グローバルスケジューリングのための計算資源予約管理機構

1.産業技術総合研究所、2.数理技研

中田秀基[1]、竹房 あつ子[1]，大久保 克彦[1,2]、

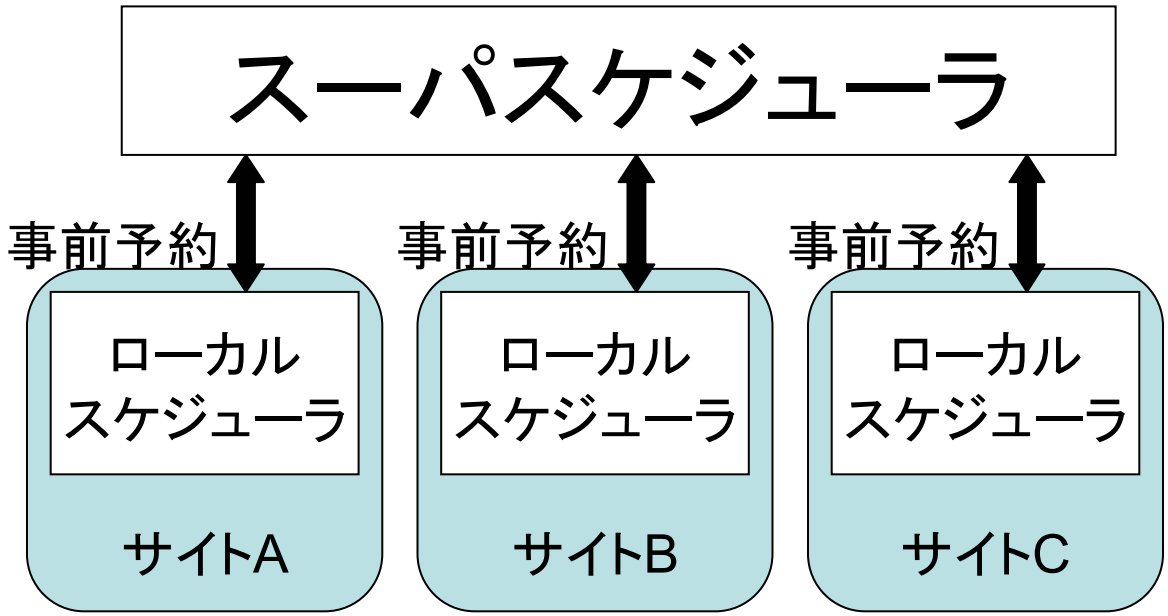
工藤 知宏[1]、田中良夫[1]、関口智嗣[1]

Grid Technology Research Center AIST

AIST

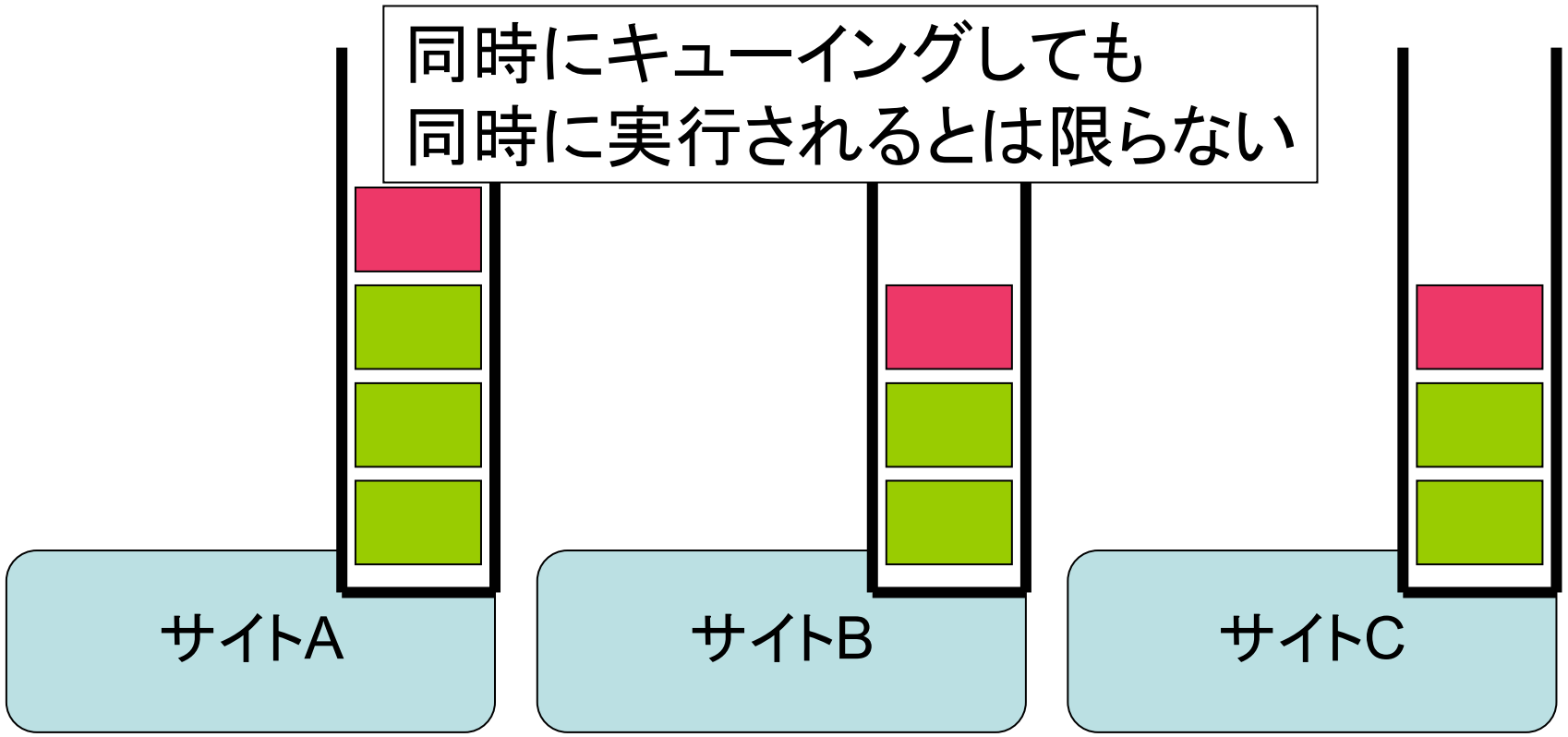
背景

- グリッドによる複数のサイトをまたいだ大規模計算
  - 資源のコアロケーション(同時確保)が必要
- 多くのサイトはFCFS(First Comes First Served)＋優先順位のキューイングシステムで運用



