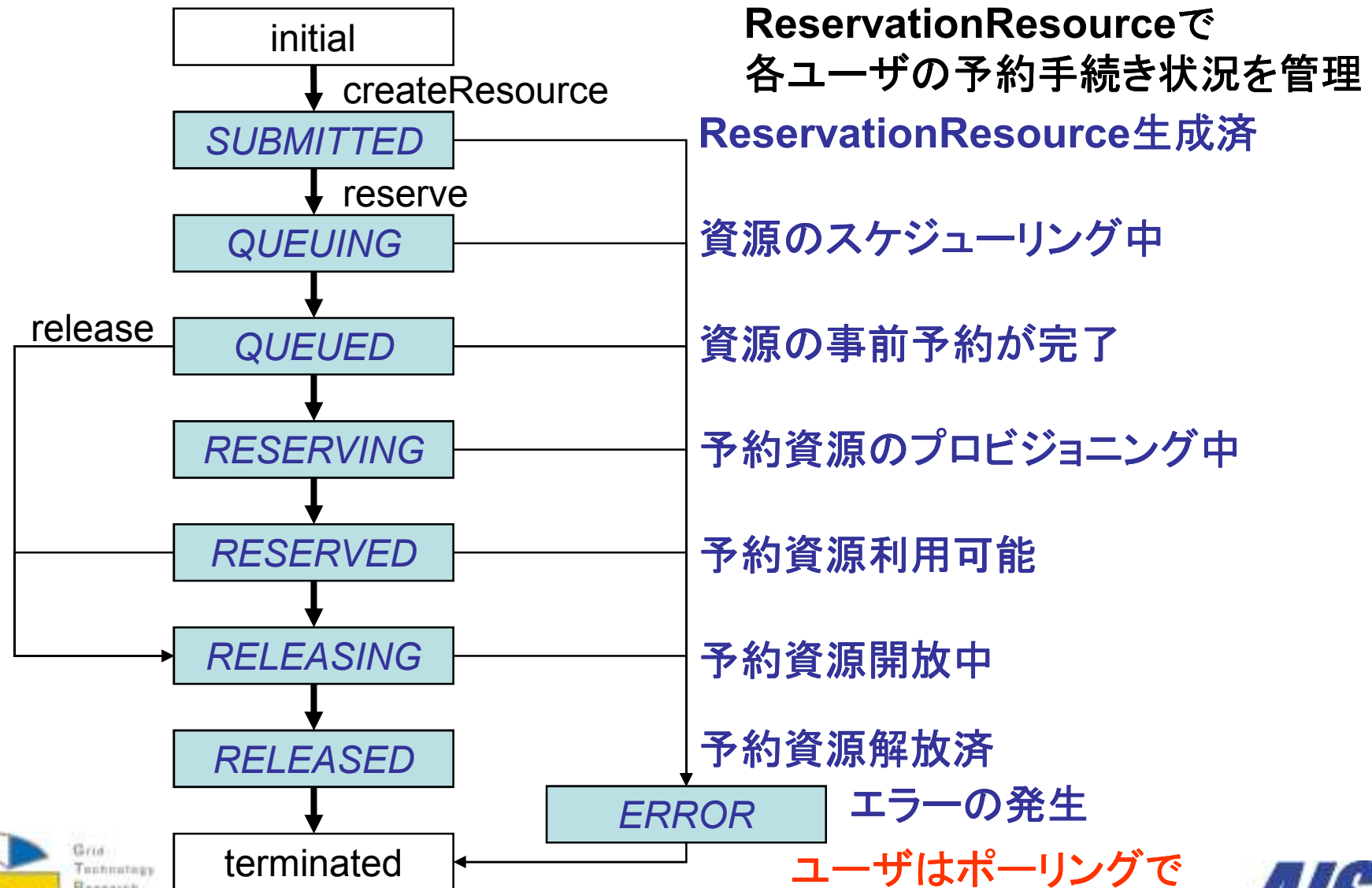
# グリッドスケジューリングシステムWSRFインタフェース

- GRS, CRMではWSRFに基づくインタフェースを提供
- GT4 (Globus Toolkit 4) を用いて開発
  - ▶ グリッドジョブの投入, モニタリング, キャンセルにWSGRAMを利用
  - ▶ Grid Security Infrastructure (GSI)とgrid\_mapfileによる認証・認可
- GRSとCRMの構成
  - ▶ ReservationFactory: ReservationResourceサービスインスタンスを生成
  - ▶ ReservationService: 事前予約に関するオペレーションを提供
  - ▶ GRSではジョブサブミッションのためのExecutionServiceを提供

# WSRFベーススケジューリングシステムアーキテクチャ



# GRS/CRMReservationResourceのステータス遷移



# GRS, CRMの事前予約プロトコル

## ● ポーリングベースのオペレーションを提供

- ▶ WS-Notificationを用いると、プロトコルが複雑になり、耐故障性のためにポーリングの併用も必要

## ● WSRFベース事前予約プロトコル

### ▶ 事前予約

- ◎ ユーザの資源要求に従い、GRSが計算・ネットワーク資源を事前予約により確保
- ◎ 複数CRM, NRMと連携し、GRSが資源要求を満たす資源を選択して事前予約

