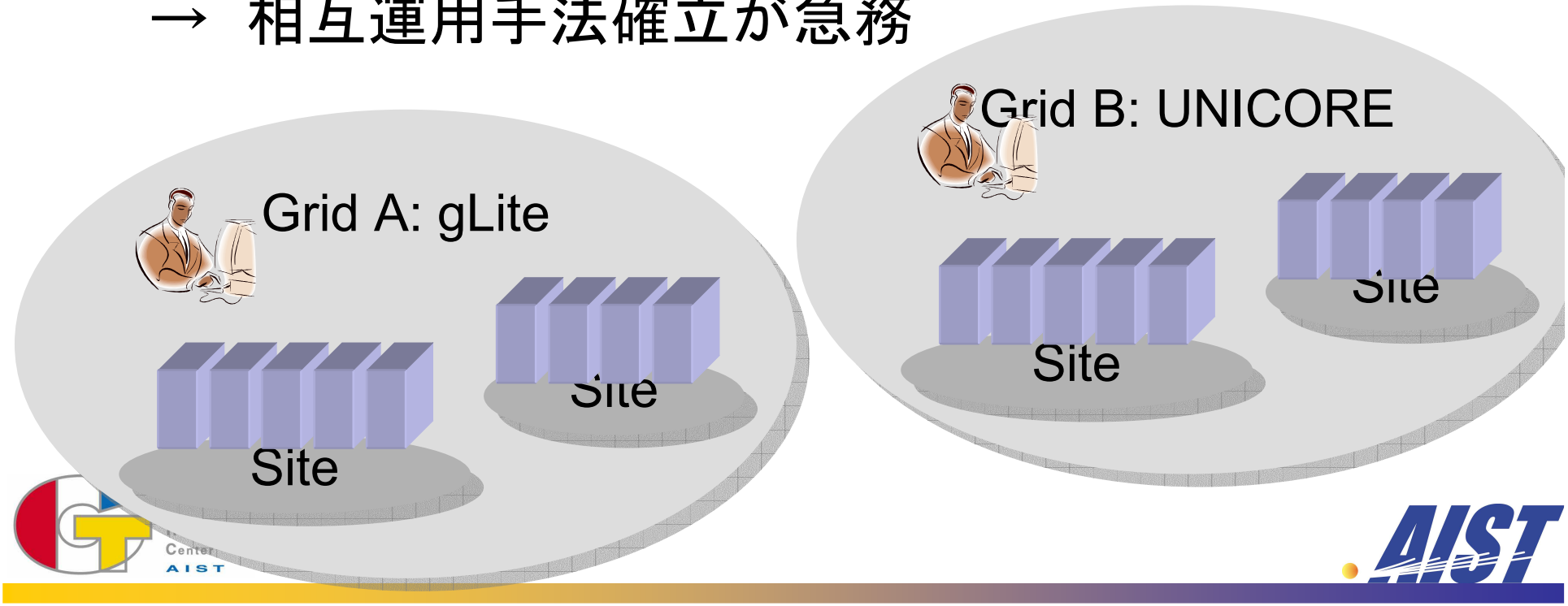
NAREGIミドルウェアβ - gLite間における相互ジョブ起動実験

中田 秀基 (産総研), 佐藤 仁(東工大),
佐賀 一繁 (NII), 畑中 正行 (富士通),
佐伯 裕治 (NII), 松岡 聡 (東工大, NII)

背景

- 様々なグリッドミドルウェアの発展
  - Globus, UNICORE, NAREGI Middleware, gLite
  - 実際の運用も始まる
  - 異なるグリッドミドルウェアで運用されているグリッド間では、リソースを共有することができない

→ 相互運用手法確立が急務



背景 (2)

- OGF(Open Grid Forum) GIN-CG
  - Grid Interoperation Now Community Group
  - 標準化への努力はさておき、現在可能な技術で相互運用を行おうという試み

目的

- GIN-CGの一環として、下記のふたつのグリッドミドルウェア間での相互運用実験を行う
  - NAREGI Middleware β
  - EGEE gLite

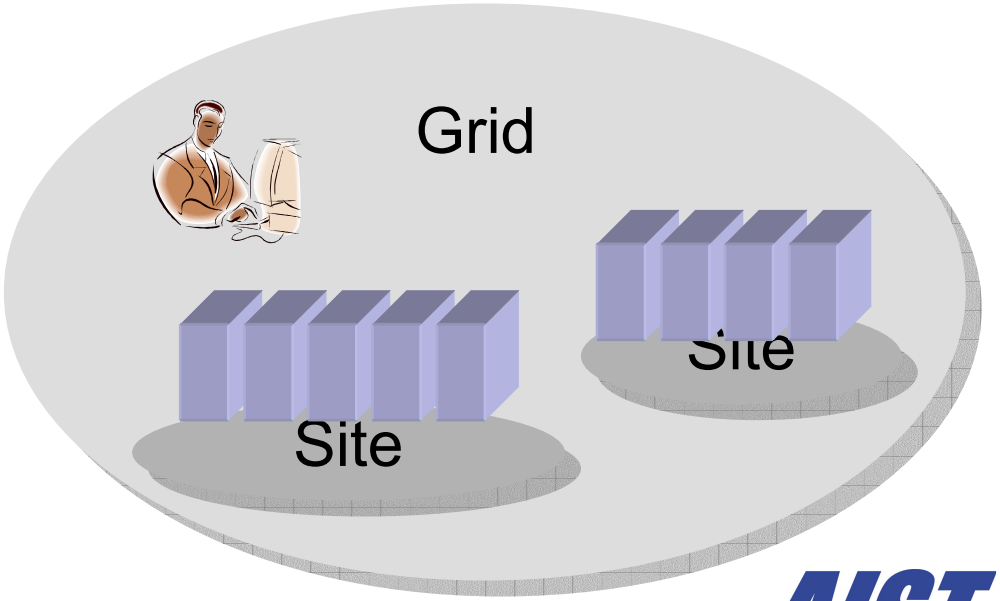
- 相互運用
  - セキュリティ機構
  - 情報サービス
  - ジョブサブミッション
  - 大規模データ転送