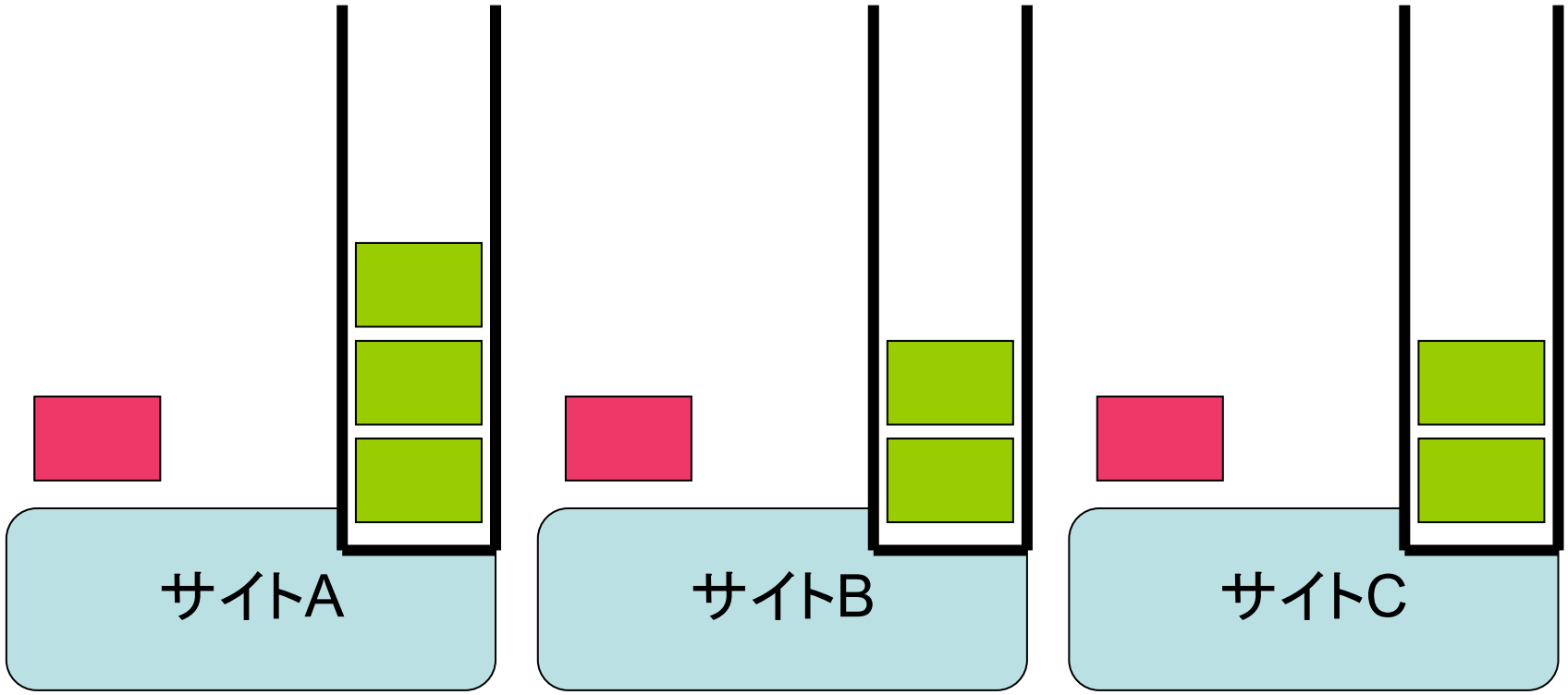
計算資源のコアロケーション (1/2)

- FCFS
  - キューイングされた順番で実行

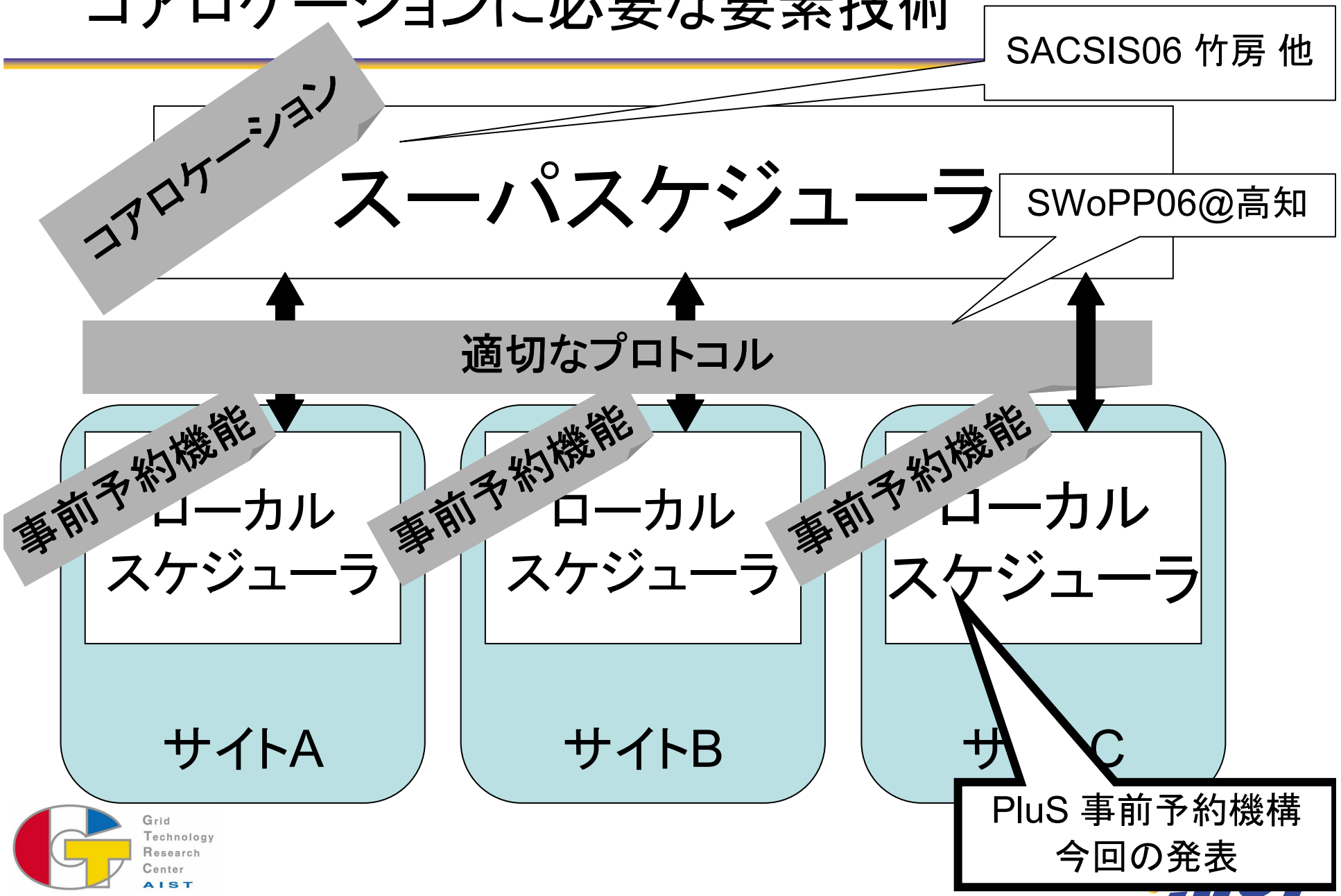


計算資源のコアロケーション (2/2)