### ▶ 解放

- ◎ *QUEUED*または*RESERVED*状態の資源の解放

### ▶ モニタリング

- ◎ フォルトリカバリのため、GRSが予約資源の状況を監視

# GRSの予約関連サービスオペレーション

	オペレーション名	機能	入力	出力
F	createResource	GRSRRの生成	なし	EPR
R	reserve	予約処理を開始	(開始時刻, )予約期間, デッドライン, 資源要求	なし
	getResourceProperty (status)	予約状況を取得	リソースプロパティ名	予約状 況
	getResourceProperty (GridResources)	予約した資源情 報の取得	リソースプロパティ名	予約資 源情報
	release	予約した資源情 報の解放	なし	なし
E	submit	ジョブを各CRM へ送信	ジョブ情報	なし
	cancel	submitしたジョ ブのキャンセル	なし	なし

F: ReservationFactory

R: ReservationService

E: ExecutionService

GRSRR: GRSReservationResource



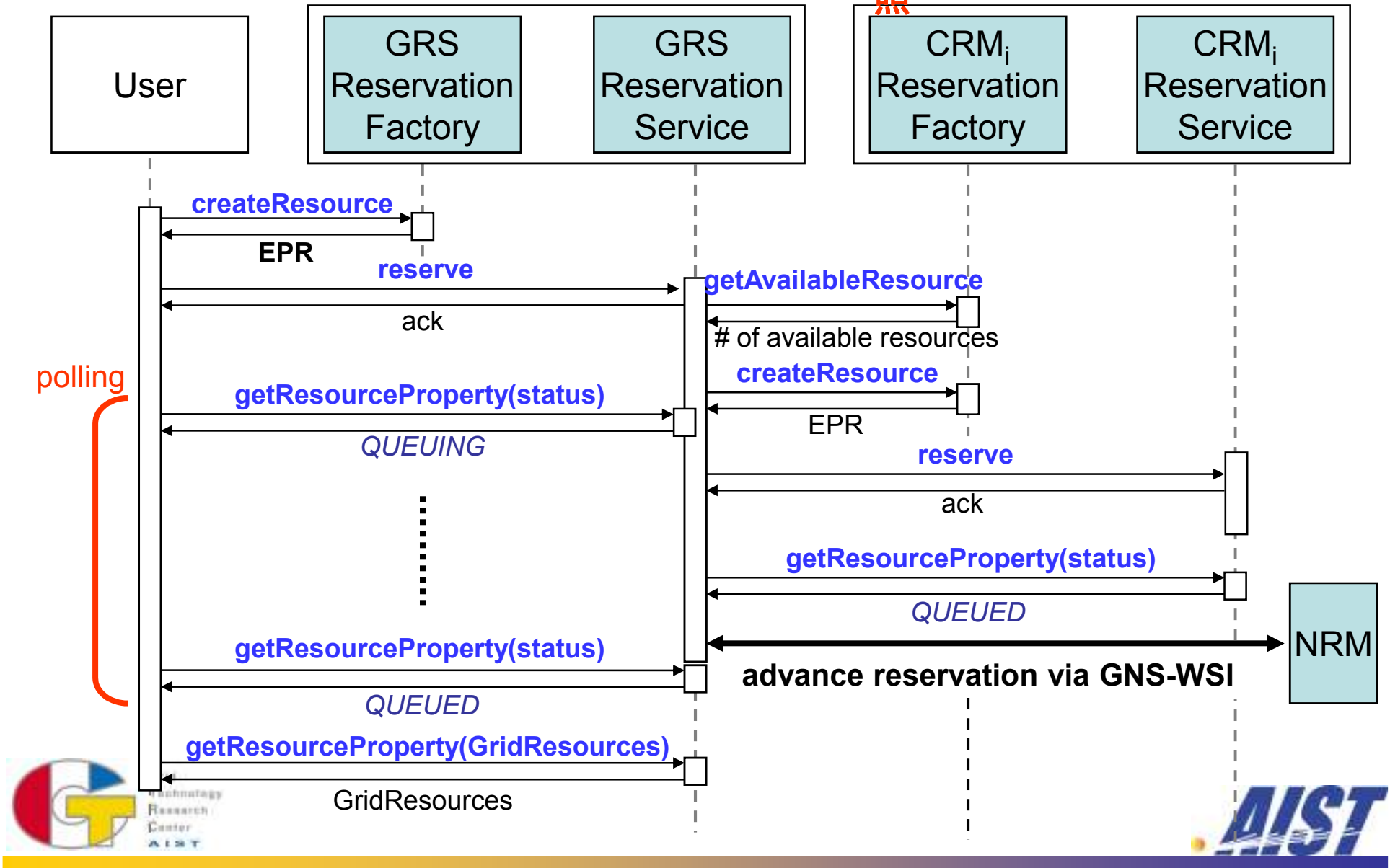
# CRMの予約関連サービスオペレーション

	オペレーション名	機能	入力	出力
F	createResource	CRMRRを生成	なし	EPR
	getAvailableResources	利用可能な資源情報を提供	開始時刻, 終了時刻	資源情報
R	reserve	予約処理を開始	開始時間, 終了時間, CPU数	なし
	getResourceProperty (status)	予約情報(IDなど)を取得	リソースプロパティ名	予約情報
	cancel	予約をキャンセル	なし	なし
	modify	予約の変更	開始時刻, 終了時刻, CPU数	なし

