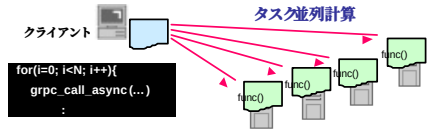
GridRPC における複数ノードにまたがる Task Sequencing の実現

谷村 勇輔，中田 秀基
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）

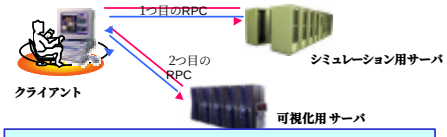
背景（1）

- **GridRPC**
  - RPC（遠隔手続き呼び出し）の仕組みをグリッドに拡張
  - グリッドアプリケーション開発のためのプログラミングモデル
- **GridRPC を用いた応用研究**
  - Ninf-G の性能，安定性の検証
  - 科学的な成果
  - ただし，単純な独立タスクの並列実行



背景（2）

- **ワークフロー的な実行モデルのサポート要求**
  - 例．シミュレーションの後に結果を可視化処理
    - 複数のマシンに対する RPC の逐次実行

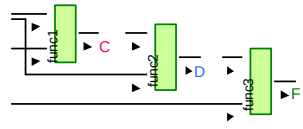


中間データがクライアントを介すことになり，効率が悪い
Ninf-G，GridSolve ではサーバ間の通信はサポートされていない
- API が用意されていない
- GridRPC の実装が複雑になる可能性がある（利点が損なわれる）

Task Sequencing

- **複数の関数の実行を１つのセットとして扱う**
  - 実行順序がある
  - 先の実行の出力データ（結果）が次の実行の入力データになる

```
begin_sequence();
  grpc("func1", A, B, C);
  grpc("func2", A, C, D);
  grpc("func3", D, E, F);
end_sequence();
```



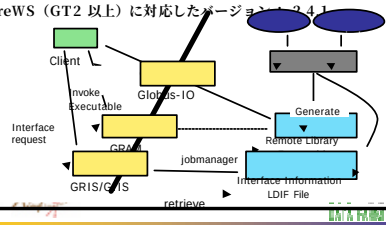
- **同一サーバで処理する場合の研究開発は行われている**
  - 中間データ C, D をクライアントに戻さない
  - クライアントは最終結果 F を受け取ればよい

研究目的とアプローチ

- **目的**
  - 複数ノードでの Task Sequencing 処理の実行を実現
    - GridRPC サーバ間のデータの直接転送を実現する
    - クライアント・サーバの特徴を保つ
    - GridRPC システムの修正を最小限に抑える
- **アプローチ**
  - Grid Filesystem を利用する
    - 大域的な名前空間が提供される
    - 高速なデータ転送機能が利用できる（特にストレージ間）
    - フォールトトレランス機能が利用できる
  - 本研究では Gfarm（Grid Datafarm）を利用する

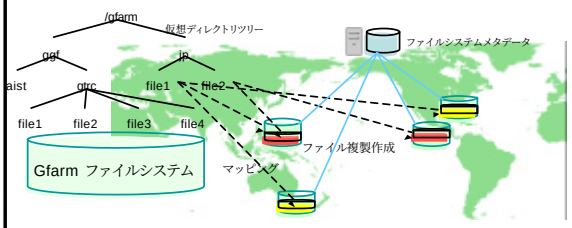
Ninf-G

- **GridRPCによるプログラム開発および実行を支援するソフトウェアパッケージ**
  - GT の上に構築されている
  - クライアント用に C, Java API，サーバ用に IDL を提供する
- **最新バージョン**
  - WS-*（GT4 以上）に対応したバージョン： 4.1.0
  - PreWS（GT2 以上）に対応したバージョン： 2.4.1