- 事前予約
  - キューとは独立に時間スロットを確保



コアロケーションに必要な要素技術



Grid Technology Research Center AIST

研究の目的

- ローカル予約管理機構 PluSの設計と実装
  - 予約機構を既存キューイングシステムに追加
    - 2つの実装手法を提案，比較
      - スケジューラ置換法
      - キュー操作法
  - 2相コミットプロトコルに対応

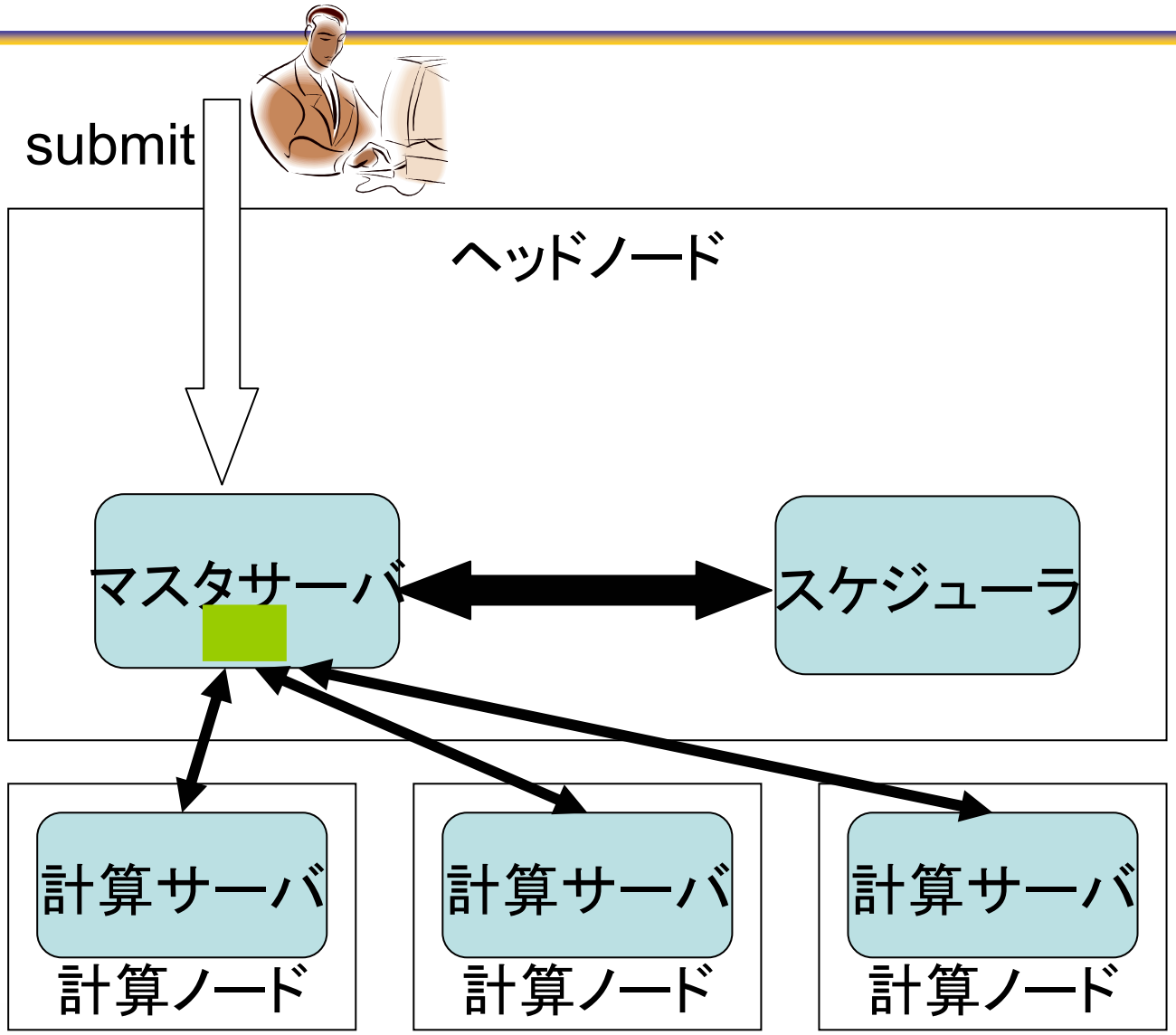
発表の概要

- 事前予約機構 PluSの設計
  - キューイングシステムの概要
  - 2つの実装法に基づく実装
  - 2相コミット
- PluSの実装
  - for TORQUE
  - for Grid Engine
- 評価：2つの実装法を比較
  - コード量の比較
  - 実行時間を比較
- おわりに

キューイングシステム とは

- 計算機資源へのジョブ投入を管理
  - 計算機資源を個々のジョブが占有することが前提
    - c.f. time share
  - 課金のための統計情報も管理
  - 大規模な計算資源ではほぼ例外なく利用されている
- 代表的な有償実装
  - LSF, NQS, PBS Professional, LoadLeveler
- 代表的なオープンソース実装
  - TORQUE　– OpenPBS ベース, Cluster Resources Inc.
  - Grid Engine – Sun Microsystems.