発表のアウトライン

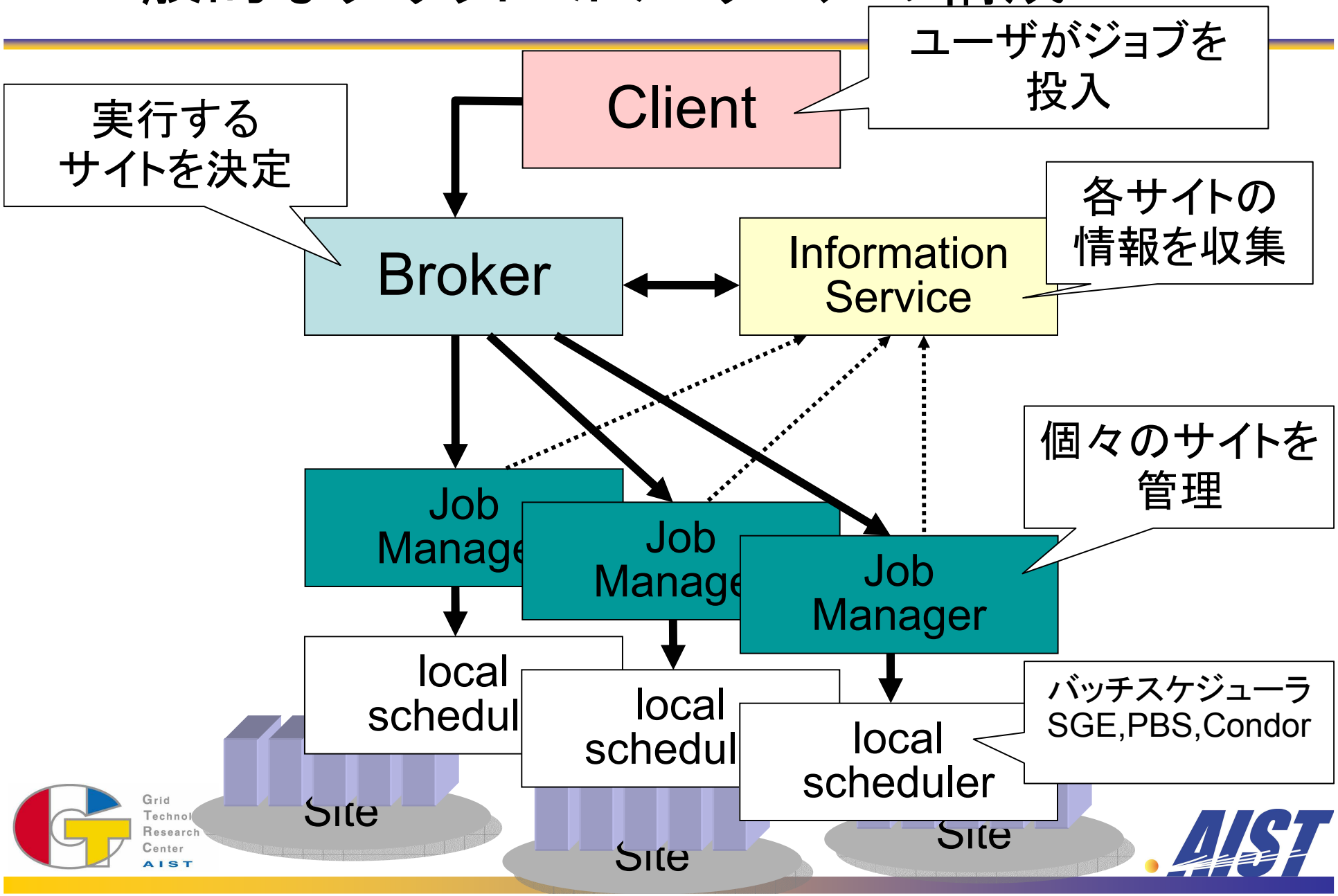
- グリッドミドルウェアの構造
  - NAREGI Middleware β
  - gLite
- 相互運用の方針と実現
- 測定
- 結論

グリッドミドルウェアの役割

- 前提
  - 個々の“グリッド”には複数の“サイト”が属している
  - 個々の“サイト”には複数の計算機があり，なんらかの “ローカルスケジューラ” で管理されている
- グリッドミドルウェア
  - ユーザからのジョブ実行依頼を「適切」なサイトへ「安全」にディスパッチ
    - 適切 – 負荷分散，VOMなど
    - 安全 – 認証・認可
  - サイト内部ではローカルスケジューラが実行を司る



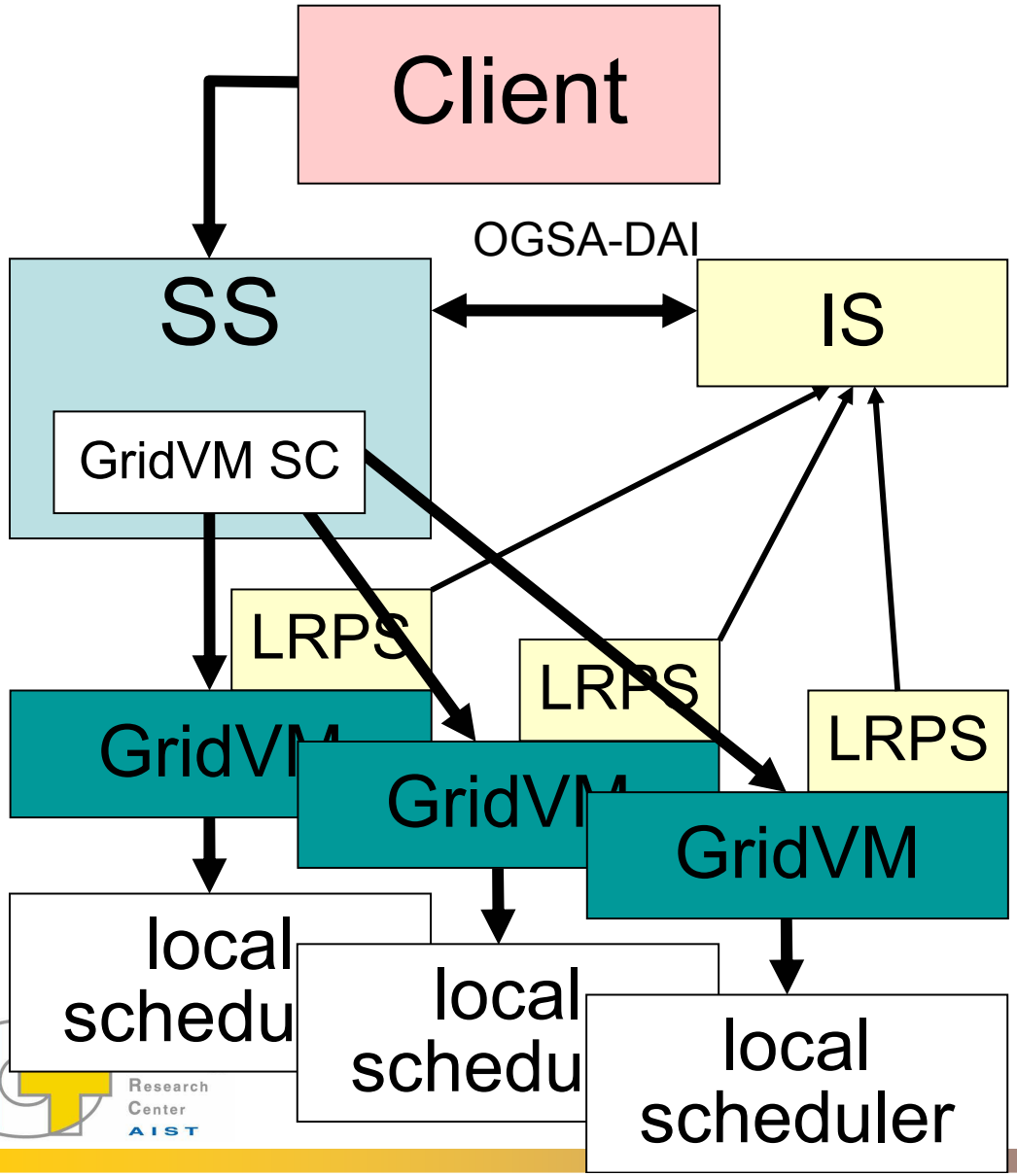
一般的なグリッドミドルウェアの構成



NAREGI ミドルウェア β

- NAREGIプロジェクトの開発したミドルウェア第2世代
  - β という名前に反して，α とは（ほとんど）無関係
    - α : 2004年度開発
      - UNICORE ベース
    - β : 2005年度以降開発
      - WSRFをベース
      - OGFで定められた標準技術に準拠
- 特徴
  - ワークフローエンジン
  - 複数サイトにまたがった並列ジョブ実行
    - 自動的に複数サイトへ分配

