グリッドRPCシステムNinf-Gの可搬性および適応性の改善

1.産業技術総合研究所, 2. 創夢

中田秀基[1]、朝生正人[1,2]、谷村勇輔[1]
田中良夫[1]、関口智嗣[1]

Grid Technology Research Center AIST

AIST

背景

- グリッド上のプログラムモデルGridRPC
  - OGF(Open Grid Forum(旧GGF))で標準化
- Ninf-G ver. 4
  - Globus Toolkitを用いたGridRPCの実装のひとつ
  - 実行ファイル起動機構はInvoke Serverとしてプラグイン可能［05年6月，06年10月のHPC研究会で発表］
  - 通信部分には Globus Toolkit のGlobus I/Oを利用

→ Globus Toolkit に依存

背景(2)

- Ninf-G4の問題
  - ユーザベースを制約
    - Globus Toolkitのインストールはそれほど容易ではない
  - プラットフォームを制約
    - Globus Toolkitが移植されていないプラットフォームでは使用できない
  - 通信レイヤを制約
    - Globus-I/O以外の通信レイヤを使用できない

研究の目的

- 通信プロキシという概念を導入して，Ninf-G4の問題点を解決
  - Ninf-G ver. 5の提案
  - 通信プロキシ
    - 通信レイヤをハンドルする独立したプロセス
    - コアライブラリを通信レイヤ非依存に
- 通信プロキシのオーバヘッドを予備評価