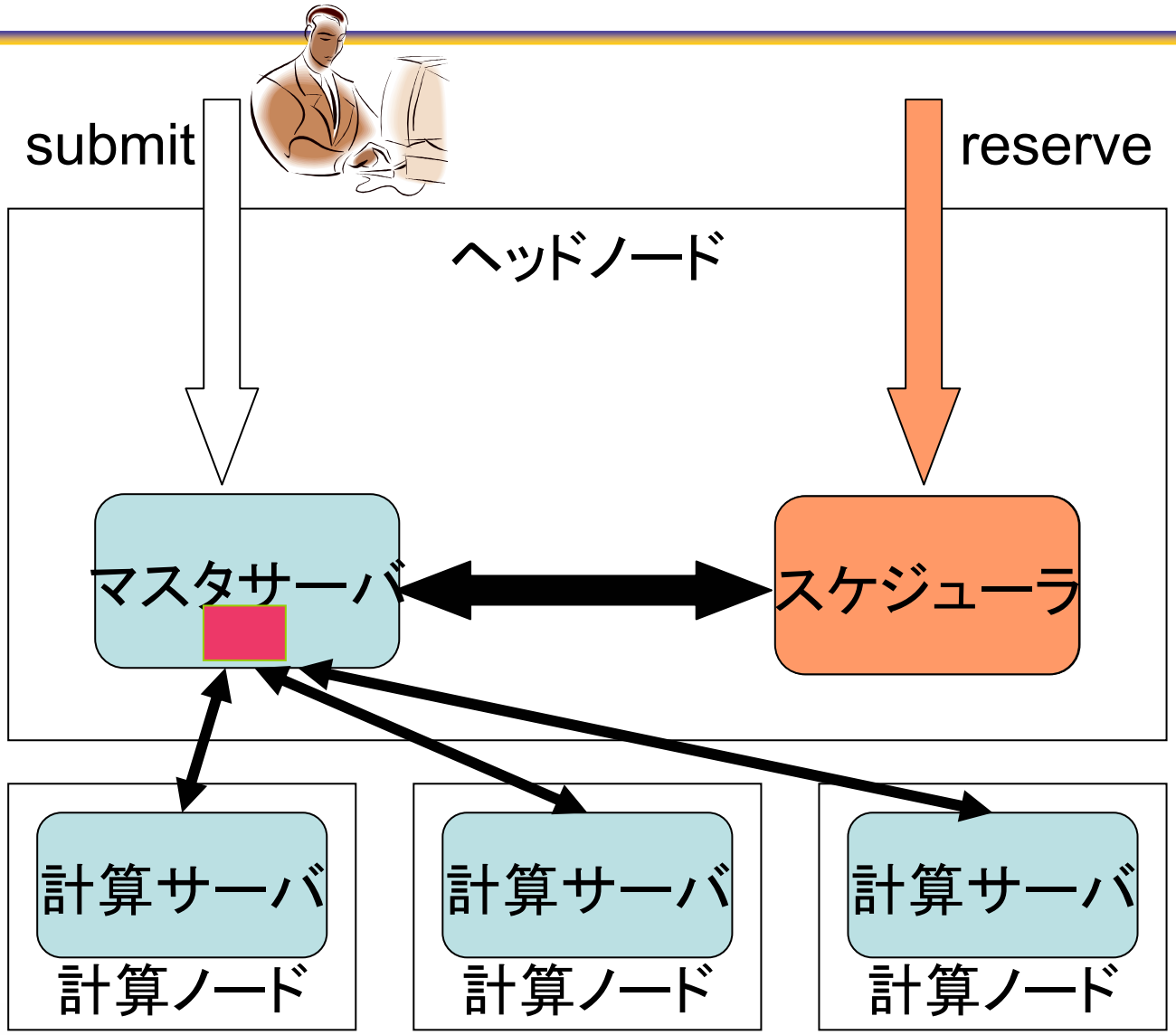
キューイングシステムの標準的な構成



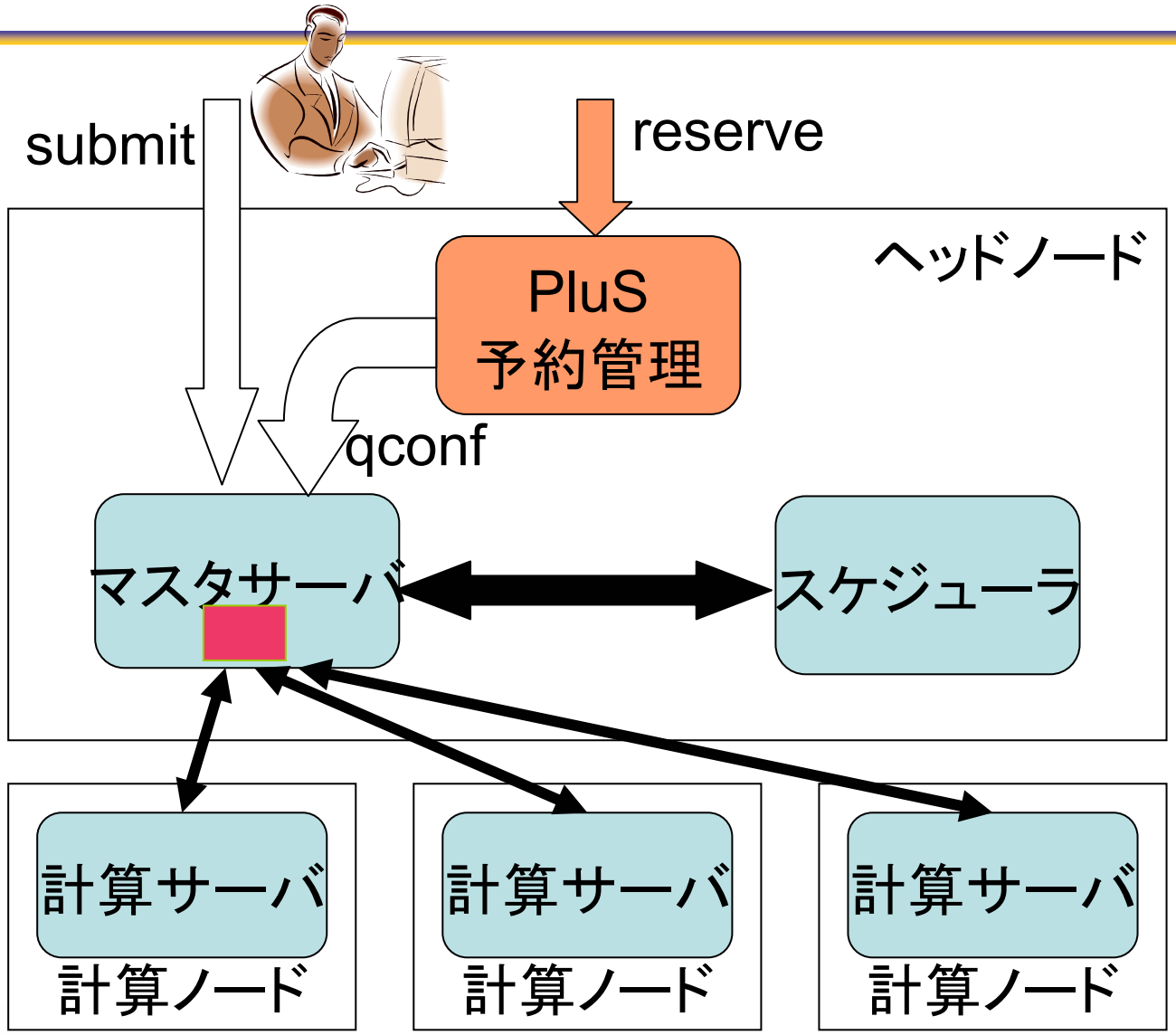
事前予約の実現方法

- スケジューラを置き換える
  - マスタサーバとのプロトコルが単純であればむしろ簡単
- スケジューラをいじらずに追加だけで済ませる
  - 「キュー」を制御
  - キューイングシステムの構造によっては，可能とは限らない