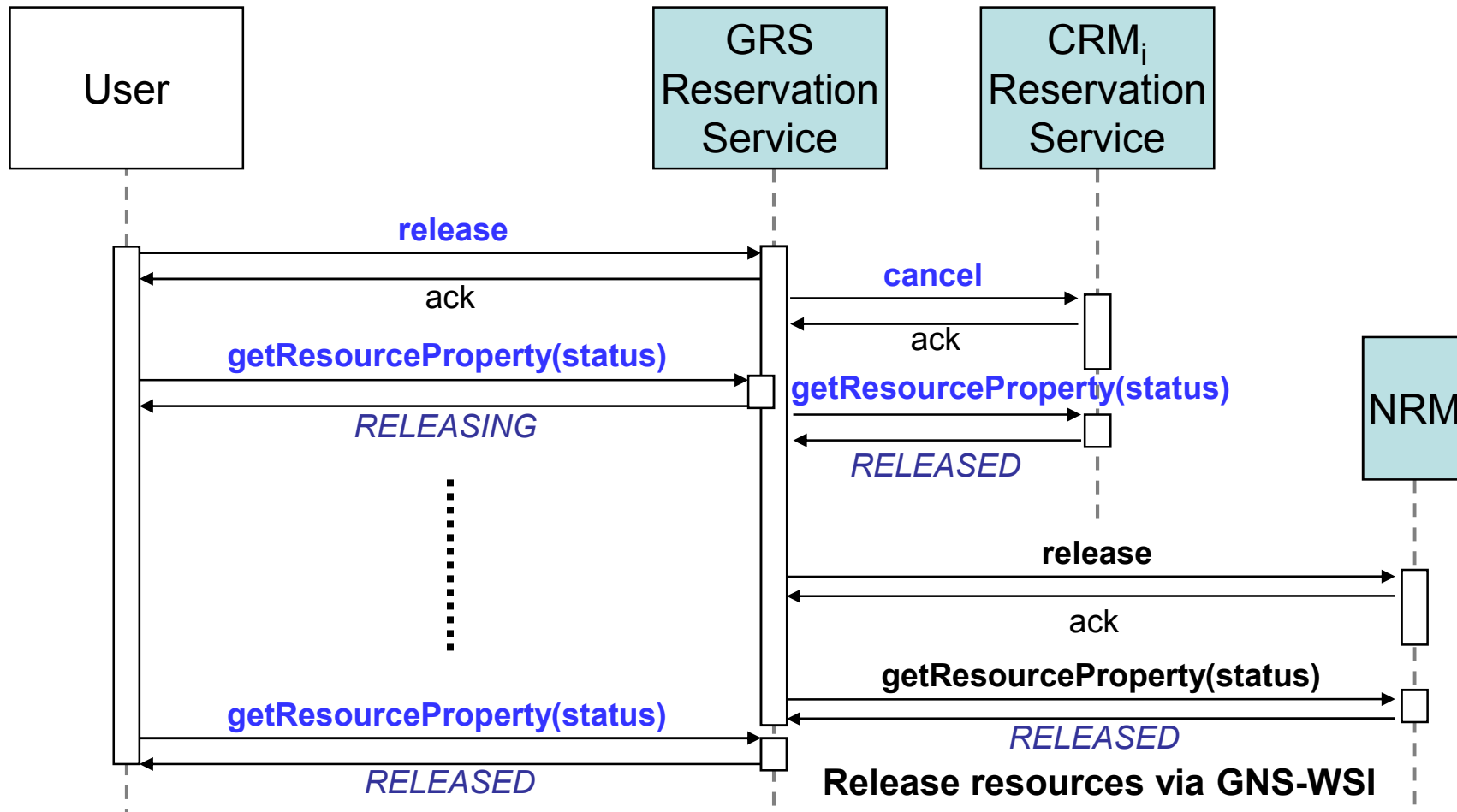
F: ReservationFactory  
R: ReservationService