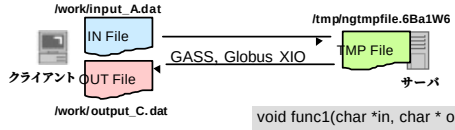
Gfarm

- グリッド上のファイルに仮想ディレクトリツリーを用いてアクセス
  - ファイルはどこかのファイルシステムに格納される
  - /gfarm にマウントしているようにアクセス可能
  - 自動複製選択により，耐故障性とアクセス集中を回避
- 最新バージョン： 1.2.9.1



Ninf-G のファイル転送機能の拡張

- Ninf-G のファイル転送
  - IN，OUT のファイルパスを RPC 実行時の引数で指定する
  - TMP ファイルは Ninf-G によって自動的に作られる




```
grpc_call(&handle,
          "/work/input_A.dat",
          "/work/output_B.dat");
```

```
void func1(char *in, char * out) {
        :
  fp1 = fopen(in, "r");
  fp2 = fopen(out, "w");
        :
```


Ninf-G のファイル転送機能の拡張

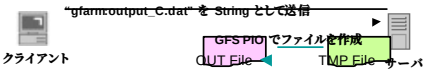
- RPC 実行時のファイル指定のパターン

No.	Input	Temporary	Output	Implementation
1	..	Local	..	No transfer
4	..	Local	Local	Implemented
	..	Local	GFS	Case 2
4	Local	Local	..	Implemented
5	Local	Local	Local	Implemented
6	Local	Local	GFS	Case 2
7	GFS	Local	..	Case 1
8	GFS	Local	Local	Case 1
9	GFS	Local	GFS	Case 1, 2
10	..	GFS	..	No transfer
11	..	GFS	Local	Case 4
12	..	GFS	GFS	Case 6
13	Local	GFS	..	Case 3
14	Local	GFS	Local	Case 3, 4
15	Local	GFS	GFS	Case 3, 5
16	GFS	GFS	..	Case 5
17	GFS	GFS	Local	Case 4, 5
18	GFS	GFS	GFS	Case 5, 6

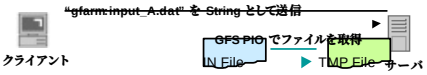
- Local：各ノードの通常のファイルシステム上のファイル
- GFS：Gfarm FS 上のファイル
- 拡張のポイント
  - ファイル型引数として Gfarm FS 上のファイルを指定する
    - この段階では転送は行わない
  - クライアントから Temporary ディレクトリを選択可能にする
- 利点
  - 遠隔プログラムを Local/GFS に関係なく作成できる

Ninf-G のファイル転送機能の拡張

- TMP/Local → OUT/GFS 転送の実装
  - デフォルトでは GFS ファイルはサーバ上に作成される
    - ただし，サーバが Gfarm のストレージノードである必要あり



- IN/GFS → TMP/Local 転送の実装
  - オンデマンド転送の有無を選択可能



Ninf-G のファイル転送機能の拡張

- API
  - ファイル型引数の記述方法の拡張

```
GFS: [ LTMP | GTMP ] : [ LFILE | GFILE ] : <file path>  ¥
                     [ : [ LFILE | GFILE ] : <file path> ]
```

GFS：後に拡張機能が続くことを示す
[LTMP | GTMP] : 一時 ファイルの Local/GFS の指定
[LFILE | GFILE] : 次の ファイルの Local/GFS の指定

  - "gfarm:exp1/input.data "をサーバに転送する場合の例

```
grpc_call (&handle,
           "GFS:LTMP:GFILE:exp1/input.data",
           "result.data");
```


Task Sequencing API ライブラリの実装

- ファイル転送の拡張機能を利用して Task Sequencing API ライブラリを作成する
- API
  - Ninf, NetSolve の API をもとにした設計する
  - grpc_function_handle はライブラリ内に隠蔽する
    - ライブラリ内での Sequence の解析と割り当て

```
grpc_begin_sequence (TMP_ON_GFS, DUPLICATION_ON);
grpc_submit ("func1", A, B);
grpc_submit ("func2", B, C);
grpc_end_sequence();
```