スケジューラを置換する事前予約の実装



キューを外部から制御する事前予約の実装



2つの実装方法

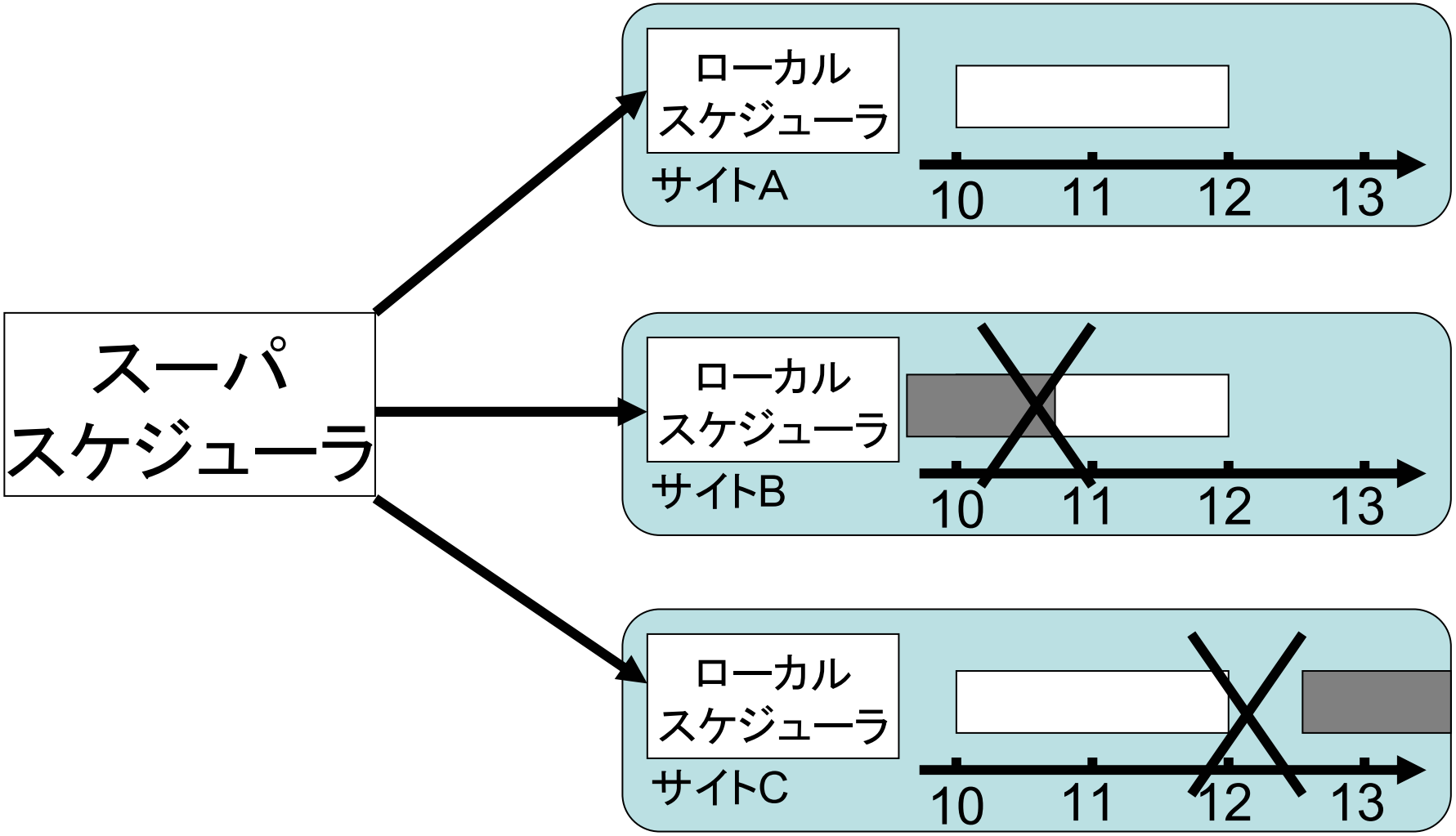
TORQUE
Grid Engine

- スケジューラ置換法
  - 心臓部であるスケジューラを置換してしまうので，なんでもできる
  - 既存のスケジューラとの互換性を保つ場合には，実装コストが大きい
- キュー操作法

Grid Engine

  - キューイングシステム[illegible]現不可能
    - ex. TORQUE
  - キュー操作のオーバヘッドが懸念される
  - 実現できる機能の自由度は低い
  - 実現可能な場合には実装コストは比較的小さい