CRMRR: CRMReservationResource

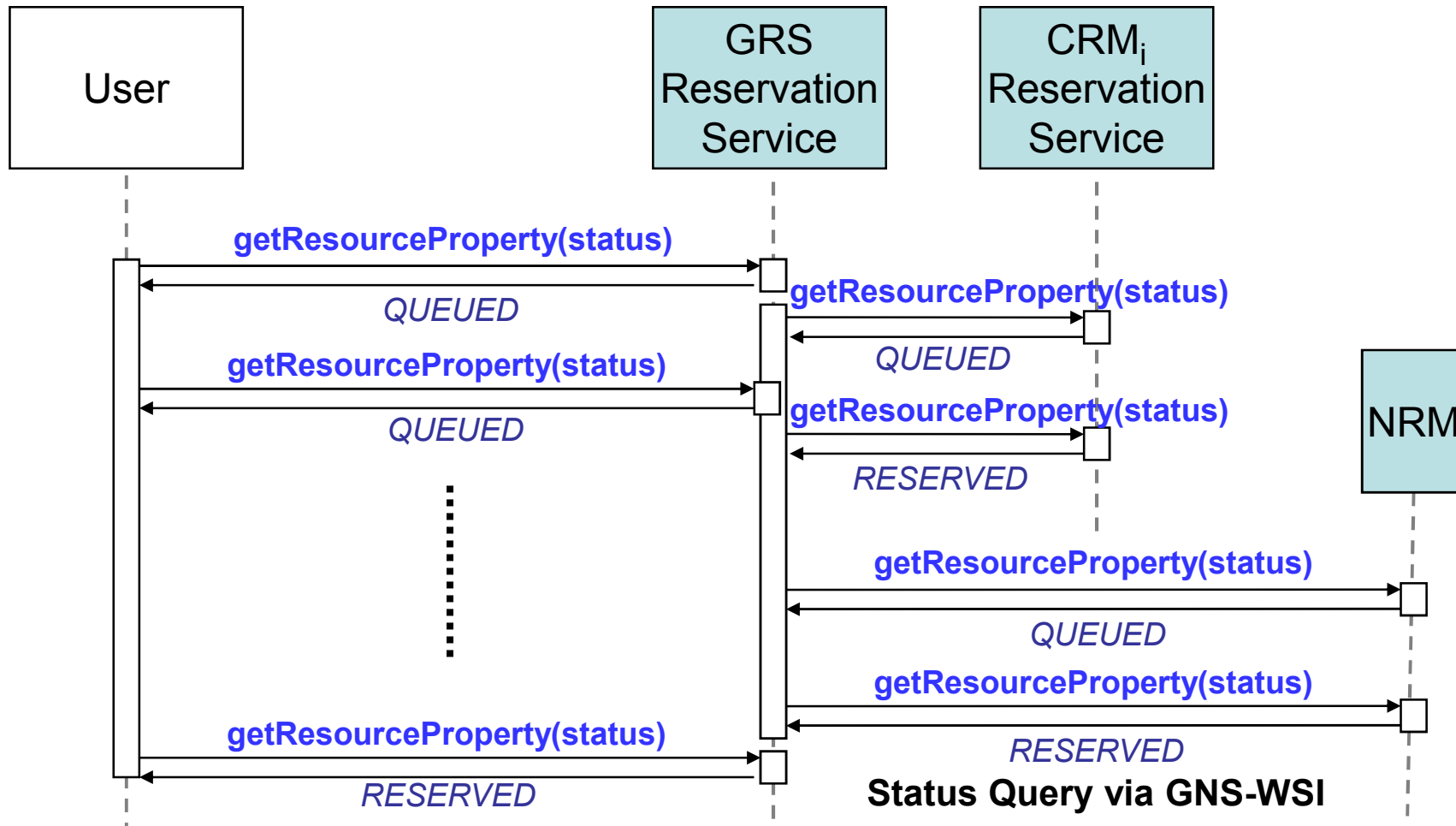
# 事前予約プロトコルシーケンス EPR: endpoint reference サービスインスタンスへの参照



# 解放プロトコルシーケンス



# モニタリングプロトコルシーケンス



# GRSの実装

---

## ● GT4 Java WS Coreを用いて実装

## ● GRSの構成

### ▶ WSRFインタフェース

Ⓞ 事前予約インタフェース: ReservationFactory, ReservationService

Ⓞ ジョブ実行インタフェース: ExecutionService

⊕ 予約資源上での自動ジョブ実行を実現

### ▶ データ管理モジュール

Ⓞ 予約情報(ReservationResource)を永続化

### ▶ スケジューリングモジュール

Ⓞ 複数CRM, NRMと連携し, 適切な資源を選択・予約

## GRSコマンドラインインタフェース (1/2)

---

### 🌐 ReservationResourceの生成

```
$ java jp.aist.gtrc.scheduler.client.CreateClient ¥  
-s [Factory URI]
```

ReservationResourceのEPRを返す

### 🌐 Reserve required resources

```
$ java jp.aist.gtrc.scheduler.client.ReserveClient ¥  
-e [EPR file name] -r [requirement XML file name]
```