NAREGI ミドルウェア β の概要

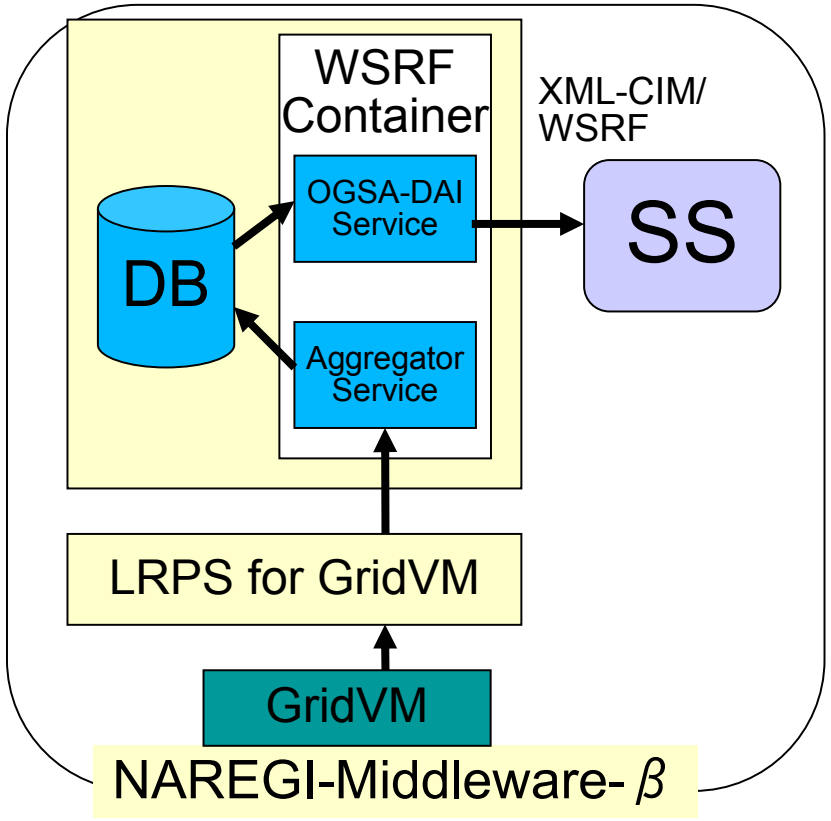


- SS (Super Scheduler)
  - ブローカー兼ワークフローエンジン
- IS (Information Server)
  - 情報収集機構
  - DBをOGSA-DAIでラップ
- GridVM
  - クラスタを管理
  - いわゆる”VM”ではない
  - GT4 のコンテナを利用

Research Center AIST

AIST

NAREGI ミドルウェア情報サービスの概要

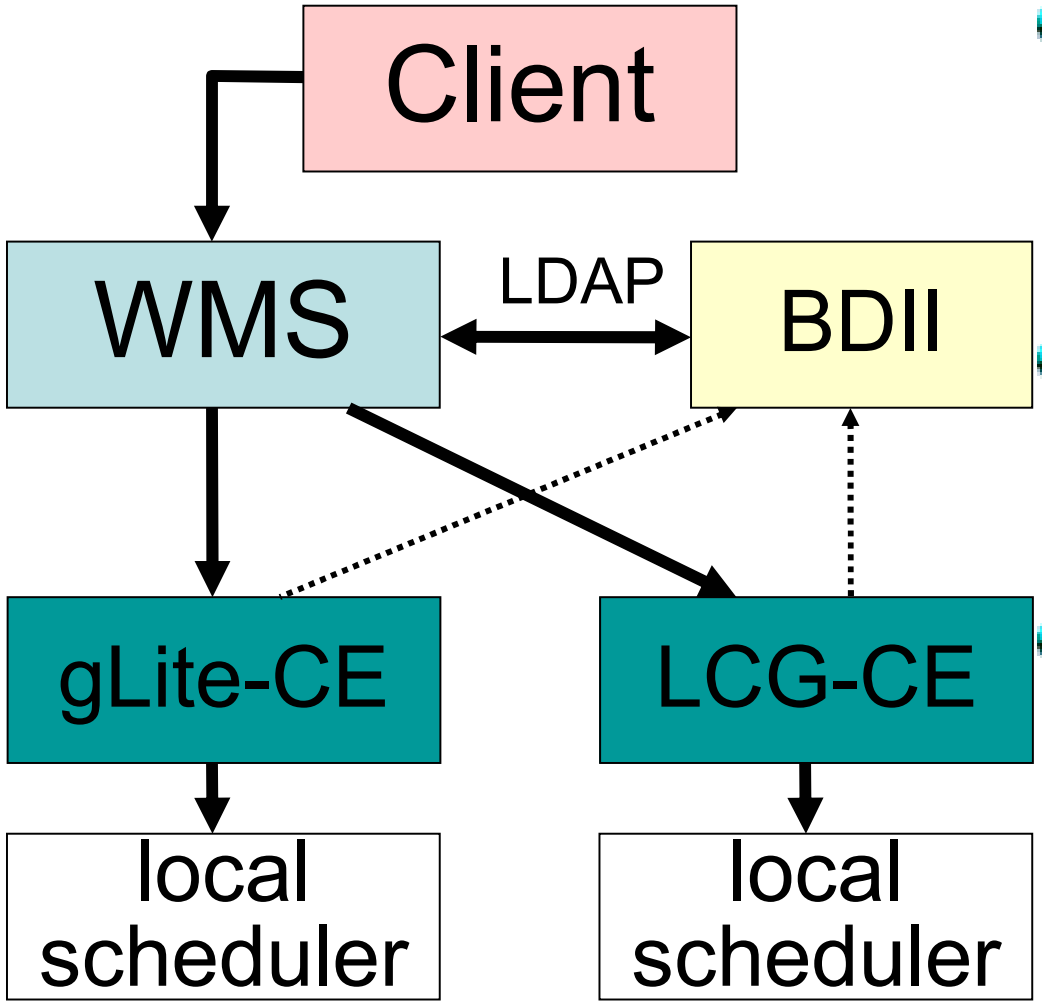


- 情報はCIMスキーマで表現，
  - DBに格納
- 情報の収集は，LRPS(Local Resource Provider Service)から，Aggregator Serviceを経由して行う
- 情報の検索はOGSA-DAIで
  - WSRFでラップされたデータベース検索インターフェイス

gLiteの概要

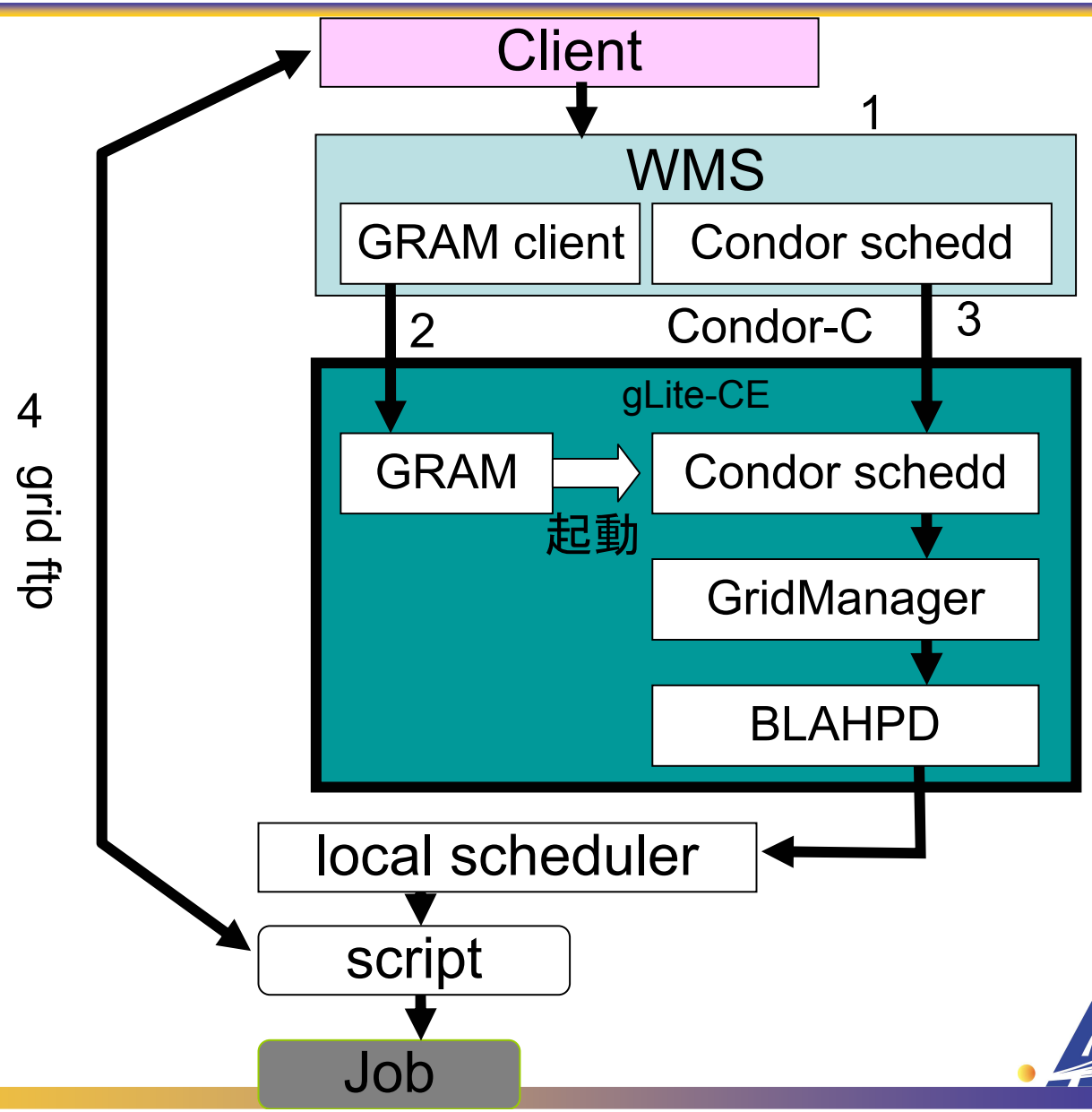
- EUのEGEE (Enabling Grids for E-Science in Europe) のミドルウェア
- Condorのモジュールをあちこちで使用
  - Condor
    - Wisconsin大学で開発された，スケジューリングシステム
  - Condorで用いられているmatch making をブローカリングに使用
  - ジョブサブミッションはCondor-C