コミットプロトコルの必要性



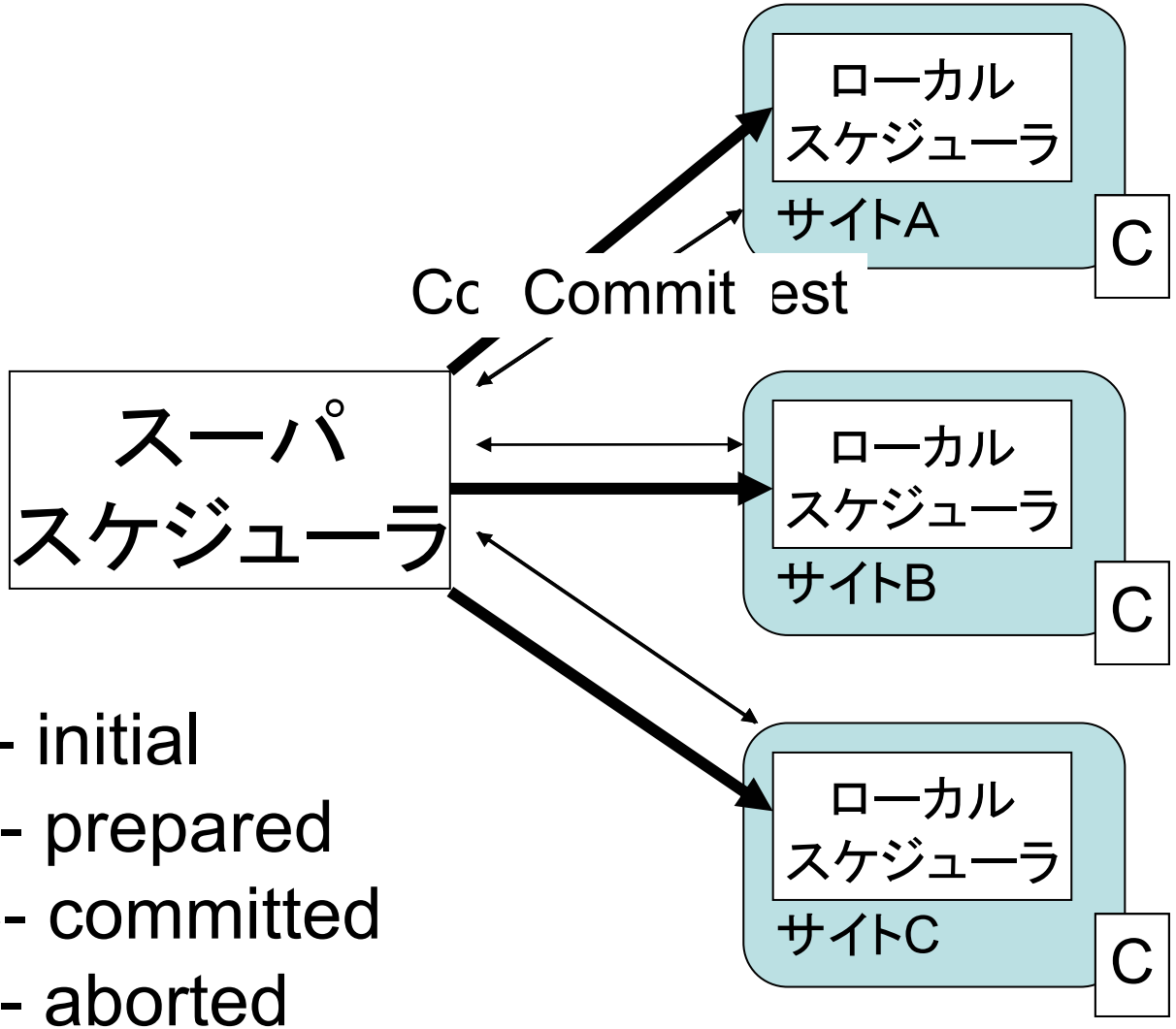
コミットプロトコル

- 分散コアロケーションは本質的に分散トランザクション
  - 同じ手法で解決が可能
  - 2相コミット，3相コミット などなど
- 2相コミットを採用
  - シンプル
  - 実装が比較的容易

2相コミットプロトコル

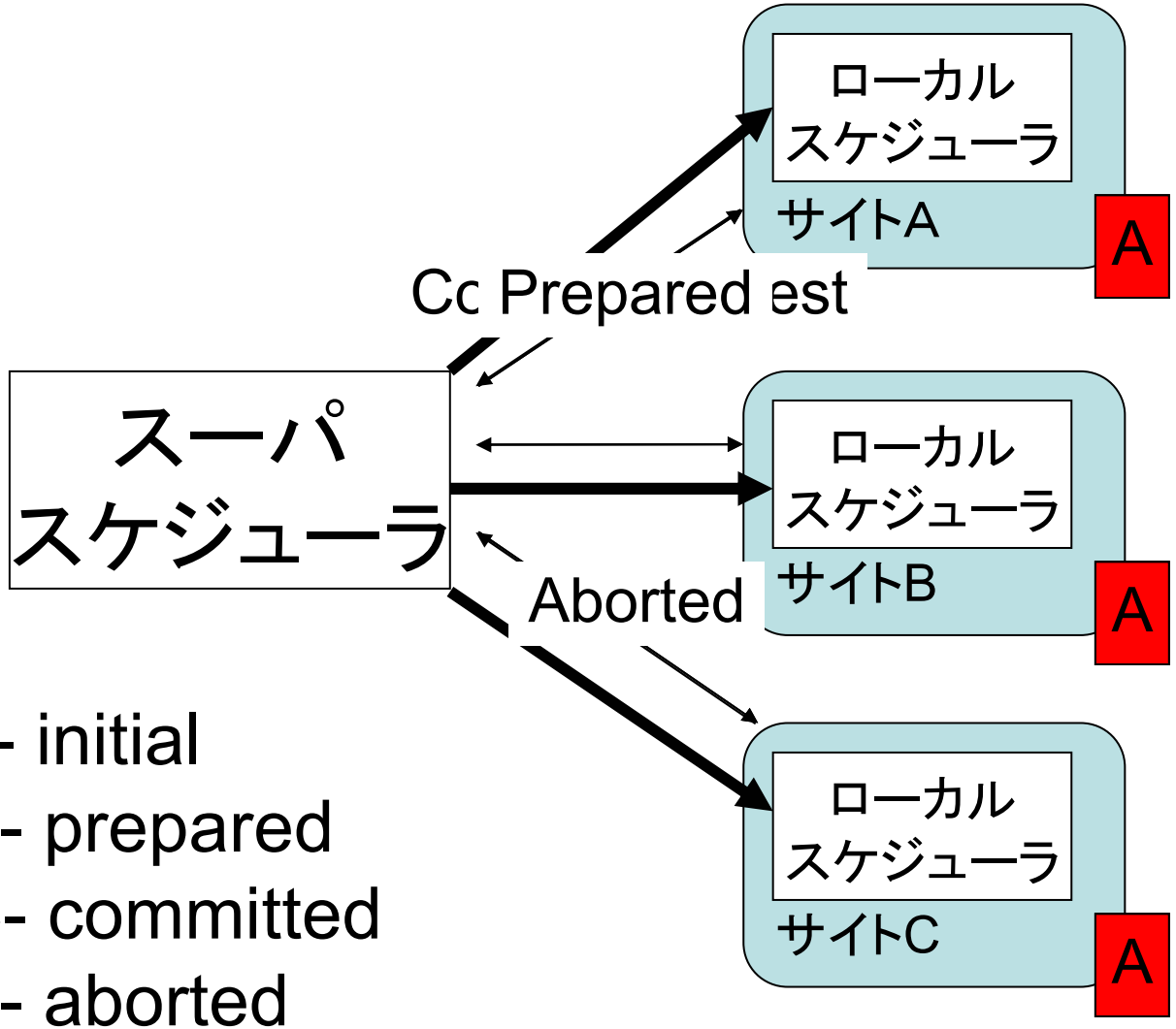
- 1度で決めてしまわないで，確認するフェイズを設ける
- 1相コミット
  - 「予約できたらしてください」
  - 「はい，予約しました」
- 2相コミット
  - 「この時間は空いていますか？空いていたら仮押さえしておいてください．」
  - 「はい仮押さえしました．」 1相
  - 「正式に予約をおねがいします．」 2相