予約が成功した場合、予約した資源情報を返す

- 🕒 予約開始・終了時刻
- 📍 計算資源のサイト名
- 🌐 各サイトのジョブマネージャのURI
- 📄 CRMとNRMから返された予約ID

## GRSコマンドラインインタフェース (2/2)

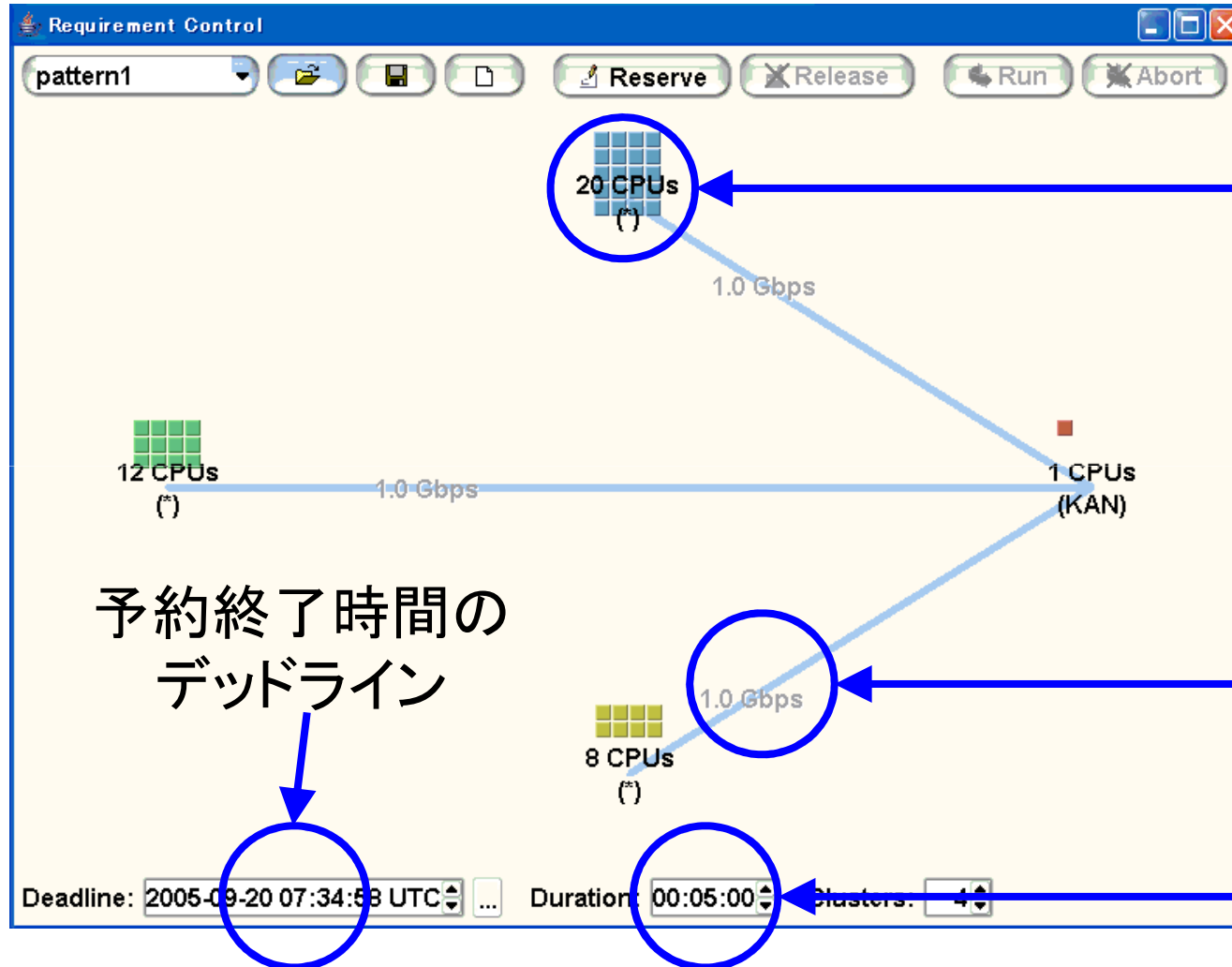
### ● 予約資源の解放

```
$ java jp.aist.gtrc.scheduler.client.ReleaseClient ¥  
-e [EPR file name]
```

### ● 資源要求XMLのサンプル

```
<requirements>  
  <deadline value="2006-05-27T18:00:00Z" />  
  <duration value="600000" /> <!-- msec -->  
  <clusters>  
    <cluster id="0" numProcessors="2" />  
    <cluster id="1" numProcessors="4" />  
  </clusters>  
  <networks>  
    <network id="0" APoint="0" ZPoint="1" bandwidth="1000000"/>  
    <!-- kbps -->  
  </networks>  
</requirements>
```

# GRS GUI



CPU数の指定  
クラスタサイト名も  
指定可能 (任意)