gLite の概要



- WMS
  - Workload Management System
  - classad をもちいたフローカリング
- BDII (Berkley Directory Information Index)
  - LDAP ベース
- CE (Compute Element)
  - gLite-CE
    - Condor-Cを使用
  - LCG-CE
    - Globus GRAM2
    - 先行プロジェクトLCG (LHC Computing Grid) からのキャリーオーバー

gLite-CE におけるジョブ実行の詳細



Grid Technology Research Center AIST

AIST

発表のアウトライン

- グリッドミドルウェアの構造
  - NAREGI Middleware β
  - gLite
- **相互運用の方針と実現**
- 測定
- 結論

相互ジョブ実行の要件

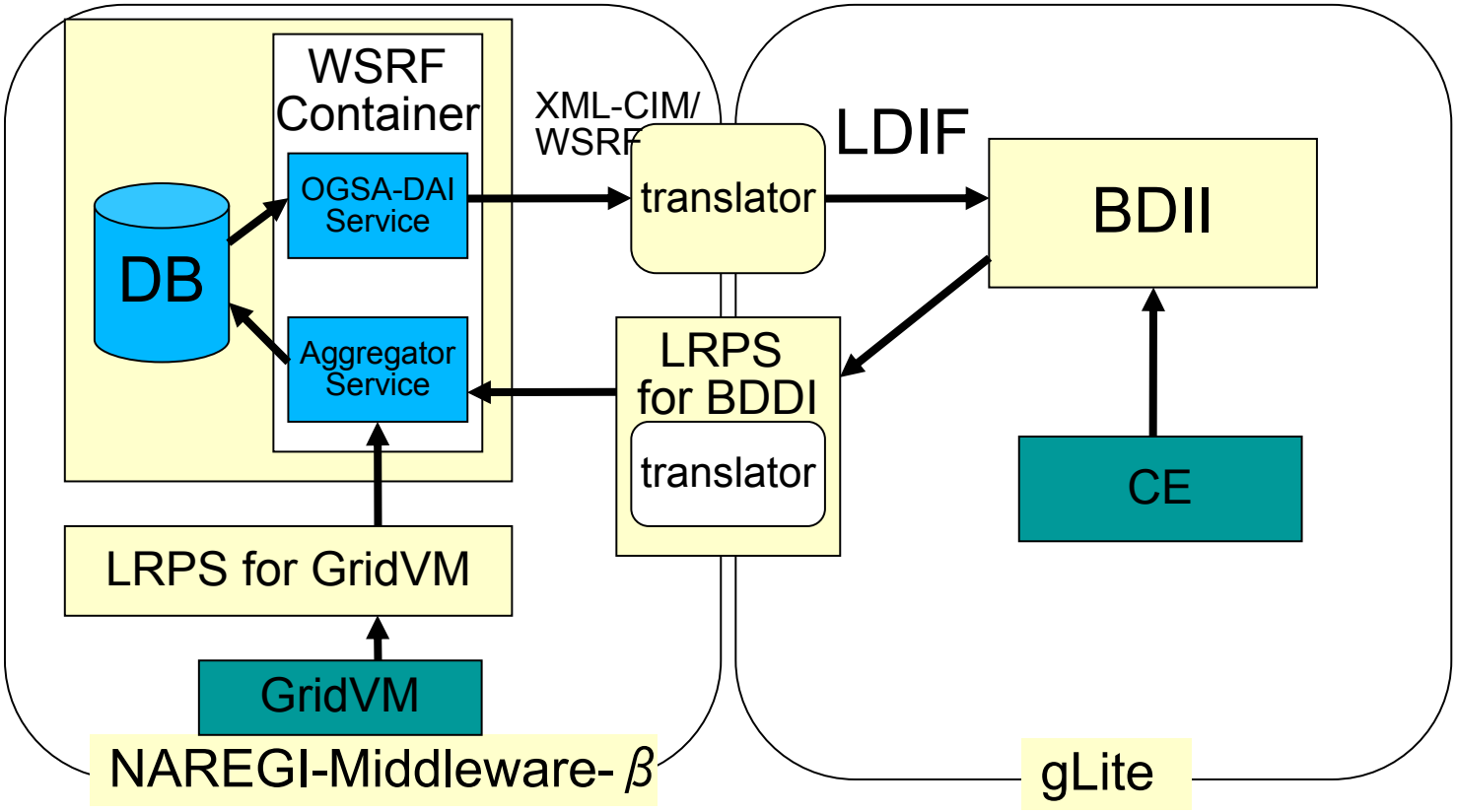
- 認証・認可基盤の相互運用
  - セキュリティ基盤
  - すべての基礎 – この部分の相互運用は必須
- 情報サービスの相互運用
  - 相互の計算機資源の状態が検索可能
- ジョブの相互投入
  - どこで相互乗り入れするかが問題

認証，認可基盤の相互運用

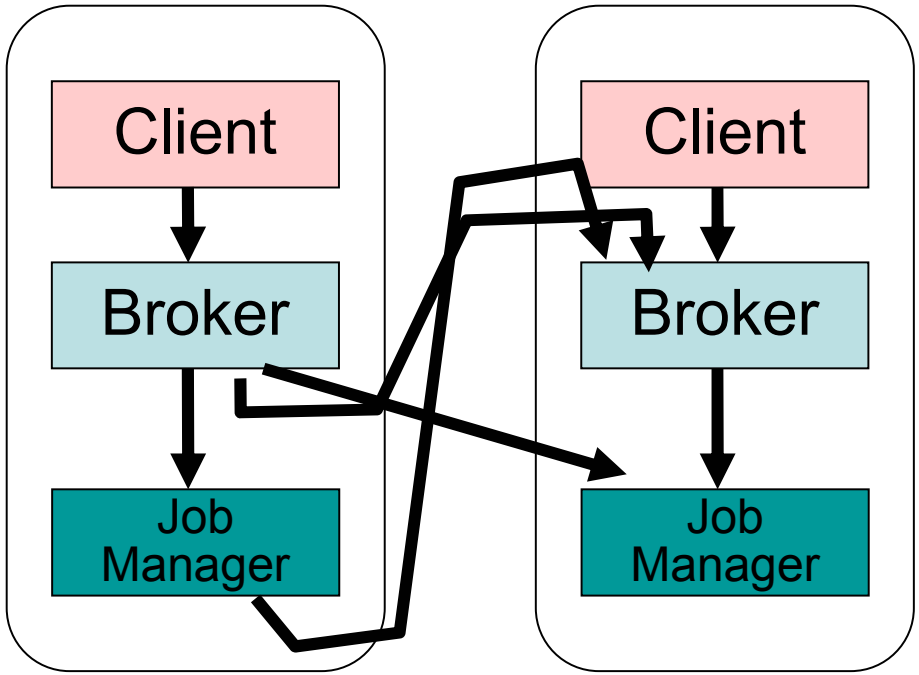
- 認証
  - ユーザが誰であるかを確認
    - 一般に証明書を用いる
- 認可
  - そのユーザに対してなにを許可するか
  - 仮想組織の管理