2相コミット



- I - initial
- P- prepared
- C- committed
- A- aborted

2相コミット



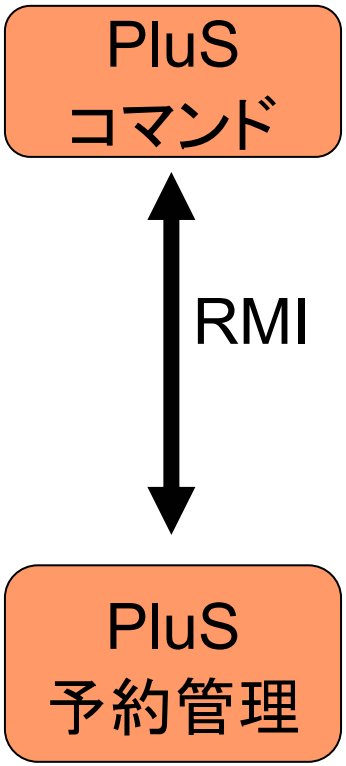
- I - initial
- P- prepared
- C- committed
- A- aborted

PluSの実装

- 3種類の実装
  - TORQUE 向け スケジューラ置換法
  - Grid Engine 向け スケジューラ置換法
  - Grid Engine 向け キュー制御法

PluS予約管理機構の実装

- PluSモジュールはJavaで実装
  - データベースにはdb4objectを使用
- コマンドラインインターフェイスはshell script + Java
- インターフェイスPluSモジュール間の通信はRMI



予約関連コマンド

- plus_reserve
  - 予約をリクエスト
  - 入力：開始・終了時刻,ノード数
  - 出力：予約ID
- plus_cancel
  - 予約キャンセル
  - 入力: 予約ID
- plus_status
  - 予約情報取得
  - 入力： 予約ID
  - 出力： 予約状態
- plus_modify
  - 予約変更
  - 入力： 予約 ID, 開始・終了時刻,ノード数

使用シナリオ

- まず予約

```
> plus_reserve -s 12:00 -e 14:00 -n 1
Reserve succeeded: reservation id is 14
```

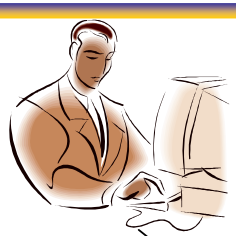
- 予約状態の確認

```
> plus_status
 id owner          start              end      duration state
 R14 nakada     Feb 20 12:00   Feb 20 14:00     2h00m Confirmed
```

- 予約ID を指定してジョブをサブミット

```
> qsub -q R14 script
```

スケジューラ置き換えによる事前予約の実現



予約リクエスト

PluS
予約管理機能付
スケジューラ

マスタサーバ

計算サーバ

計算サーバ

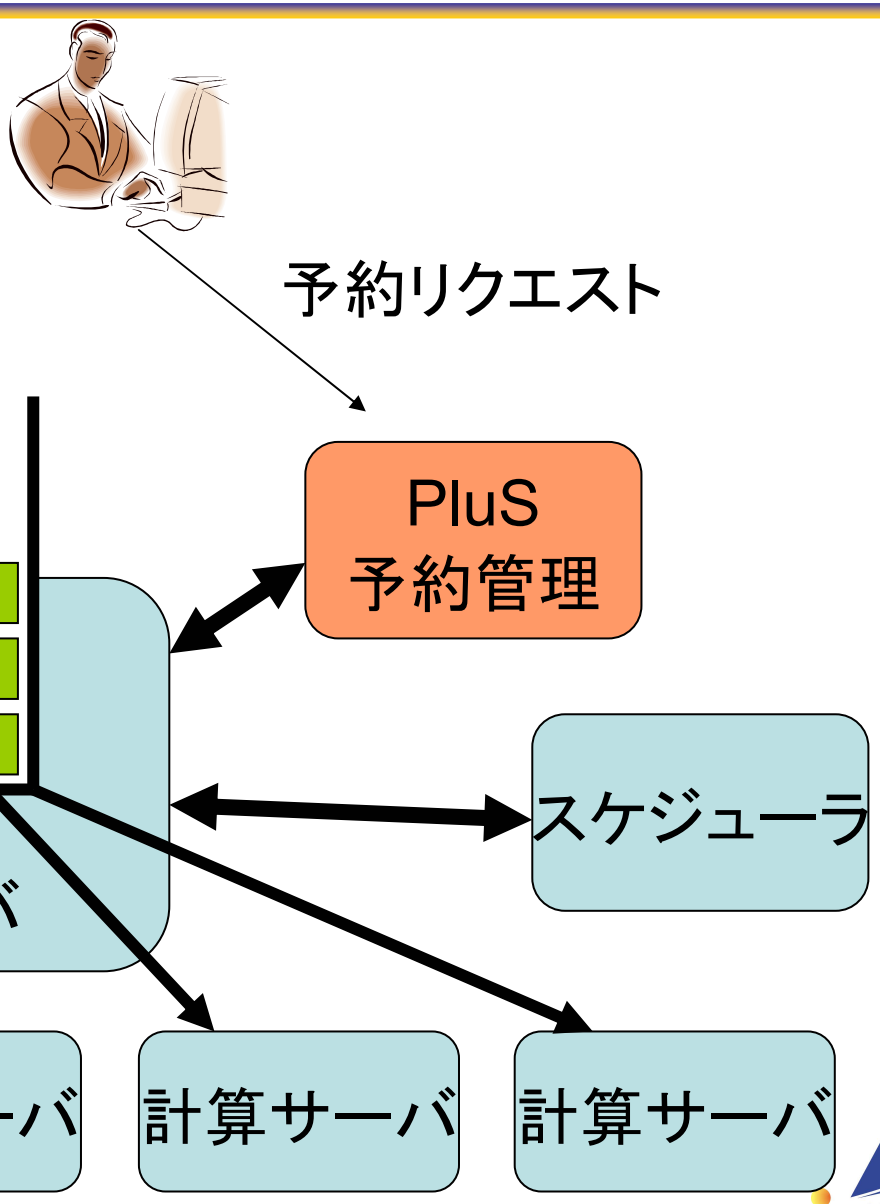
計算サーバ

計算サーバ

キュー制御による事前予約

- キューとは
  - ジョブをサブミットする抽象的な「口」
  - 特定のユーザ群に対して定義される
  - 特定のノード群に割り当てることもできる
- キュー制御による事前予約
  - 事前予約をキューとして作成
  - キューを特定の時間だけ排他的に使用可能するように制御することで事前予約を実現
- 実装法の特徴
  - ○実装が容易
  - ○内部プロトコルに手を触れないため，本体のバージョンアップへの対応が容易
  - ×キュー操作の外部コマンドを何度も呼び出すことになるので低速

キュー制御による事前予約



2相コミットの実現

- 予約時
  - リクエスト時に予約
  - アボートしたら予約をキャンセル
- キャンセル時
  - リクエスト時にはなにもしない
  - コミット時にキャンセル
- モディファイ時
  - リクエスト時にモディファイ前の資源とモディファイ後の資源の双方を確保
  - アボート・コミットに応じて不要な資源を解放