TMP_ON_GFS : Gfarm を利用したサーバ間の直接転送を利用
DUPLICATION_ON : 中間ファイル作成時にレプリカを作成

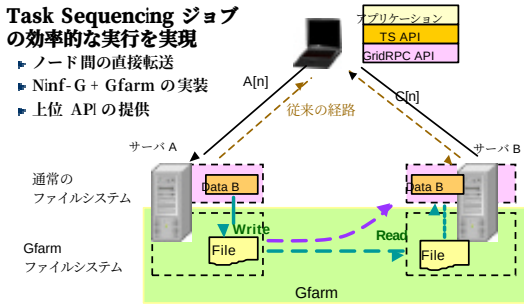
Task Sequencing API ライブラリの実装

- **ライブラリ内の処理**
  - 今回は単純なアルゴリズムで実装

1. grpc_submit() による RPC 要求を保存
2. grpc_end_sequence() の中で Sequence の解析と実行
   a. Task のサーバへの割り当て
      i. Task 1 をサーバリストの先頭にあるサーバに割り当て
      ii. Task 2 を同じサーバ, あるいはリストの上位にある別のサーバに割り当て
   b. 引数の解析（中間データの検知）
      Task[i] の OUT データと Task[i+1] の IN データを順に比較
      データ型とポインタが同一であれば中間データと判定
      中間データのファイルパスを GFS ファイルのパスに書き換え
   c. 各タスクを順次実行
   d. （中間データのクリーンアップ）

提案機能の実装のまとめ

- **Task Sequencing ジョブの効率的な実行を実現**
  - ノード間の直接転送
  - Ninf-G + Gfarm の実装
  - 上位 API の提供



評価実験

- **実験方法**
  - func1 → func2 の順に実行する Task Sequencing ジョブを投入
    - 入力／出力ファイルを要求
  - 計測項目
    - TS ジョブ全体の実行時間 -（func1とfunc2の処理時間の合計）
    - 直接転送／従来経路を利用した際のデータ転送時間
  - 提案手法との比較
    - Gfarm : Gfarm による直接データ転送（提案手法）
    - GASS : GASS を用いたデータ転送 （Ninf-G v.2.3）
    - Protocol : Globus XIO を用いたデータ転送（Ninf-G v.2.4/4.0 以降）
    - Remote object : 単一ノードでの Sequence 処理
- **実験環境**
  - AIST のノードをクライアントに利用
  - AIST，NCSA ノードをサーバに利用
  - Gfarm のメタデータサーバは SDSC のサイトで稼動

計算機環境

	AIST (client)	AIST (server)	NCSA (server)
CPU	Pentium III 1.4 GHz	Xeon 2.8 GHz	Xeon 2.0 GHz
OS	Linux 2.4.20	Linux 2.6.9	Linux 2.4.21
Globus	2.4.3 (gcc32)	3.2.1 (gcc32)	2.4.3 (gcc32pthr)
Ninf -G	2.4.0 改	2.4.0 改	2.4.0 改
Gfarm	1.2.9.1	1.2.9.1 / 1.2	1.2.9.1 / 1.2
LFS I/O read	144 (46.0)	122 (98.2)	991 (71.8)
LFS I/O write	33.3	112	22.5
GFS I/O read	-	72.6	91.9
GFS I/O write	-	92.7	21.8

※ I/O 性能の単位は MB/sec
※ NCSA のノードは NFS を経由して RAID にアクセスする

ネットワーク環境

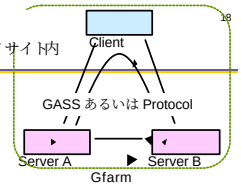
- **1GB 転送時のサイト間のネットワーク性能**
  - 未計測の部分はファイアーウォールにより計測できなかった

From	To		
	AIST (client)	AIST (server)	NCSA (server)
AIST (client)	-	59.59	0.34
AIST (server)	51.46	86.13	未計測
NCSA (server)	0.19	0.34	82.39