- すべての基礎 – この部分の相互運用は必須
  - 今回はほとんど問題にならなかった
  - 認証基盤 GSI
  - 仮想組織管理 VOMS
    - NAREGIがEGEEのVOMSを使用

情報サービスの相互運用



相互ジョブ起動の3つの方法



- Broker -> JobManager
  - 比較的速い
  - ジョブ受け入れ側グリッドの管理ポリシが反映しにくい
  - 情報サービスの相互運用が必須
- Broker -> Broker
  - 比較的遅い
  - 受け入れ側の管理ポリシが反映しやすい
- JobManager -> Broker
  - さらに遅い
  - 受け入れ側の管理ポリシが反映しやすい

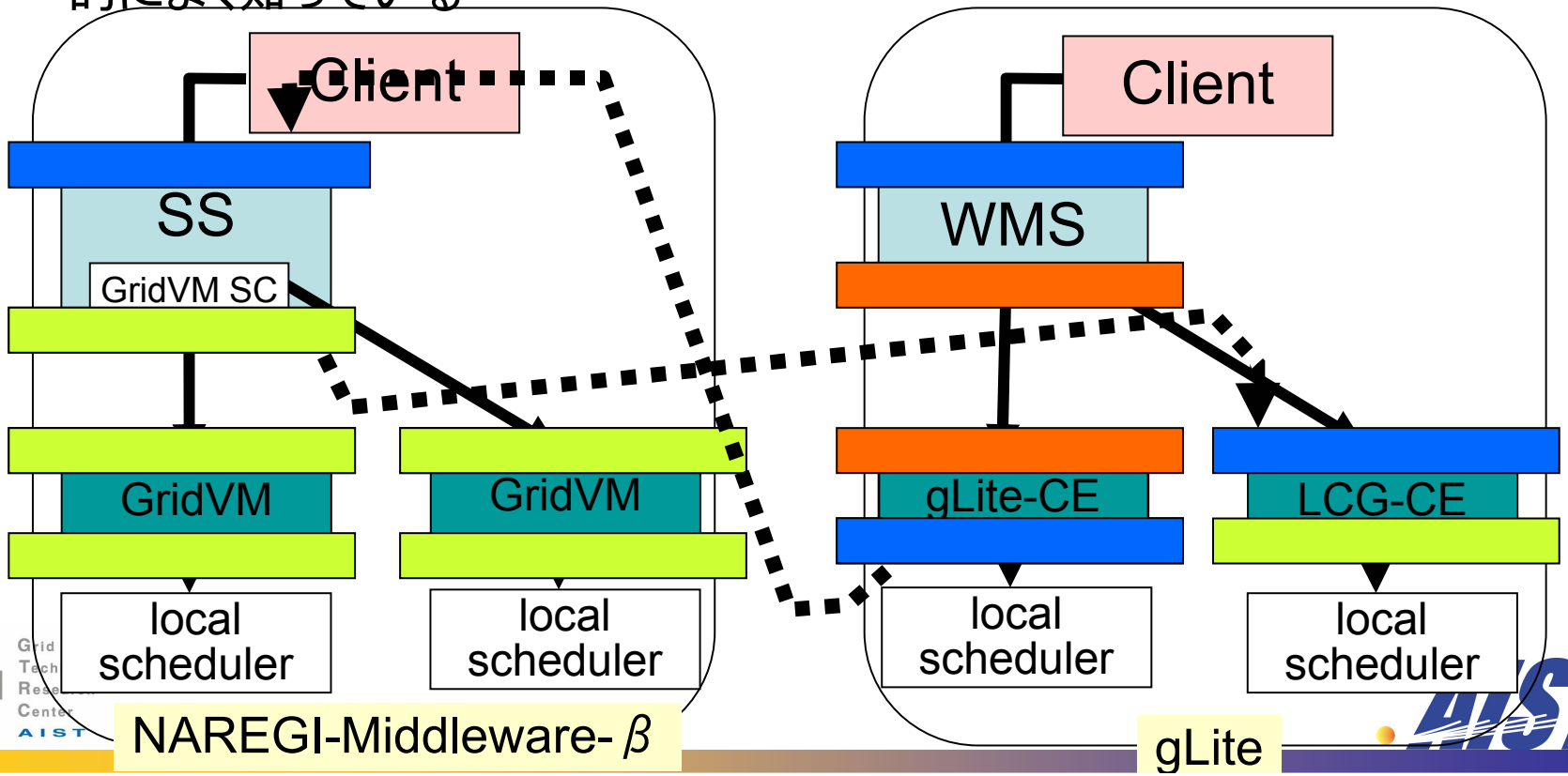
相互ジョブサブミッションの設計

- どこからジョブ起動を対象グリッドに出すか？どこから受け入れるか？
  - 共有できる・標準化されたプロトコルを使用している部分があればそこがよい
  - NAREGIのプロトコルは，改変可能
  - gLite-CEの内部プロトコル(BLAHP)は個人的によく知っている