発表の概要

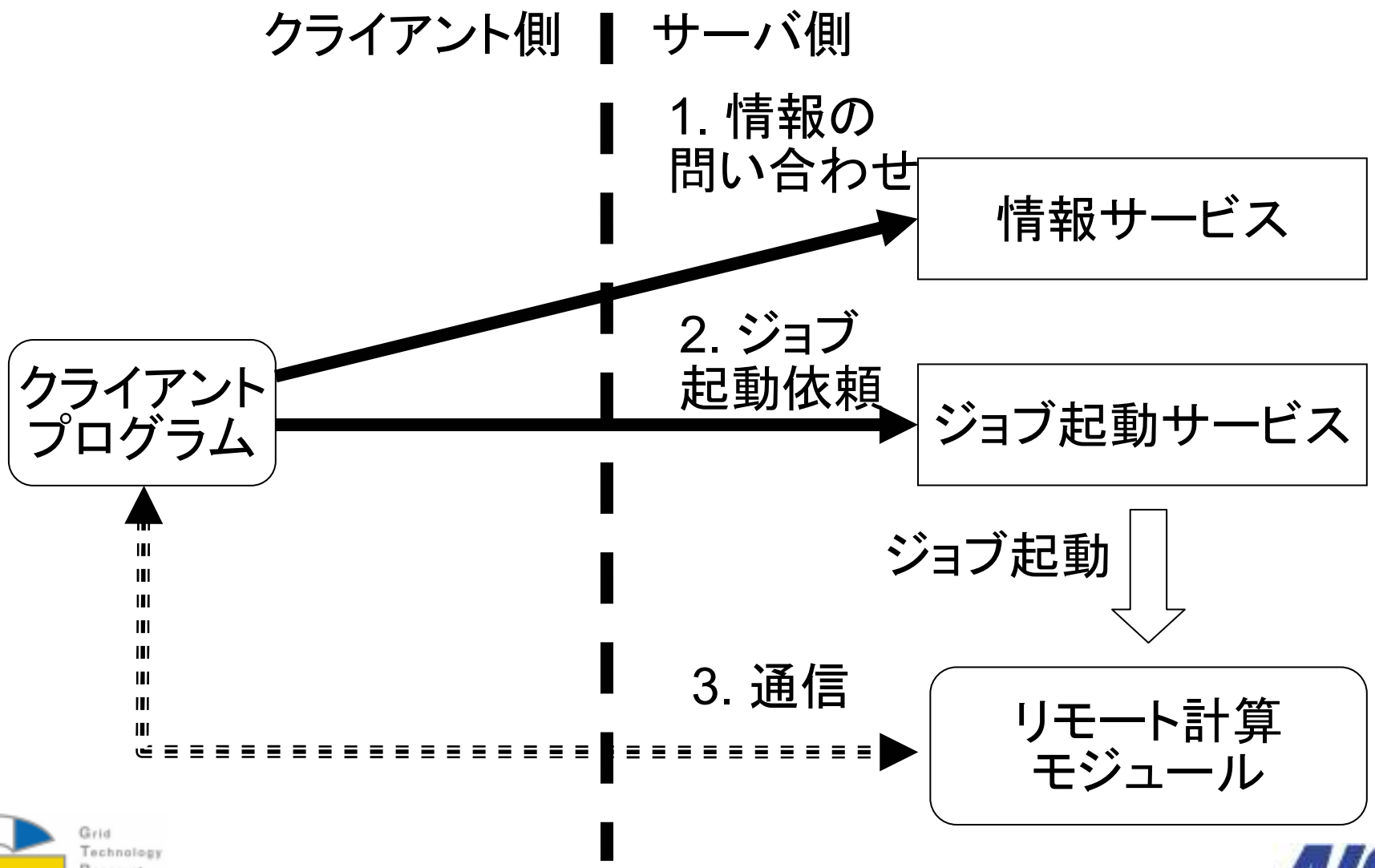
- Ninf-G の概要
- 通信プロキシの提案
- 予備評価
- おわりに

Ninf-G

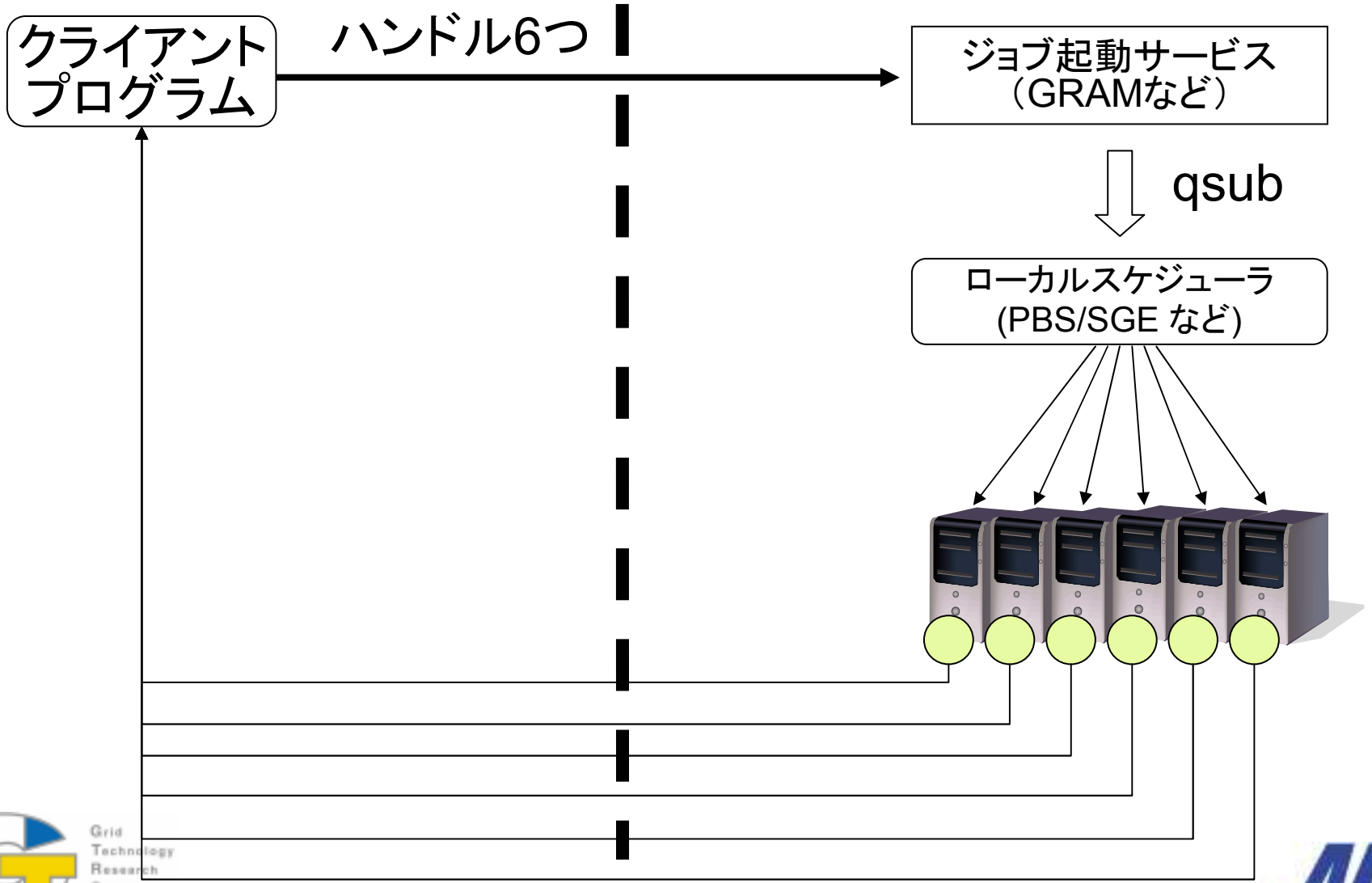
- グリッド上でRemote Procedure Call を実現
  - サーバ側の関数をクライアント側から使用
  - 引数転送は暗黙裡に行われる

```
/* initialize function handle */
  for (i = 0; i < NUM_HOSTS; i++)
    grpc_function_handle_init(&handles[i], hosts[i], port, "pi/pi_trial");
  for (i = 0; i < NUM_HOSTS; i++)
   /* parallel invocation */
    gprc_call_async(&handles[i], i, times, &count[i]);
  /* wait for all the calls */
 grpc_wait_all();
```

Ninf-G の構造


クライアント側
サーバ側
1. 情報の
問い合わせ
情報サービス
2. ジョブ
起動依頼
クライアント
プログラム
ジョブ起動サービス
ジョブ起動
3. 通信
リモート計算
モジュール

典型的な利用法


クライアント
プログラム
ハンドル6つ
ジョブ起動サービス
（GRAMなど）
qsub
ローカルスケジューラ
(PBS/SGE など)