※ 単位は MB/sec

実験結果（1）

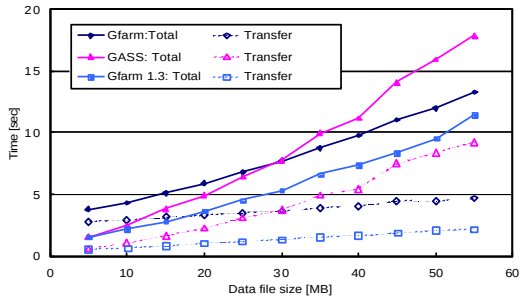
AIST サイト内



- **Gfarm のオーバーヘッド**
  - 共通鍵認証に 0.2 秒
  - メタサーバへのアクセス
  - MD5 チェックサム
  - スケジューリング

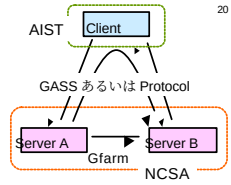
	10 MB		50 MB	
	Total-Func	Transfer	Total-Func	Transfer
Gfarm	4.31 秒	2.90 秒	11.9 秒	4.45 秒
GASS	2.51	1.10	16.0	8.41
Protocol	1.10	0.470	4.45	1.99
Remote object	1.33	0.00955	4.71	0.0467

ファイルサイズによる影響



実験結果（2）

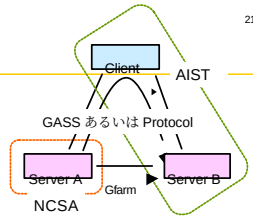
- 直接転送の効果が得られた結果
- Protocol は広域ネットワークで遅い
- 本環境では Gfarm の並列ストリームの効果は小さかった



	10 MB		50 MB	
	Total-Func	Transfer	Total-Func	Transfer
Gfarm	70.4 秒	7.82 秒	302 秒	11.8 秒
GASS	122	59.2	575	286
Protocol	181	89.7	868	439
Remote object	62.4	0.00842	291	0.0413

実験結果（3）

- 広域ネットワークでの Gfarm と GASS の転送性能の差が現れた結果



	10 MB		50 MB	
	Total-Func	Transfer	Total-Func	Transfer
Gfarm	59.0 秒	26.2 秒	219 秒	66.8 秒
GASS	62.2	29.2	295	143
Protocol	90.5	53.5	440	263

考察

- Gfarm を用いた直接転送により，Task Sequencing の実行時間を短縮できた
  - クラスタ内，かつファイルサイズが小さい時のオーバーヘッドはクライアントを経由するコストより大きい
    - メタサーバをローカルに設置
    - スケジューリングが改善された Gfarm v.1.3 を利用
- 広域ネットワークでは Gfarm の転送が高速である
- 速度面以外のメリット
  - クライアントに大きなファイルを作成できない
  - 中間データのレプリカを作成したい
  - ユーザは TS API を利用することで，中間データの所在や転送方式を気にする必要がない

まとめ

- 複数ノードでの効率的な Task Sequencing ジョブの実行を実現した
  - GridRPC の特徴を失わないこと，GridRPC システムの実装の修正を最小限に抑えることに配慮
    - Grid Filesystem を用いたアプローチ
  - Task Sequencing API を実装
    - Ninf-G のファイル転送機能の拡張
    - 拡張機能を利用した上位 API ライブラリの実装
  - 評価実験
    - サーバ間の直接データ転送による速度向上
  - 速度面以外の利点
    - 耐障害性，上位インタフェース

今後の課題

- ファイル型引数以外に適用
  - アプローチ：データを 1 度，GFS ファイルに落とす
    - データハンドルを定義して RPC 実行時の引数として渡す
      - データハンドルを他の用途でも使いたいという要求
    - 完全に GridRPC システム内部に隠蔽する
- 実アプリケーションを用いた評価
  - Sequence の解析と Task の割り当てアルゴリズムを改良
- 実装のチューニング
  - 一時ファイルを GFS に作成することによる速度向上
  - システムコールフックライブラリでの実装