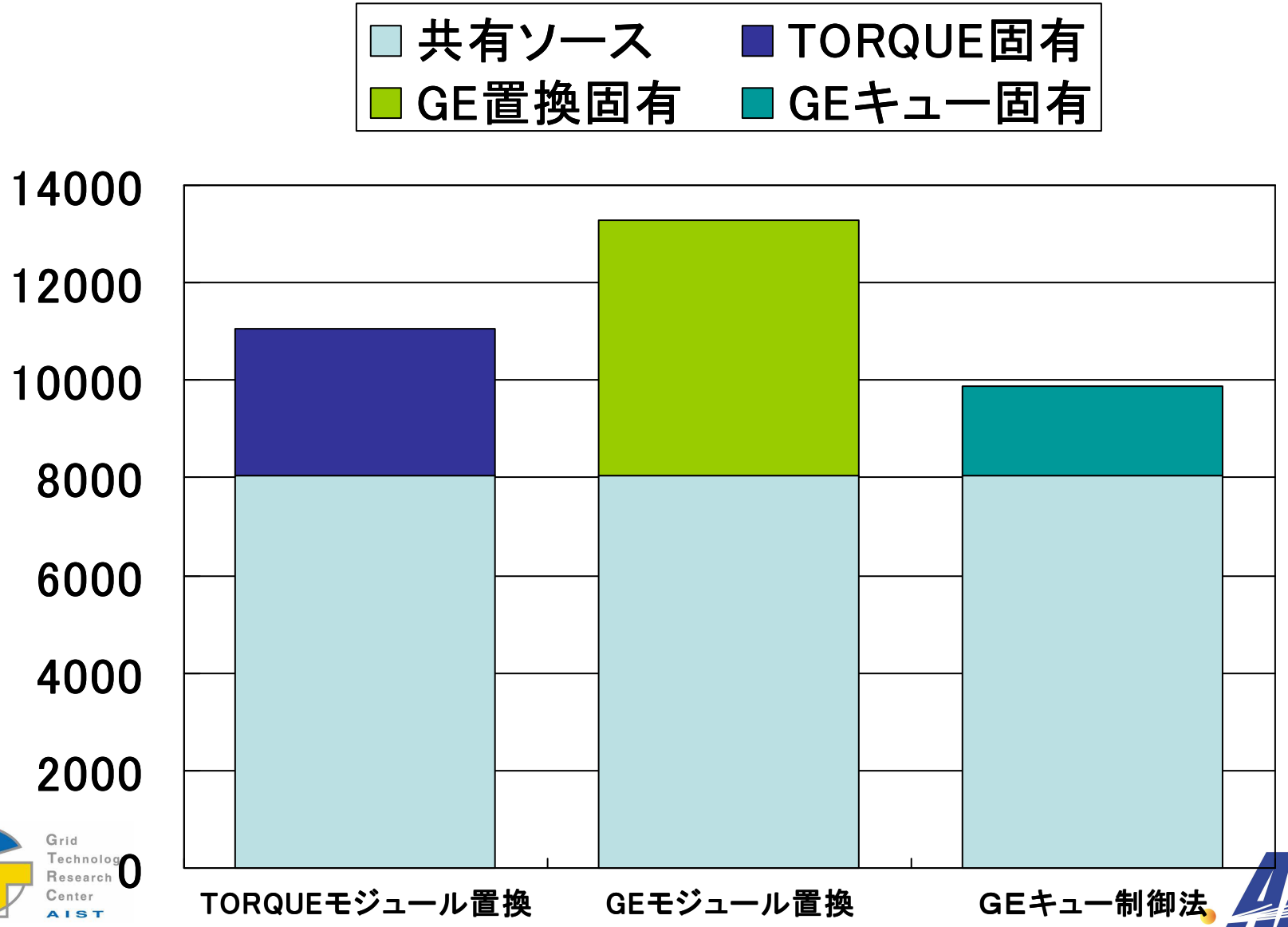
コマンドの2相コミットへの対応

- plus_reserve, plus_modify, plus_cancel
  - オプション ‘-T’ を追加
    - Two-phased を示す
  - -T がなければ1相コミット
- plus_commit, plus_abort を追加
  - 予約IDに対して commit, abort を行う

評価

- 実装の容易性
  - キュー制御法のほうが本当に容易か？
  - コードの量を行数で比較
- 実装の効率
  - キュー制御にかかるコストは？
    - あまり遅いと上位スケジューラのレイテンシに悪影響
  - 予約・キャンセルコマンドの実行時間を比較

コード行数による評価


共有ソース
TORQUE固有
GE置換固有
GEキュー固有
14000
12000
10000
8000
6000
4000
2000
0
TORQUEモジュール置換
GEモジュール置換
GEキュー制御法

コマンド実行時間による比較

- 実験環境
  - Pentium III 1.4 G
  - Memory 2G byte
  - Linux RedHat 8
- 測定
  - time を用いてコマン 行
  - 10回試行，平均値 大値，

キュー制御法のほうが遅い
→キュー操作(qconf)のコスト

実行時間は1-2秒程度
→ この程度なら許容範囲

	予約				キャンセル			
	平均	分散	最小	最大	平均	分散	最小	最大
スケジューラ置換法	1.02	0.04	0.91	1.54	0.92	0.00	0.85	1.03
キュー制御法	1.95	0.02	1.76	2.25	1.02	0.00	0.97	1.11

関連研究

- Maui
  - Cluster Resources 社が無料で提供
  - TORQUEのスケジューラを置き換えるスケジューラ
- Catalina [Yoshimoto 05]
  - SDSCで利用されているスケジューラ
  - Python で実装
  - TORQUEのスケジューラを置換して事前予約機能を提供
  - すべてのジョブが予約ベースで稼動
    - 事前予約は他のジョブがないときのみ可能

おわりに

- 事前予約管理機構PluS
  - スケジューラ置換法とキュー操作法の2種を実装
  - TORQUEとGrid Engine の双方に対応

  - スケジューラ置換法は自由度が高いが実装が煩雑
  - キュー操作法は実装は容易だが，自由度が低く，オーバヘッドもある
    - オーバヘッドは許容範囲内.

今後の課題

- 事前予約のポリシ
  - 現状：事前予約が常に最優先
    - 実システムでの運用には不適
  - 事前予約許可のポリシを宣言的な言語で設定
    - CondorのclassAd を検討
    - 各サイト管理者が独自に設定
- 他のキューイングシステムへの適用
  - キュー操作法は理論的には可搬性に優れる
  - 実際に移植して確認
    - Load Leveler
    - Condor

謝辞

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.

http://www.g-lambda.net/plus