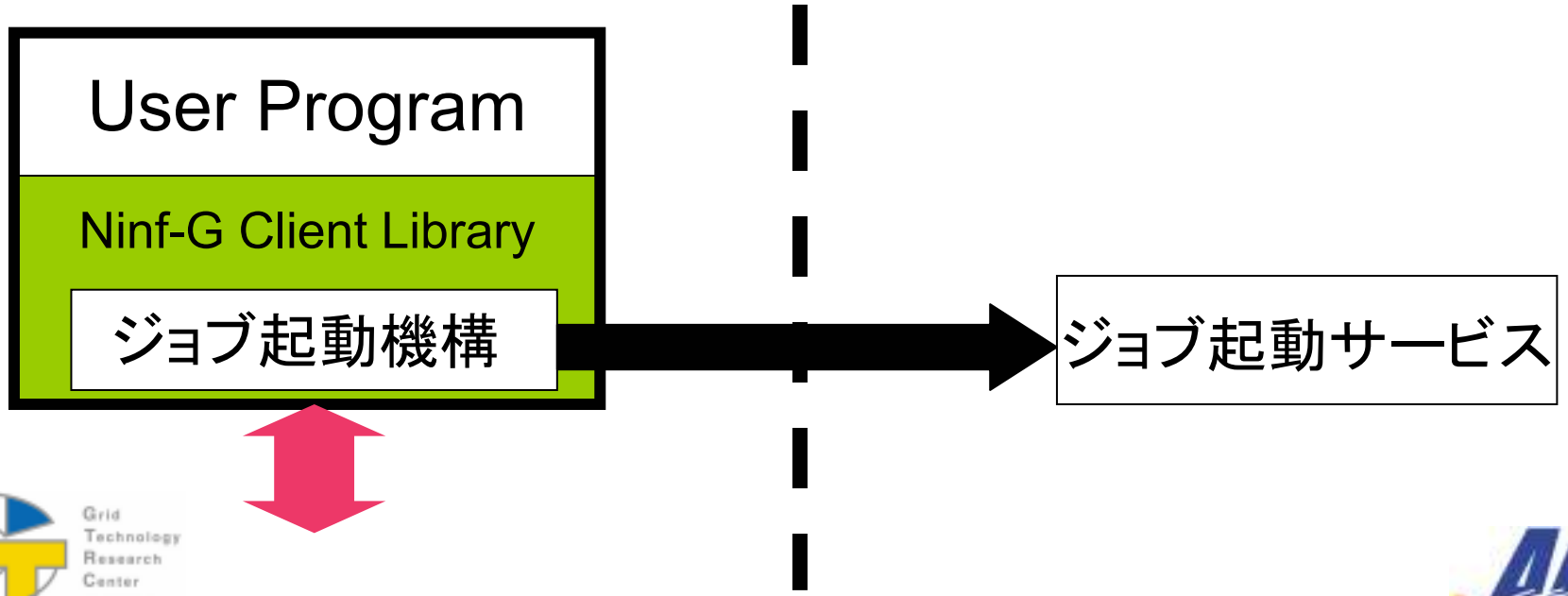
特定のミドルウェア（Globus Toolkit）への依存を解消

- 可搬性，適応性向上を目指す

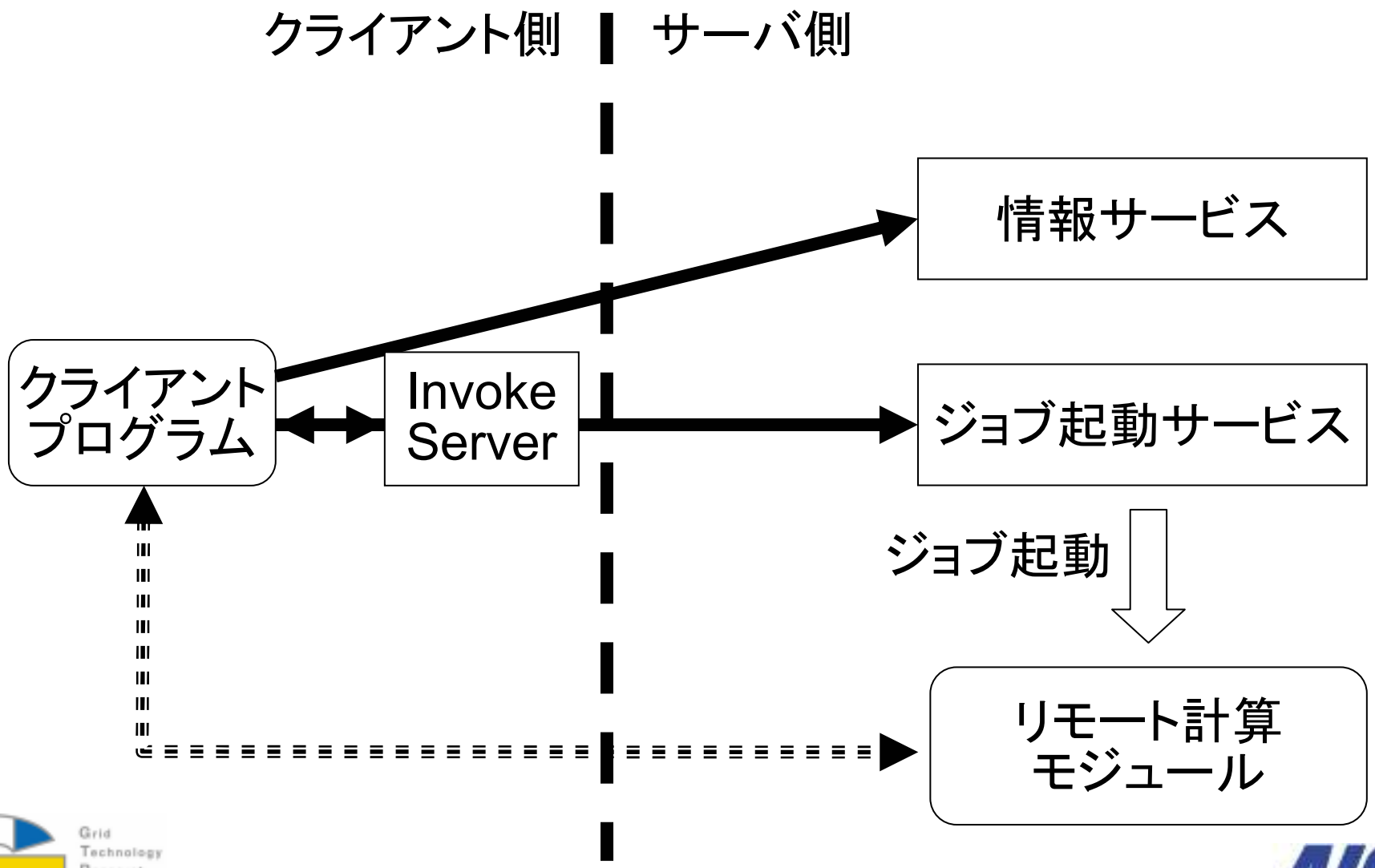
- 下記の3点に対処が必要
  - 情報サービス
    - ファイルを用いた情報取得が可能
  - ジョブ起動サービス
    - Invoke Server [中田 05]を用いることで解消
  - 通信
    - 本稿で対処

Invoke Server

- Invoke Server 以前
  - 起動機構をクライアントライブラリに内蔵
  - クライアントとのリンクが必須
  - 複数のジョブ起動機構を利用することができない
- Invoke Server ：ジョブ起動機構をクライアントライブラリから外に
  - ジョブ起動部分をクライアントライブラリとは別の言語で実装することが可能に
  - クライアントライブラリ本体を修正することなく、他のリモートジョブ起動機構に対応可能