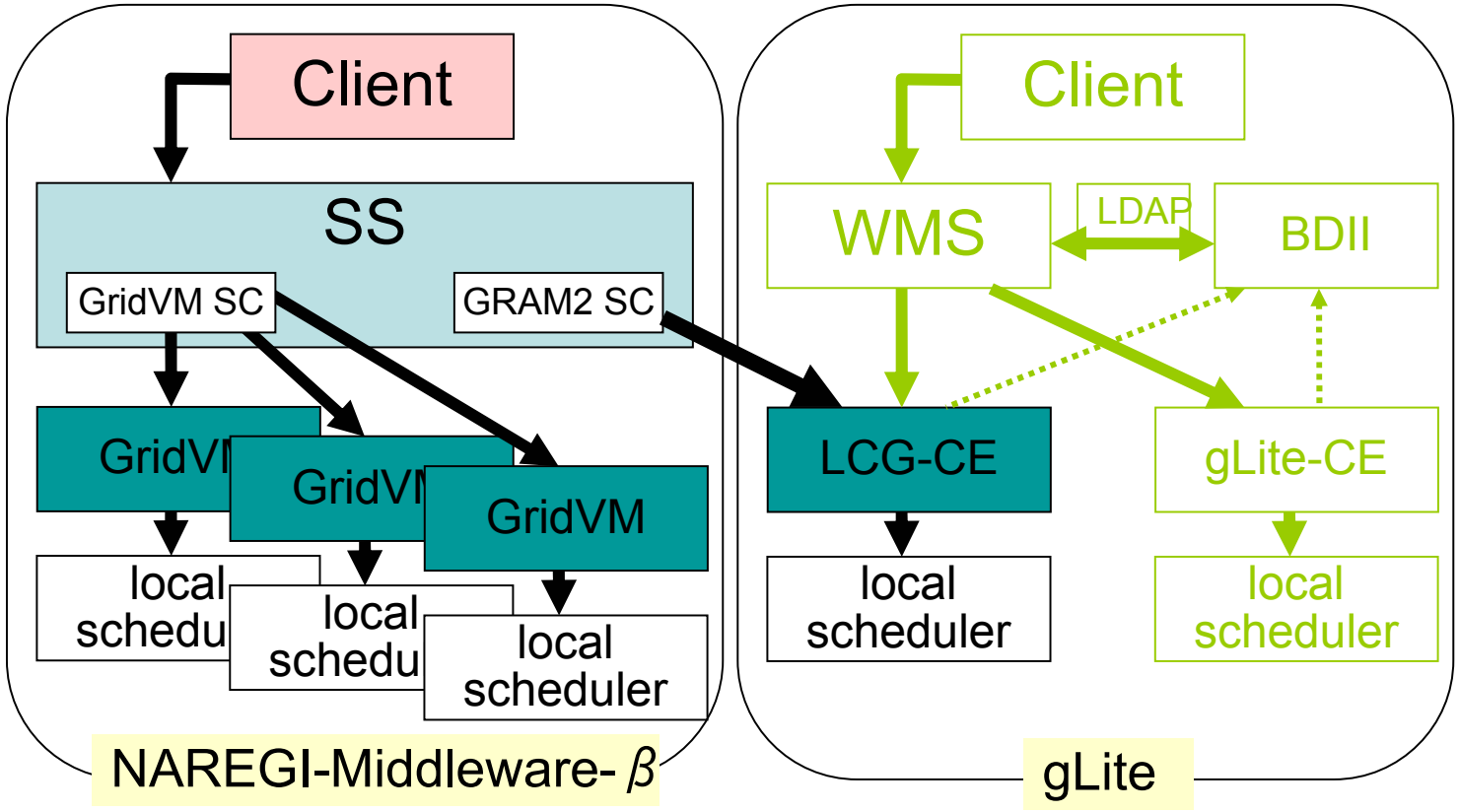
APIライブラリが存在

インターフェイスが定義・公開されている

非公開の内部プロトコル



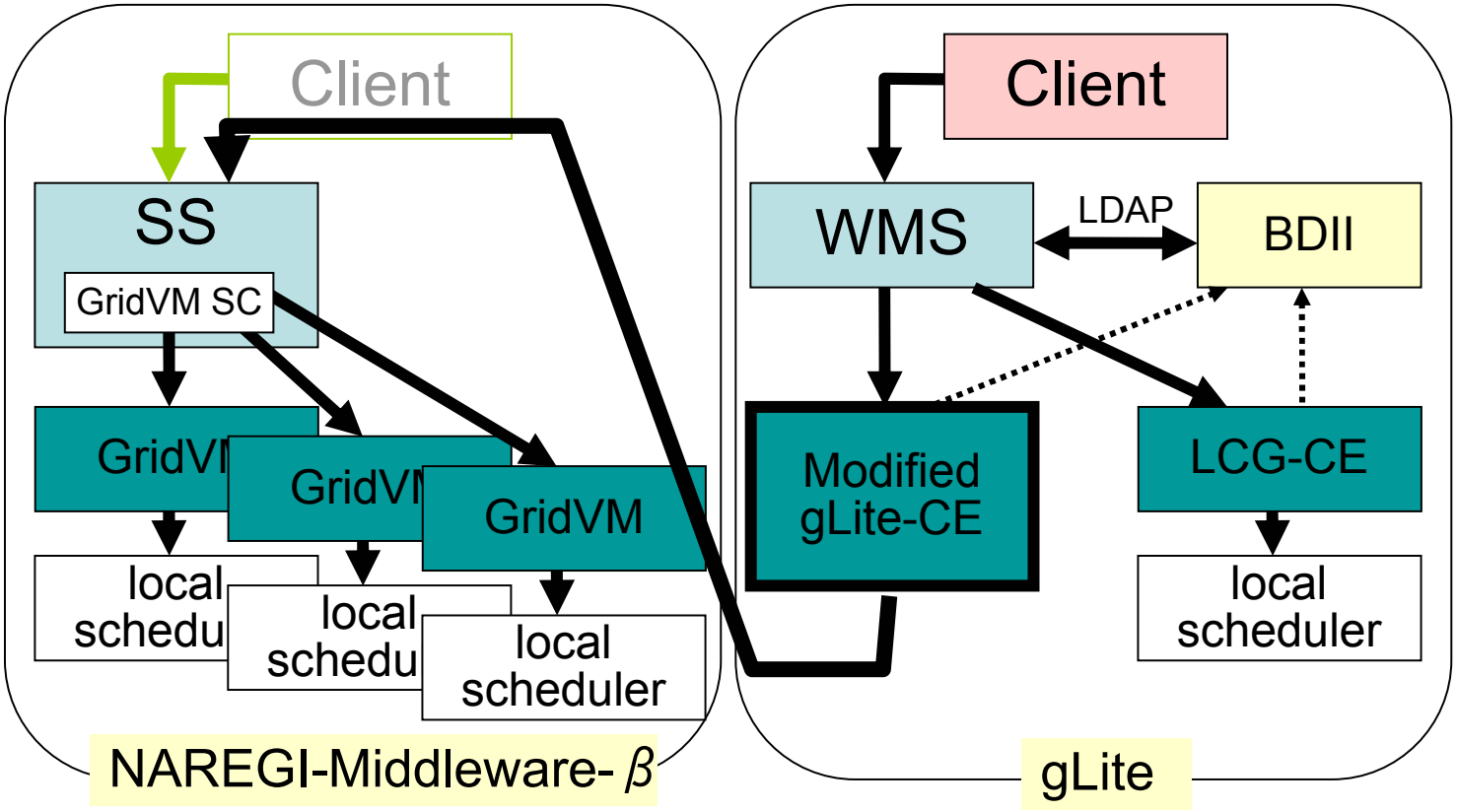
NAREGI→gLite ジョブ起動の実現



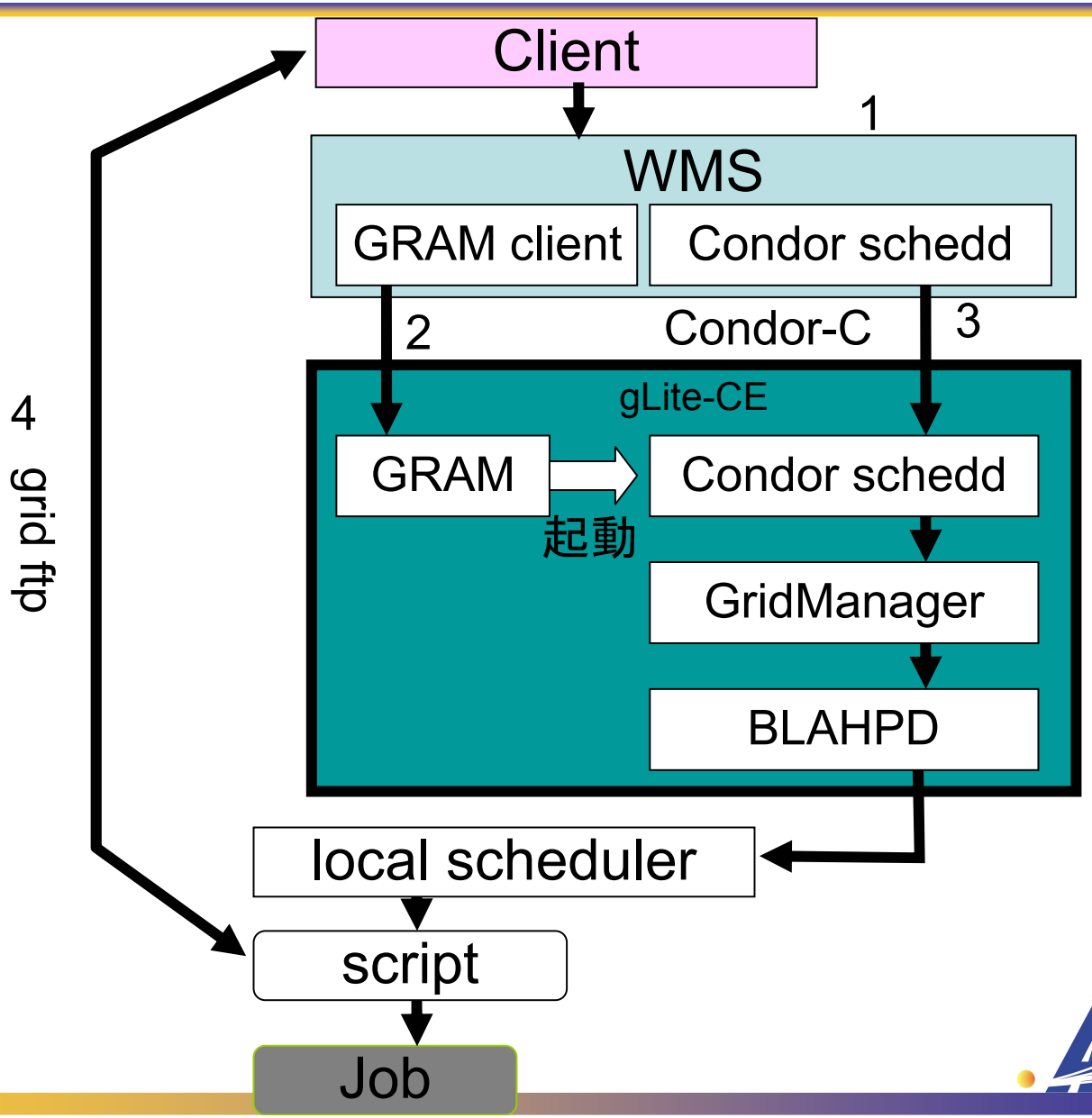
NAREGI→gLite ジョブ起動の実現

- GridVM の代わりに GRAM2(LCG-CE)を呼び出すSCを開発
  - SCは動的にリンクされるモジュールとして実装されているため，比較的容易