予約終了時間の  
デッドライン

パスとバンド幅の  
指定

予約時間

4クラスタ37CPUの計算資源と  
スター型パスネットワークを要求

# CRMの実装

## ● CRMの構成

▶ GT4 Java WS Core  
を用いた WSRF事前  
予約インタフェース

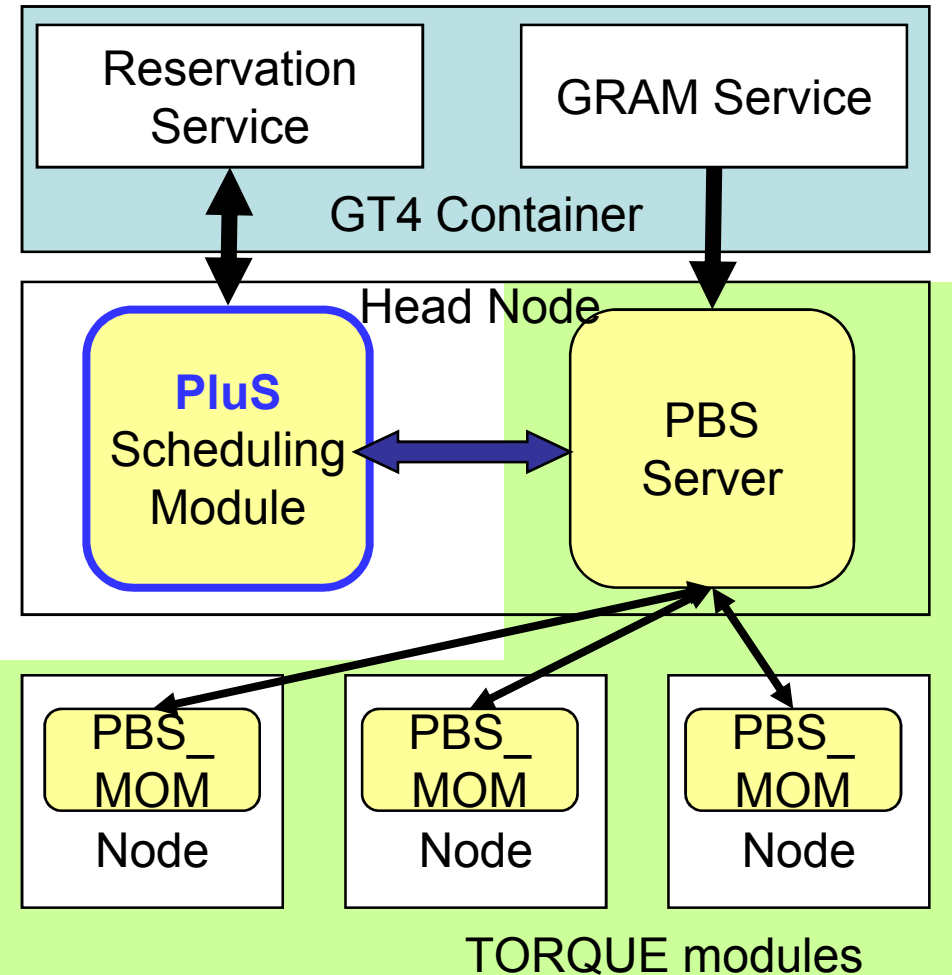
▶ 既存ローカルスケ  
ジューラ

◎ TORQUE (OpenPBS  
実装の一つ), SGE

▶ PluSローカルスケ  
ジューラプラグイン

◎ 事前予約機能を付与

◎ Javaで実装



Overview of PluS and TORQUE

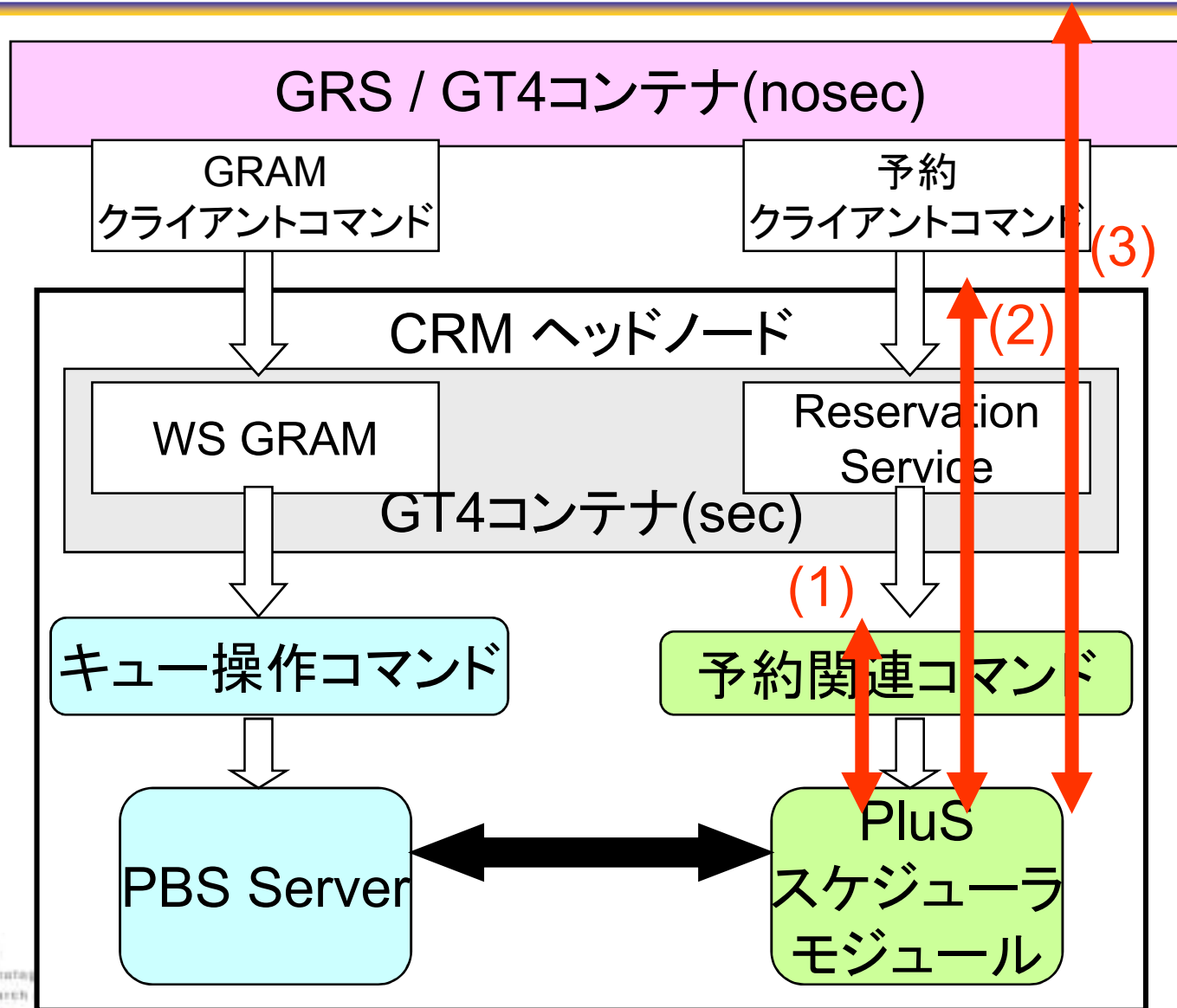
## PluSスケジューラプラグインの概要

- 既存ローカルスケジューラ(TORQUE)のスケジューラモジュールを独自実装 [HOKKE06]
  - ▶ スケジューラモジュールに予約表管理機構を導入
  - ▶ スケジューラモジュールにアクセスする予約関連コマンドを提供(RMIで通信)
    - ◎ pbs\_reserve, pbs\_rsvcancel, pbs\_rsvmodify, pbs\_rsvstatus
  - ▶ キュー操作コマンドは既存スケジューラが提供
    - ◎ qsub, qdel, qstat
- 予約関連コマンドを実行するWSRF I/Fを提供
- GT4 WS GRAM経由で予約IDを指定してジョブ投入

# 予備実験1

- CRMやNRMに対する予約処理のレイテンシがスケジューリング時間に大きく影響
  - ▶ スケジューリング時間 = ユーザがreserveを呼び出してから予約が完了するまでの時間