Invoke Server



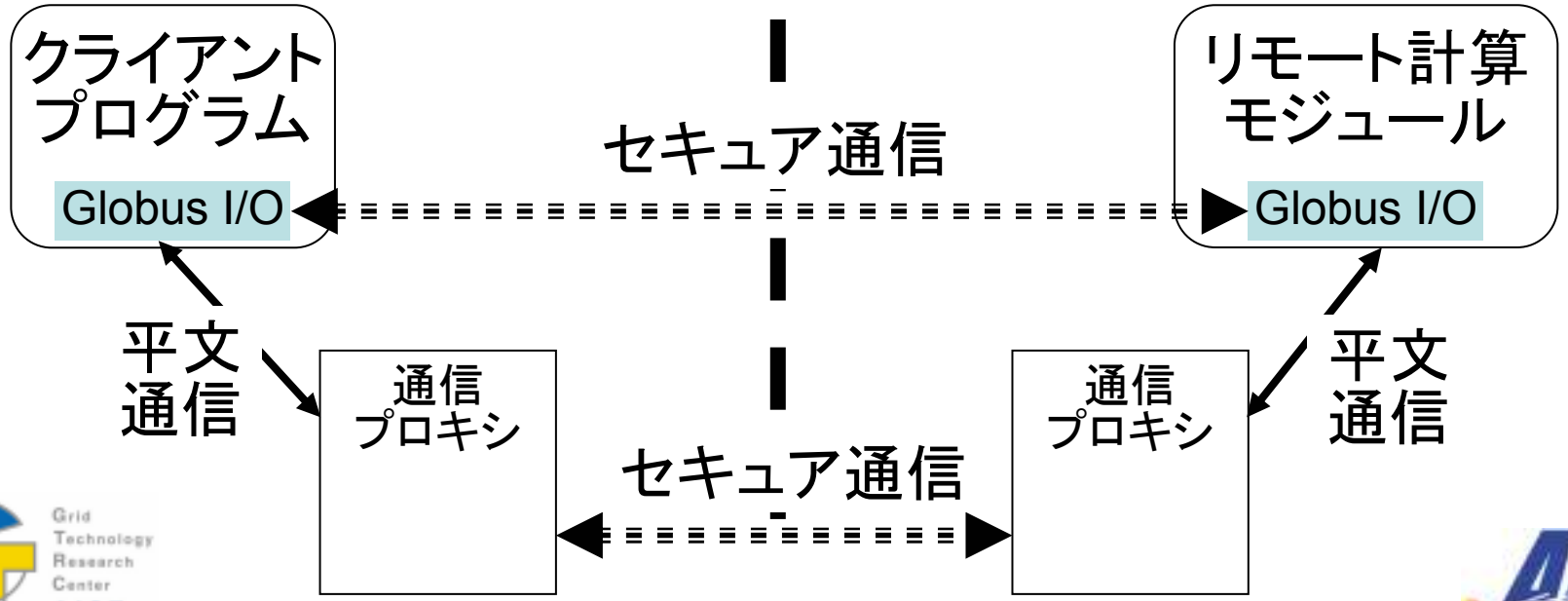
Ninf-G version History

- Ver.1 – GT 2 向け, プロトタイプ
- Ver.2 – GT 2 向け, 完全な再実装
- Ver.3 – GT 3 向け,
- Ver.4 – GT 4 むけ, Invoke Server による対応