  - GRAM2では予約が利用できないため，予約したフリをする
- どちらのSCを利用するかは，情報サービスから得た情報によって分岐
  - ユーザは特に意識することなく双方を利用することができる

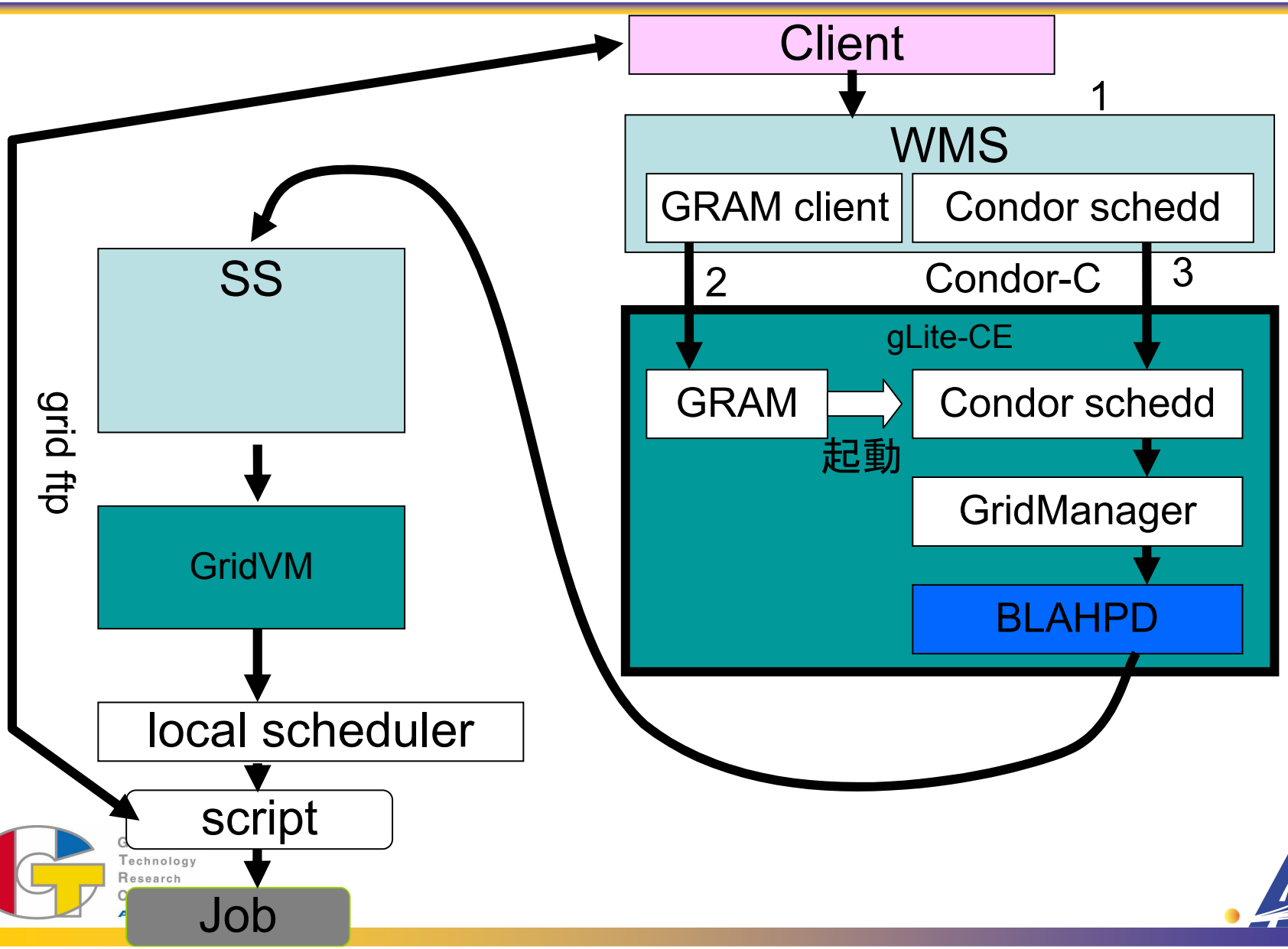
gLite -> NAREGI ジョブ起動の実現



実装の詳細


Client
1
WMS
GRAM client
Condor schedd
2
Condor-C
3
gLite-CE
GRAM
起動
Condor schedd
GridManager
BLAHPD
local scheduler
script
Job
4
grid ftp