- GRS-CRM間の手続き(予約, キャンセル)に要する時間を測定
  - ▶ 1台の計算機内で実行
    - @ クライアント, GT4コンテナx2 (GRS, CRM), PBSサーバ
      - ⊕ GRS GT4コンテナ GSIなし
      - ⊕ CRM GT4コンテナ GSIあり(https+認証+grid\_mapfile認可)
    - @ Pentium III 1.4GHz, 2CPU, 2Gbyte, RedHat v.8

# 予備実験1



# 予備実験1の結果

(1) PluS RMI (予約関連コマンド) からの予約, 解放所要時間			
pbs_reserve	0.78 s	(1),(2)のキャンセル時間の差 0.6 = GT4 (GSIあり) のオーバヘッド	
pbs_rsvcancel	0.68 s		
(2) CRM WSRFインタフェースからの所要時間(GSIあり)			
予約手続き	1.7 s	(2),(3)CRMにおける予約とキャンセルの差 0.3-0.4 ReservationFactoryでRRを生成するオーバヘッド	
キャンセル手続き	1.3 s		
(3) GRSからの所要時間 (ユーザ-GRS間GSIなし)			
	全所要時間	CRM((2)と同じ)	WSRFオーバヘッド
予約手続き	1.9 s	1.6 s	0.27 s
キャンセル手続き	1.6 s	1.4 s	0.18 s

(3)のGT4(GSIなし)オーバヘッド 0.2

# 予備実験2



デモGUI  
(+クライアントライブラリ)

iGrid2005での  
実証実験の概要  
(ポスター発表)  
サンディエゴ

グリッド資源スケジューラ(GRS)