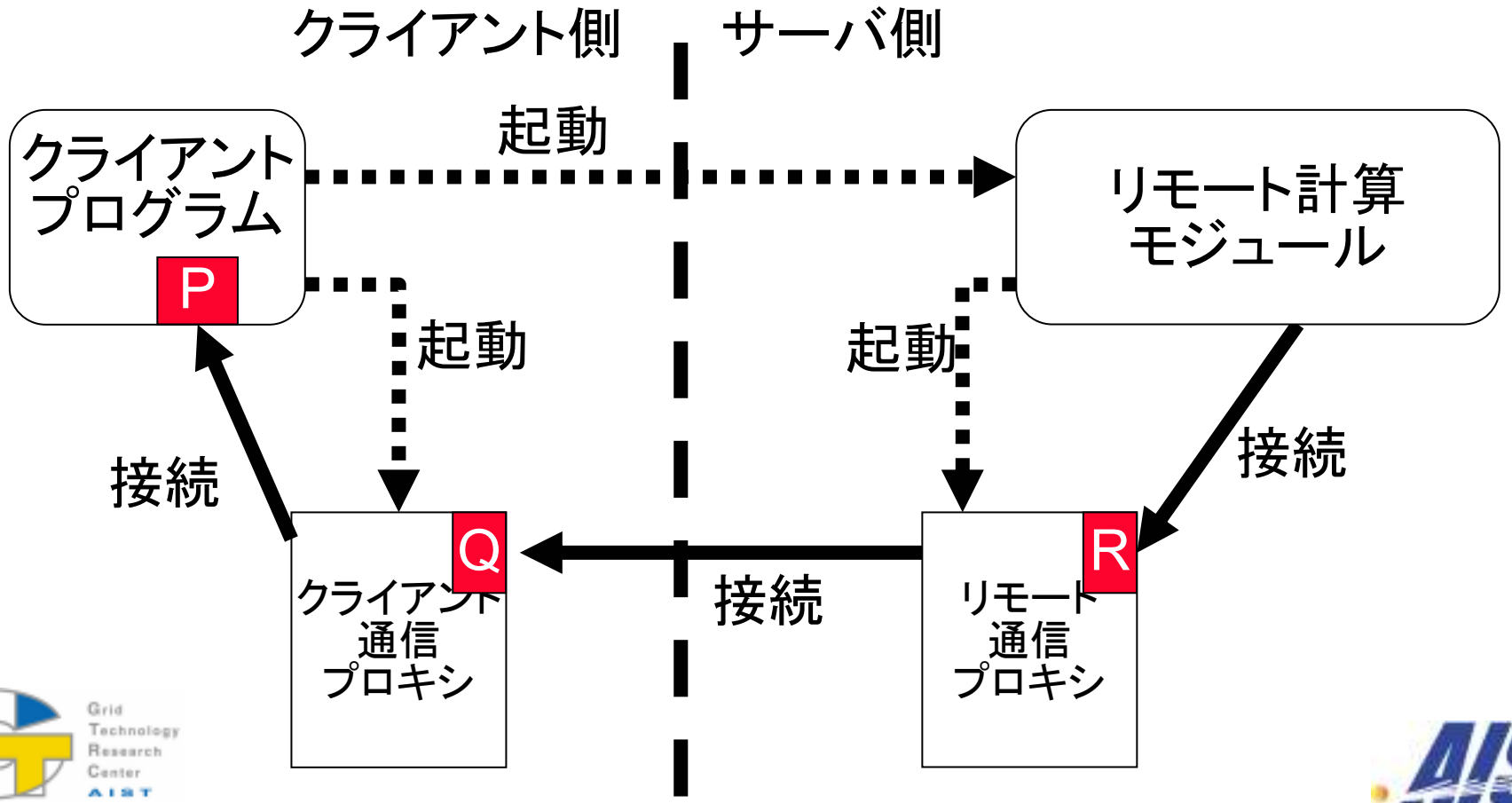
- Ver.5 – 特定のミドルウェアに依存せず

通信プロキシの導入

- 通信プロキシ
  - クライアントとリモートモジュールの間で通信を中継
  - 通常のUNIXソケット通信とグリッドミドルウェアに依存した通信を変換



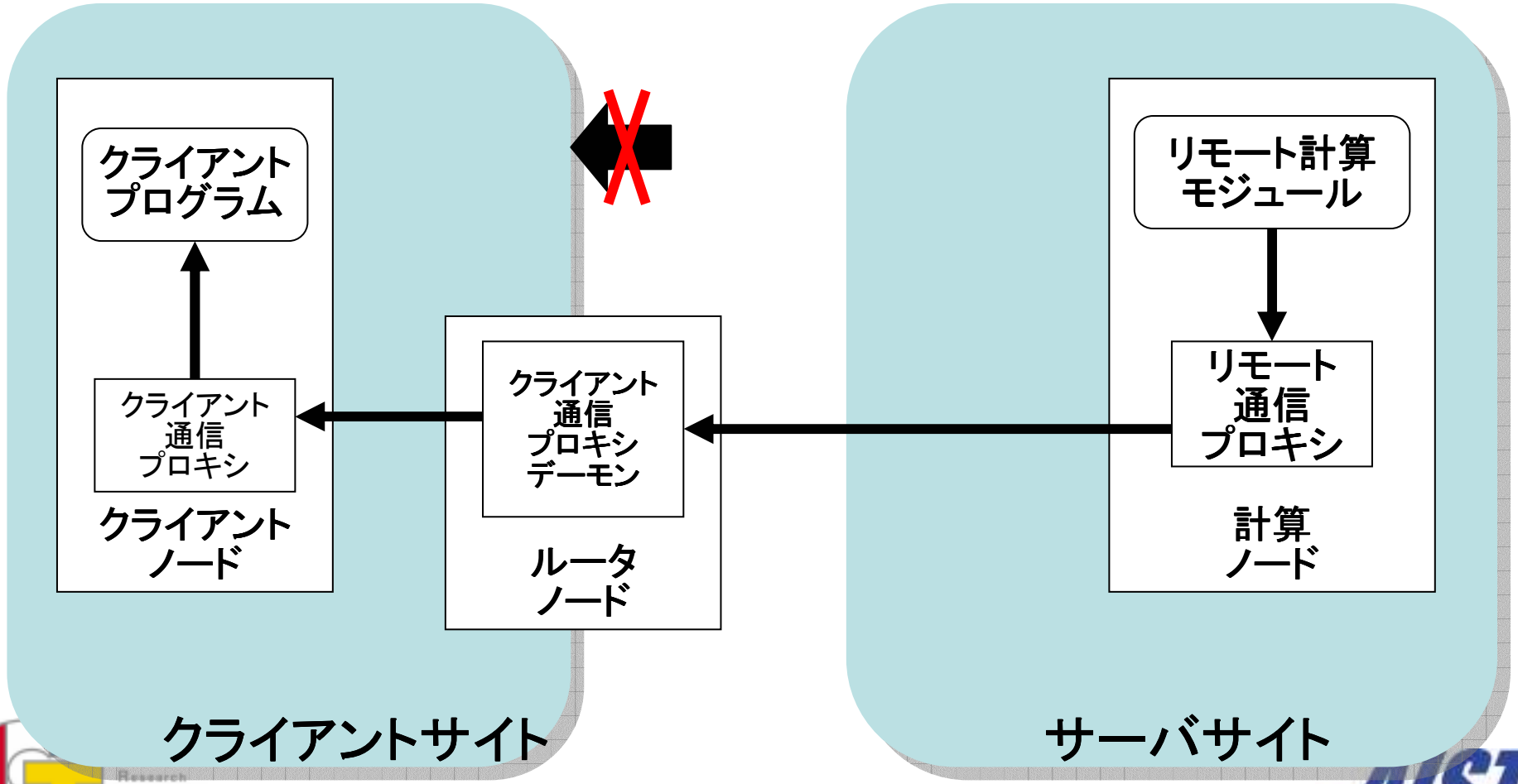
通信プロキシ起動プロセス



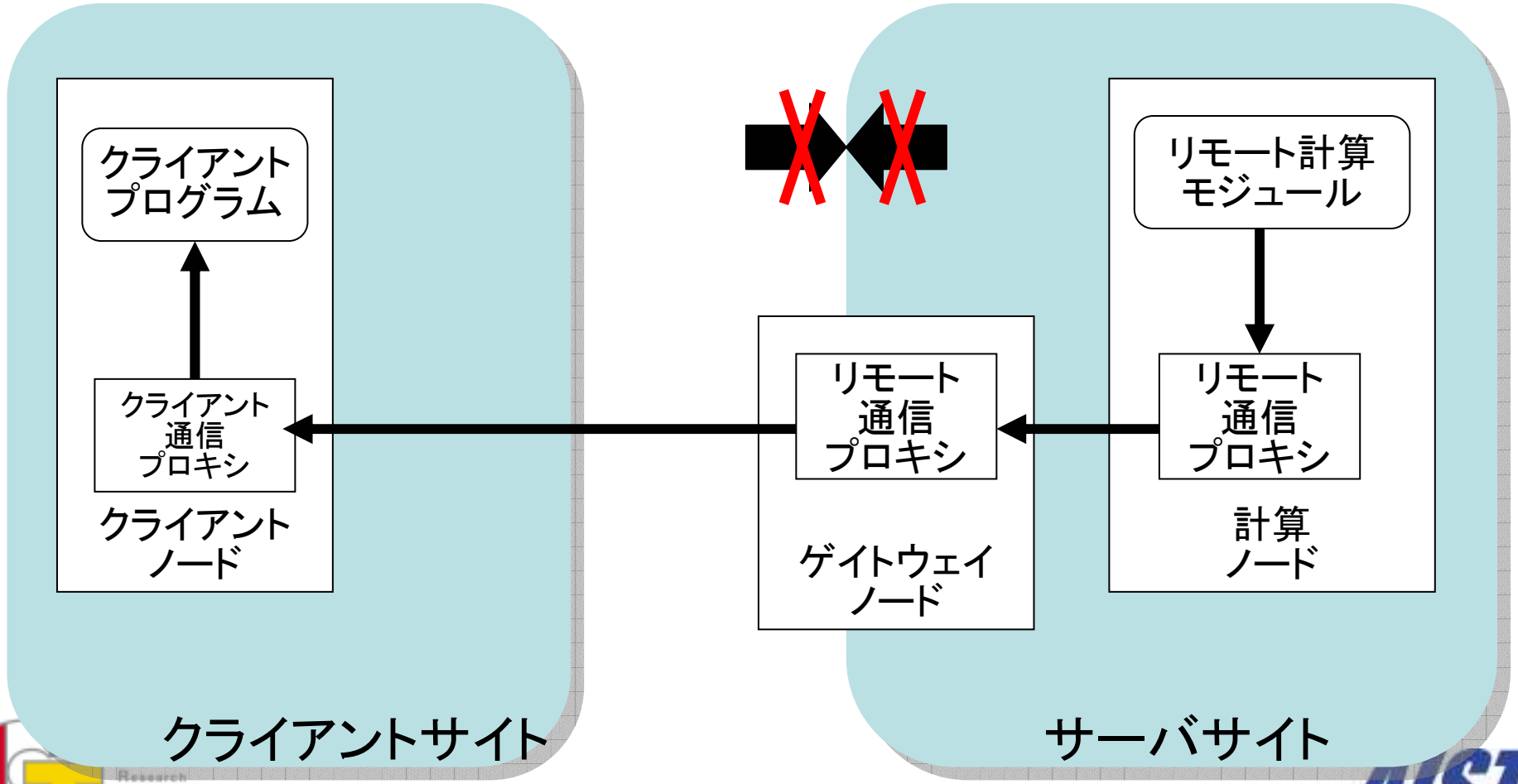
一般化通信プロキシ

- クライアントは平文ポートを開けて待つ
- リモート計算モジュールはどこかのポートに向けて平文でコネクト
- なんらかのプロキシ網によって通信がフォワードされる

プライベートアドレスのクライアントサイトへの対応