実装の詳細

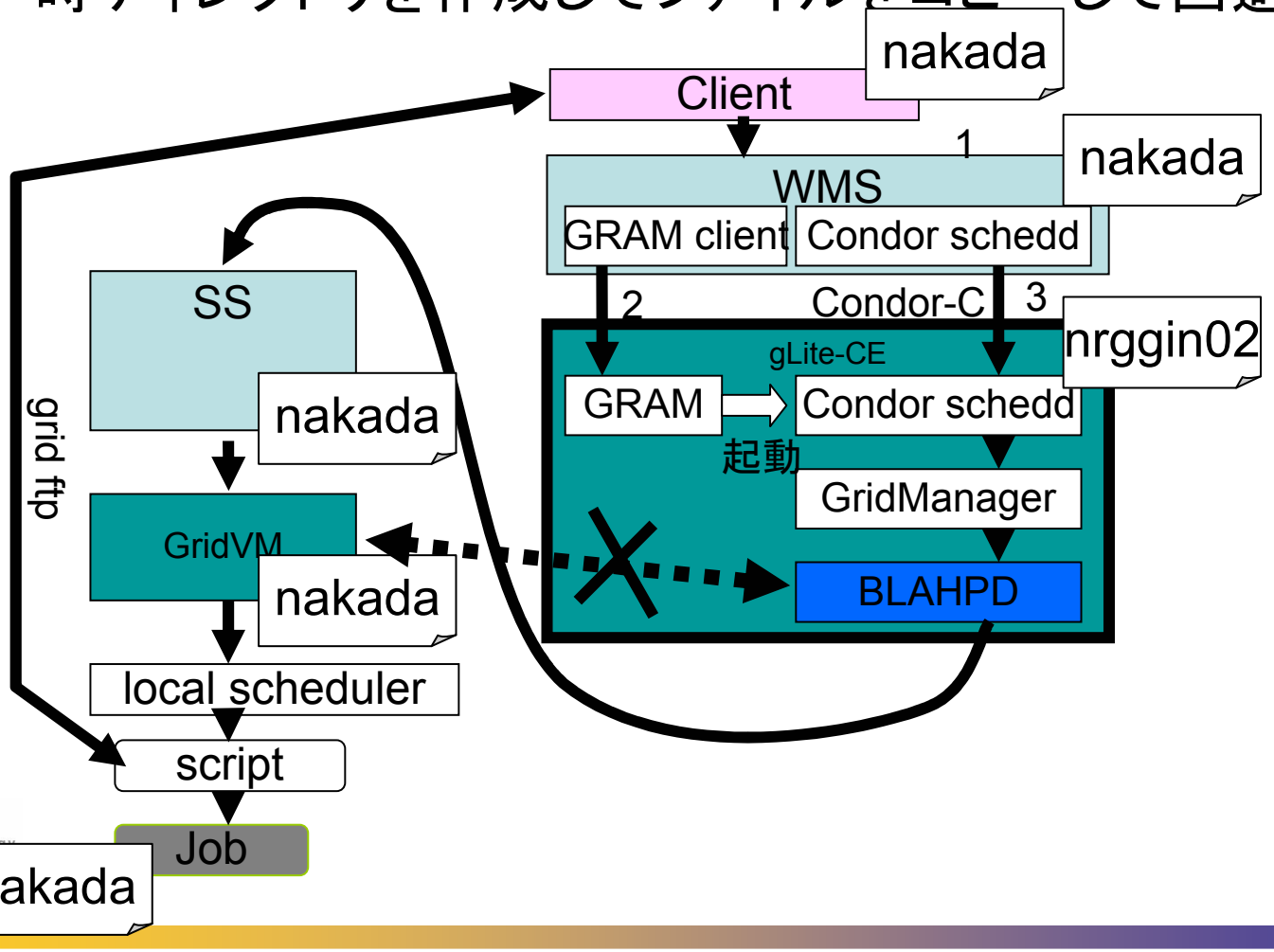

Client
1
WMS
GRAM client
Condor schedd
2
Condor-C
3
gLite-CE
GRAM
起動
Condor schedd
GridManager
BLAHPD
SS
GridVM
local scheduler
script
Job
grid ftp

BLAHP プロトコル

- テキストベースのアダプタ用プロトコル
  - GAHPにコマンドセットを定義したもの
  - GAHP(Globus Ascii Helper Protocol) – Condorから Globus を呼び出すために定義されたテキストプロトコル
- UNICORE 用 GAHP（中田 ‘04）コマンドセットをベース
  - BLAH_JOB_SUBMIT
  - BLAH_JOB_STATUS
  - BLAH_JOB_CANCEL
- UNICORE用 GAHPDのコードを流用可能

技術的問題点と解決法

- CE上でのユーザが仮想的なユーザになるためNAREGIへのジョブサブミッション時にファイルステージングがうまくいかない
  - 一時ディレクトリを作成してファイルをコピーして回避



技術的問題点と解決法

- プロキシ証明書の段数制限
  - プロキシ証明書 – 子証明書発行機関認証の枠組みを利用
  - 理論的には段数に制限はない
  - globusによるgridftp実装上の問題
    - openssh ライブラリのデフォルト証明書発行機関段数による制限 – 8段
  - 解決法
    - gridftpを利用するプログラムを特殊なオプションで起動して回避
    - 本来はgridftp を修正するべき

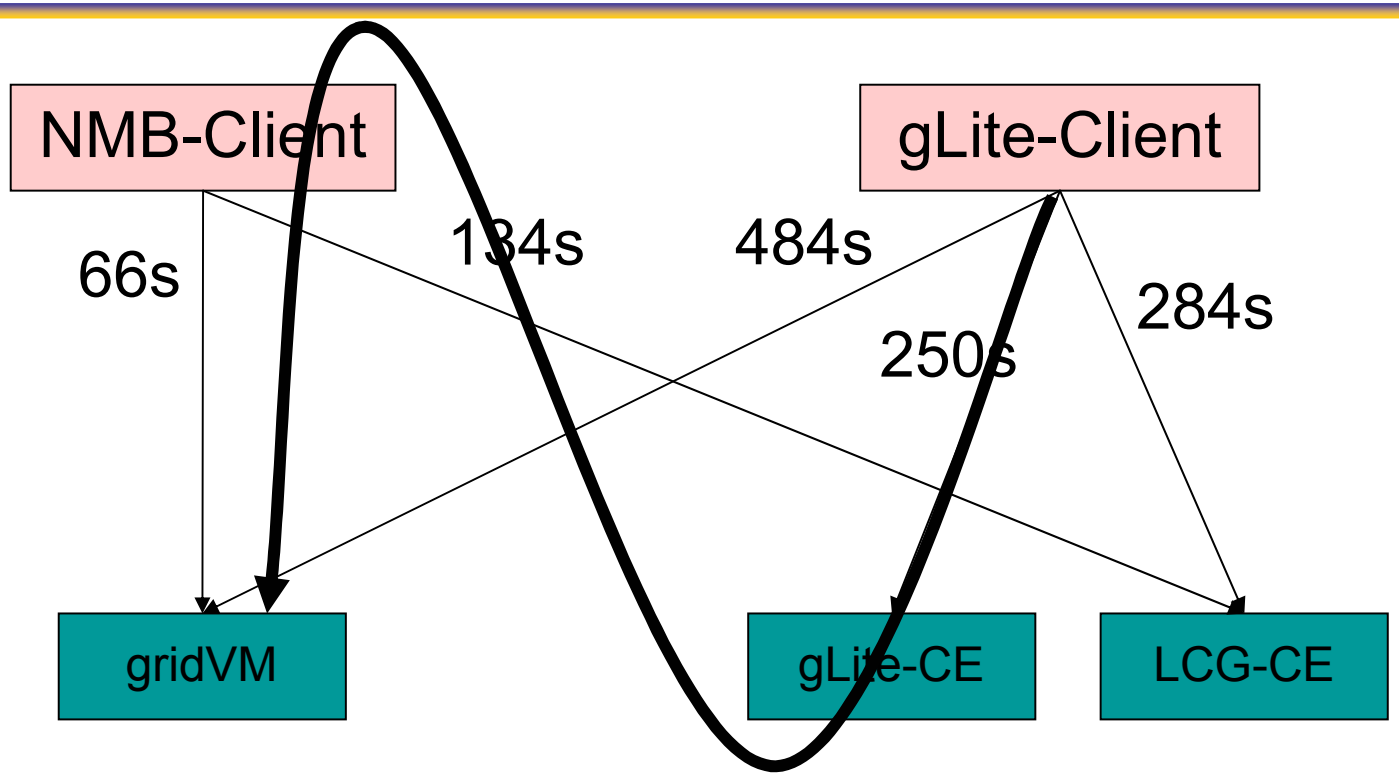
発表のアウトライン

- グリッドミドルウェアの構造
  - NAREGI Middleware β
  - gLite
- 相互運用の方針と実現
- **測定**
- 結論

実験

- 相互ジョブ実行にかかる時間を計測
  - 比較対照として各ミドルウェア内部での実行時間も測定
  - 10回測定した平均値
- 実行環境
  - すべてNAREGI 側に配置したノード内で実行

実験結果



- 使用機材
  - Pentium 4 Xeon 3GHz dual, メモリ1Gbyte, RedHat 8
  - ネットワーク 1000base-T

発表のアウトライン

- グリッドミドルウェアの構造
  - NAREGI Middleware β
  - gLite
- 相互運用の方針と実現
- 測定
- **結論**

結論と今後の課題

- NAREGI Middleware β と gLite の間でジョブ起動に関する相互運用実験を行い下記を確認
  - 証明書や仮想組織管理のレイヤでは相互運用性に問題はない
  - 情報サービスのレイヤでも相違が吸収できる
  - その情報を用いて相互のジョブ起動が可能である

今後の課題

- より精密な測定と内訳の解析
- 実運用環境を用いての実験
  - 実環境のVOMSでの運用性を確認
  - 日欧間のレイテンシの影響
- より洗練された 相互ジョブ実行法の検討
  - N x N のブリッジングははばかげている
  - 標準化されたインターフェイスをすべてのグリッドミドルウェアが利用するのが理想

謝辞

- 本研究は文部科学省「経済活性化のための重点技術開発プロジェクト」の一環として実施している，超高速コンピュータ網形成プロジェクト(NAREGI：National Research Grid Initiative)によるものである．