WSRF

- ①GUIから予約要求送信
- ②GRSがNRMと連携し  
要求を満たす資源を確保

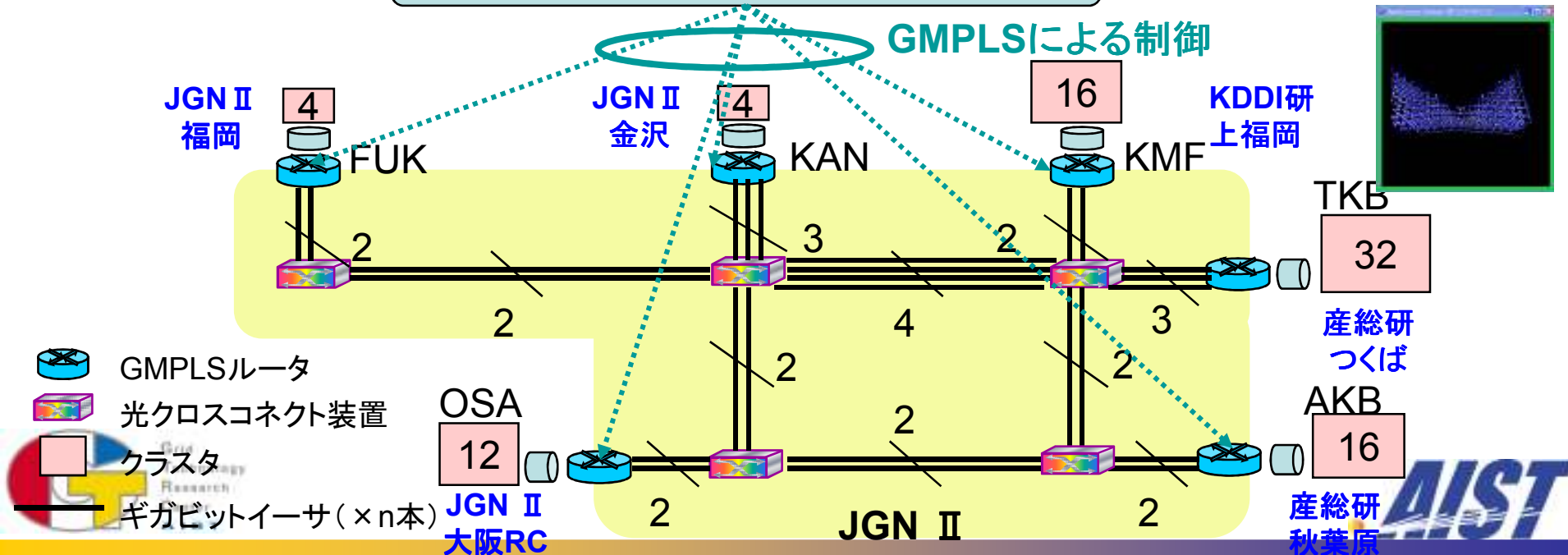
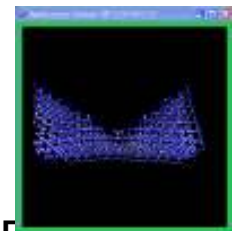
日本

ネットワーク資源管理システム(NRM)

GNS-WSI

- ③予約時刻に予約された  
資源上でアプリを実行

GMPLSによる制御



## GRSスケジューリングの課題

- ユーザの要求を満たす資源の組合せは複数通りあるが、どれが最適であるかの基準がない

- 割当方針も複数

- ➡ ▶ デッドラインまでのできるだけ早い時間に割当てて

- ▶ フェアシェア

- ▶ 課金がある場合、値段の低いものを選択

- 資源選択方法

- ➡ ▶ 資源を順次確保していく → GRS-CRM/NRM間の遅延が隠蔽不可

- ▶ 全体の情報を収集し複数予約プランを作成して複数CRM, NRMに対して同時に資源を要求 → 解の収束遅い, RMが資源情報を公開するか？

# CRMスケジューリングの課題

---

## ● 事前予約ジョブとバッチジョブの共存

- ▶ 事前予約ジョブが優先
- ▶ バッチジョブが実行中に予約時刻になる場合
  - ◎ ジョブは強制終了, キューに戻す
  - ◎ サスペンド

## ● 事前予約とプライオリティ

- ▶ First Come First Servedで事前予約
- ▶ 低プライオリティジョブを解除して高プライオリティジョブを入れることができない

→ 予約解除をGRSに伝える方法やSLAが必要

# 関連研究

---

## ● VIOLA

- ▶ グリッドで計算・ネットワーク資源を同時に提供
- ▶ UNICOREベースグリッド環境でWS-Negotiation/-Agreementに基づくメタスケジューラを開発中

## ● グリッドスーパースケジューラ

- ▶ Silver, CSF, NAREGIスーパースケジューラ
- ▶ 計算資源の割当てが主体
- ▶ ネットワーク資源も含めたコアロケーションは実現していない

## まとめ

- 計算・ネットワーク資源を事前予約により同時確保するグリッドスケジューリングシステムを開発
  - ▶ グリッド資源スケジューラ(GRS)と計算資源マネージャ (CRM)からなる
  - ▶ WSRFベースインタフェースを提供
  - ▶ CRMでは既存ローカルスケジューラ(TORQUE)に事前予約機能を提供するPluSプラグインを開発
- ネットワークキャリアの提供するネットワーク資源管理システム(NRM)とGNS-WSIを介して連携

# 今後の課題

---

## ● GRS, CRMの改良

- ▶ GNS-WSI ver. 2インタフェースを実装した複数NRMとの連携
- ▶ 分散トランザクションのため, 2フェーズコミットプロトコルのサポート
- ▶ 予約インタフェースの統一(GRS, CRM, NRM)

## ● GRS, CRMのスケジューリングアルゴリズムの改良

## ● 認可, SLA, 課金

# 謝辞

---

- **G-lambdaプロジェクトの皆様に感謝いたします**
  - ▶ <http://www.g-lambda.net/>
- **本研究の一部は，文部科学省科学技術振興調整費「グリッド技術による光パス網提供方式の開発」による**