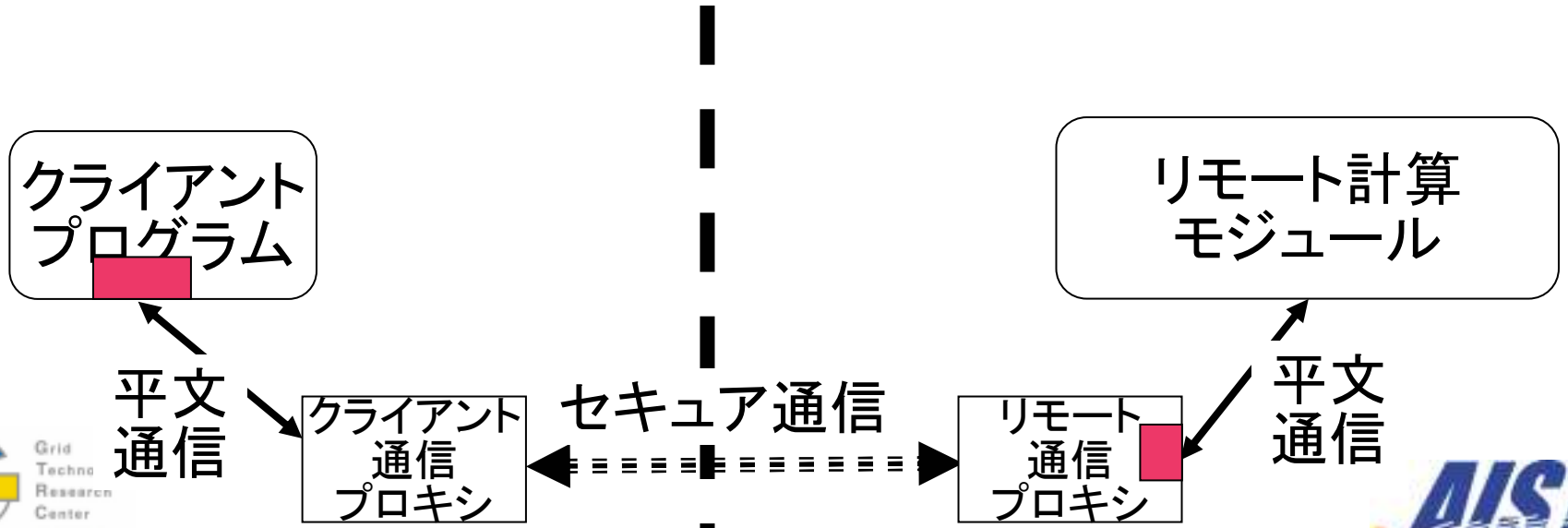


非NATプライベートアドレスクラスタへの対応



セキュリティに関する考察

- 平文ソケットが外部に露出する点が存在
- 多くの場合はノード内通信
  - セキュリティ的には問題になりにくい
  - UNIX Domain Socket を使用すればOS依存で通信相手のユーザIDを認証することが可能

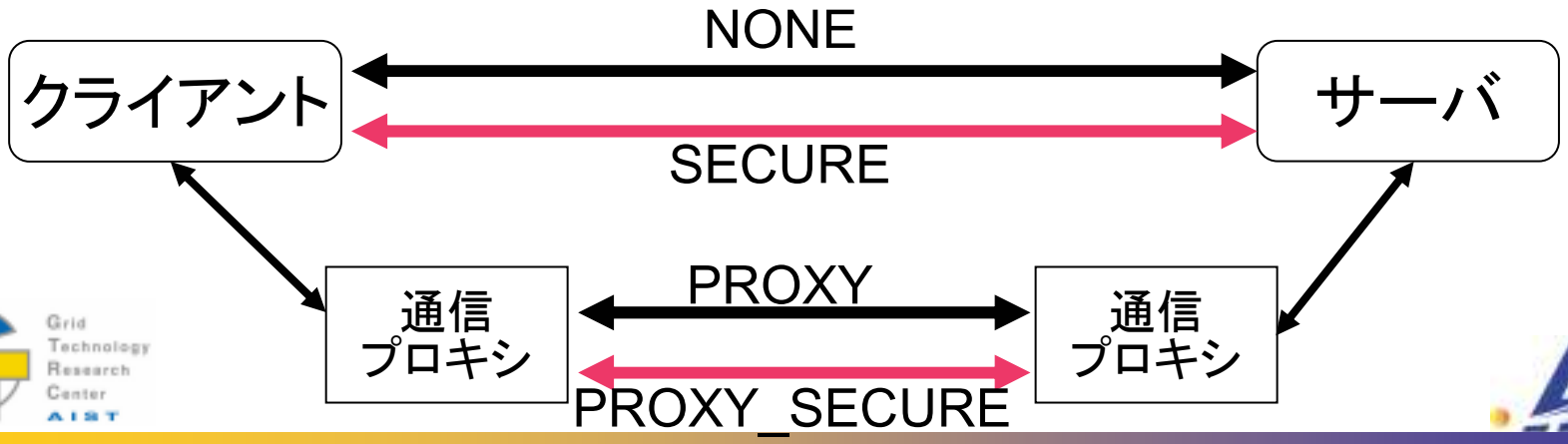


セキュリティに関する考察（2）

- 中継プロキシを利用する場合は，ノード外に対するソケットが露出
- 共通鍵を利用した認証を検討中
  - 共通鍵をクライアントごとに生成して，ファイルとしてジョブに添付して転送
  - ファイルのパーミッションをユーザのみread可能に
  - c.f. Globus のプロキシ証明書 ー ファイルとして存在

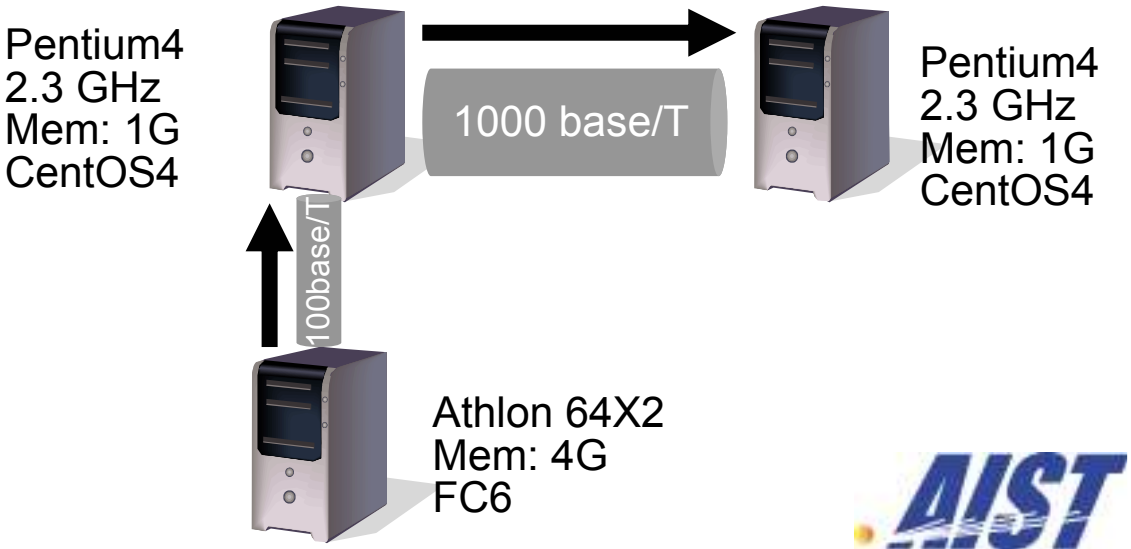
予備評価

- 通信プロキシによる通信性能への影響を調査
- Globus I/O を用いたベンチマーク用プログラム群を作成して実験
  - クライアント，サーバ，プロキシ
  - セキュリティ ON/OFF
  - プロキシ ON/OFF

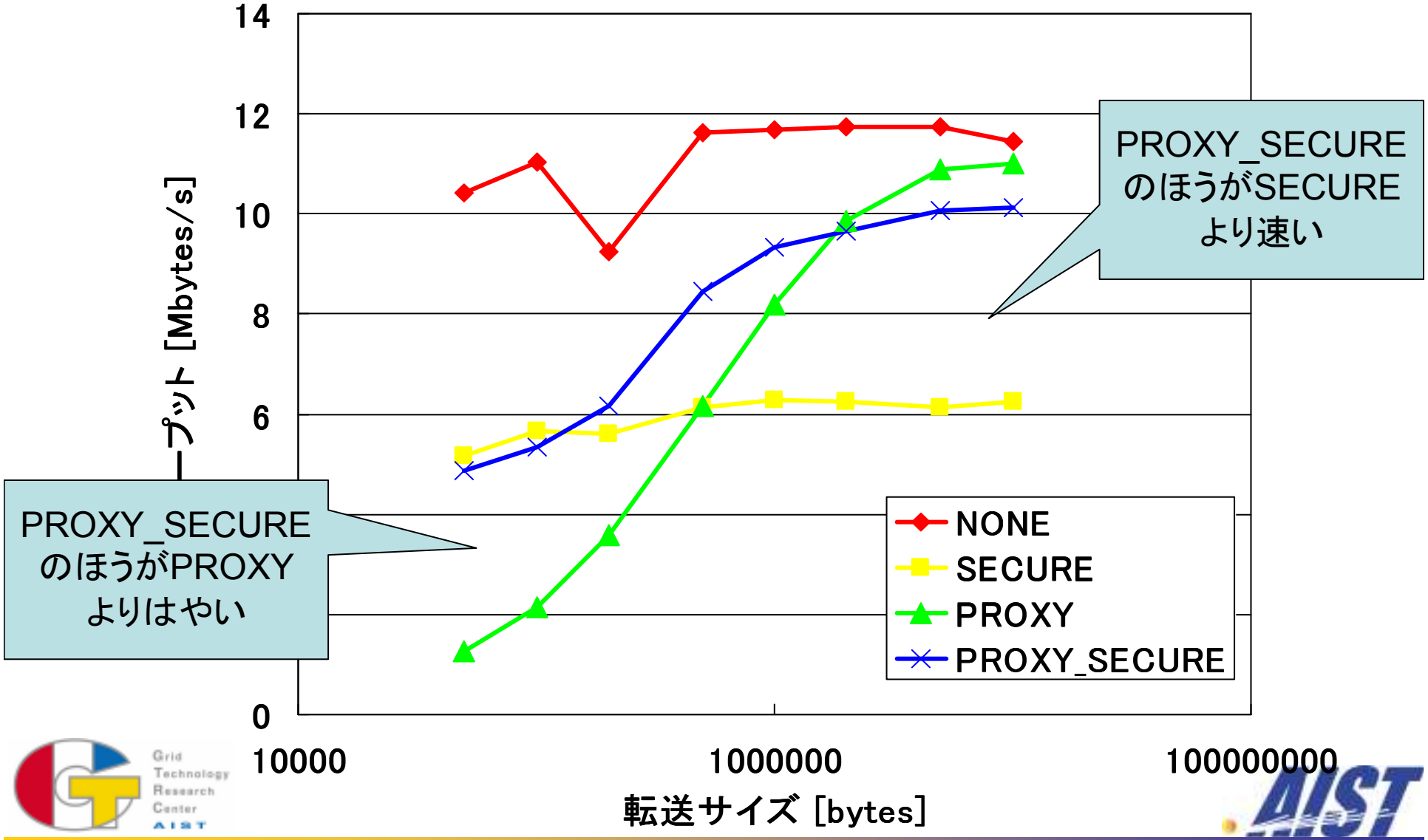


評価環境

- 100base/TX と 1000 base/T で比較
- プログラム
  - バッファしたデータをピンポン
  - 多数回繰り返し，平均値を算



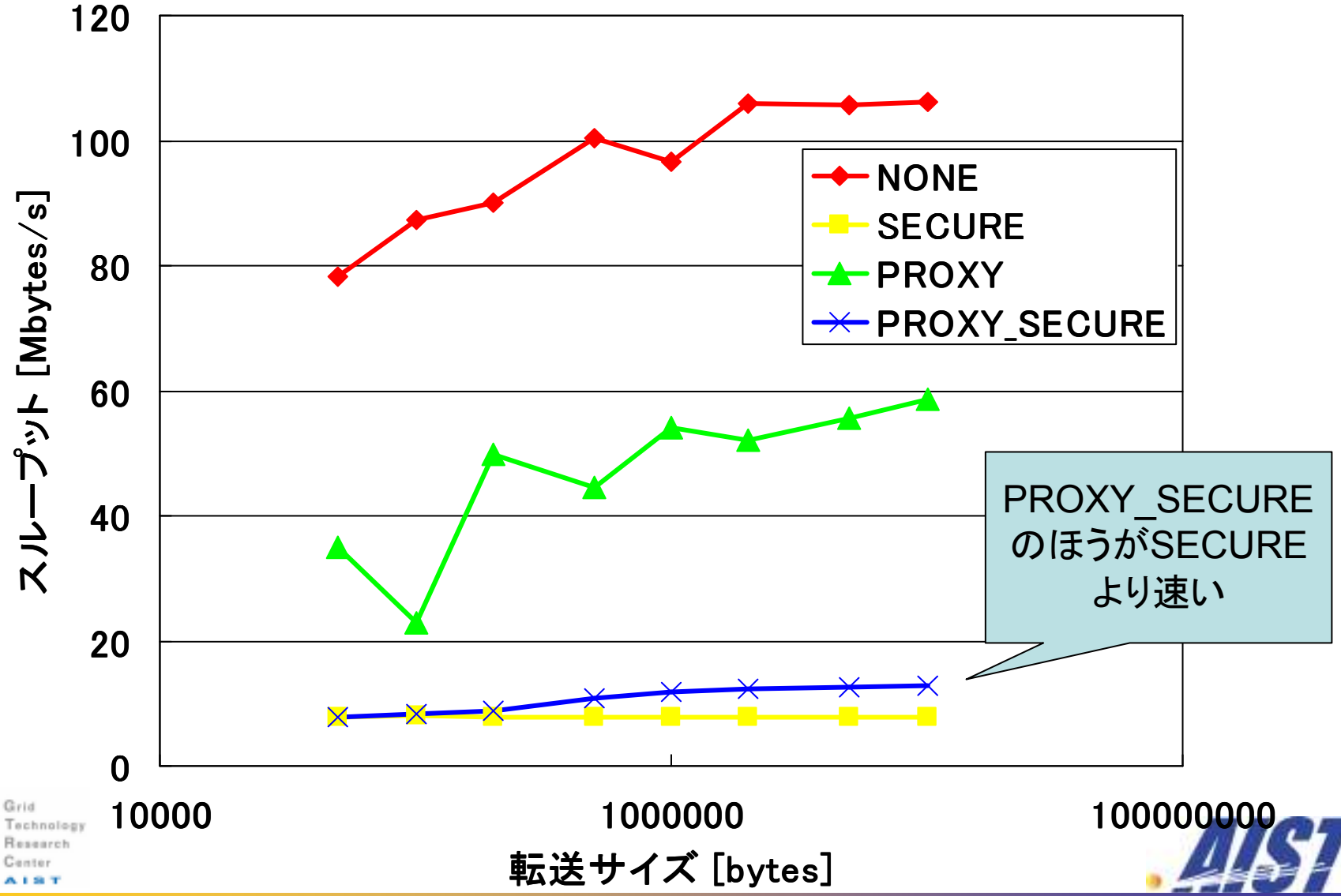
100 base/TX での評価



Grid Technology Research Center AIST

AIST

1000 base/T での評価



考察

- 100 base/TX 環境では通信プロキシ導入のオーバヘッドは大きくない.
- 1000 base/T 環境ではかなり大きなオーバヘッドがある
  - 直接接続・セキュリティ ON時の挙動が明らかにおかしい
  - Ethereal で調査したところ，通信の方向が切り替わるところで，一時的にストールしている様子
- より詳細な調査が必要
  - ストールの原因調査
  - Ninf-Gの実アプリケーションを用いた実験

関連研究

- OmniRPC (筑波大学)
  - Globus Toolkit の使用不使用を ビルド時に決定可能
  - 他の通信プロトコルへ対応するためには，ソースコードの変更が必要

おわりに

- GridRPCシステムに対する通信プロキシ機構の導入の提案
  - 特定のグリッドミドルウェアに対する依存から脱却
  - 高い可搬性
  - 他のグリッドミドルウェアの提供する通信レイヤへの適応が容易に
- 通信プロキシによるオーバヘッドの予備評価
  - 低速ネットワークでは無視できる程度
  - 高速ネットワークではオーバヘッドは大きい
    - メリットを考えれば許容できる

現状 と今後の課題

- 現状
  - 今年度末にNinf-G5の実装が完了予定
- 今後の課題
  - 通信プロキシ使用時のオーバヘッドのより詳細な計測
    - 実際のプログラムを使用
    - チューニング
  - 通信プロキシをいくつか実装し設計の有効性を確認
    - Condor Chirp
    - P2P通信網
  - リモート計算機がTCP/IP を解釈しない場合への対応
    - Ex. Cray XT3 （TeraGrid BigBen）
    - より抽象化された通信レイヤを定義して吸収

謝辞

- 本研究の一部は文部科学省「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
- 産総研Ninf開発チームの皆様に感謝します