1T-6

GridRPC アプリケーションデバッグ支援ツール

小林 孝嗣 † 渡邊 啓正 ‡ 本多 弘樹 ‡
† 電気通信大学 電気通信学部 ‡ 電気通信大学 大学院情報システム学研究科

1. はじめに

近年，広域ネットワークに接続された計算資源および情報資源を統合し，ユーザが容易に利用できる環境を構築するグリッドに関する研究が盛んに行われている．現在では，グリッド環境を構築するためのミドルウェアとして Globus Toolkit[1] が事実上標準として用いられている．また Ninf-G[2][3] や NetSolve[4] のように，従来の RPC をグリッドに対応させた GridRPC[5] に関する研究も盛んである．

本研究では，GridRPC を用いたアプリケーション開発におけるプログラマのデバッグおよびチューニングの手間を軽減するための支援ツールを開発した．なお，本研究では Ninf-G を用いた GridRPC アプリケーションの開発を支援の対象としてツールを実装している．

2. デバッグおよびチューニング時の問題点

現状ではプログラマが必要な情報を出力するコードを記述するなどして GridRPC プログラムのデバッグおよびチューニングを行っている．しかし，これらの作業は非常に煩雑で手間のかかるものである．特に利用する計算機の数が多い場合には，出力される情報の数や量も膨大なものとなってしまい，効率の良いデバッグおよびチューニングが行えるとは言い難い．

他にデバッグおよびチューニングを困難にしている要素としては以下のようなものが挙げられる．

- GridRPC システムによるリモートライブラリの起動等の処理時間が，ネットワーク状況や計算機の負荷状況の影響を受けやすい．
- プログラムが多数の計算機上で動作している．

そこで本研究では GridRPC を用いたプログラムのデバッグおよびチューニングを支援するツールを開発した．本ツールでは，RPC 実行に関する情報の自動取得，取得した情報の可視化等を行なう．

3. ツールの設計および実装

3.1 支援ツールの構成

本ツールでは，2 章で述べた問題点を解決するため，次のような条件を満たすことを目標とした．

- 実行時間や RPC 回数などのプログラム全体の情報が取得できる．
- 個々の計算機や RPC の情報が取得できる．
- ツールを利用するまでの手間が少ない．
- 膨大な数や種類の情報の中から，自分の必要な情報だけを容易に取得できる．

3.2 支援ツールの動作概要

本ツールの動作概要は以下のとおりである (図 1)．

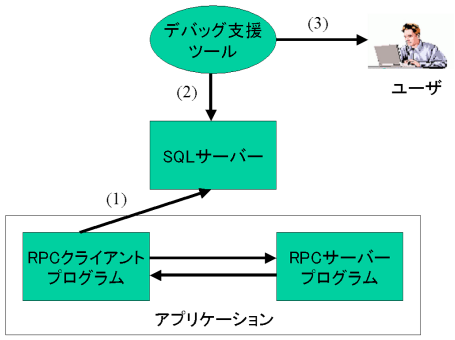

図 1 ツールの動作概要

(1) 本ツールの情報収集機構が RPC 実行情報を収集し，収集した情報をデータベースに格納する．
(2) 可視化プログラムがデータベースに格納された RPC 実行情報を取得する．
(3) 可視化プログラムが RPC 実行情報を可視化する．

3.3 RPC 実行情報の収集と格納

プログラマが RPC 実行情報収集のために行なう作業は，クライアントプログラムにおいて本ツールが提供するヘッダファイルをインクルードするだけである．

また，本ツールでは収集した情報を SQL サーバに格納することにした．その理由は，本ツールが RPC 実行情報以外にも以下に示すような情報を収集するため，テーブル間でリレーションを取れると検索の効率が良いためである．

- CPU やメモリなどのサーバ側の計算機性能
- RPC 実行の履歴
- サーバの管理者情報

3.4 収集した情報の可視化

収集した情報の可視化は Java で実装した GUI ツールによって行なう．このツールは，SQL サーバに対して問い合わせを行ない，取得した RPC 実行情報を可視化してユーザに提示する．このツールでは主に次の機能を提供する．

- 実行時間や RPC 回数などのプログラム全体の情報を提供する機能

Debug assistance tool for GridRPC application
Takatsugu Kobayashi†
Hiromasa Watanabe‡
Hiroki Honda‡
†The University of Electro-Communications
‡Graduate School of Information Systems,
The University of Electro-Communications

- 実行環境を構成している計算機や RPC 実行状況を可視化により容易に把握できるようにする機能
- 個々の計算機や RPC の情報を提供する機能
- RPC に失敗したサーバおよび失敗した RPC を強調表示して容易に把握できるようにする機能
- 計算機の情報や RPC の情報が欲しいときに，多数の情報の中から指定した条件に合うものだけを絞り込んで表示する機能

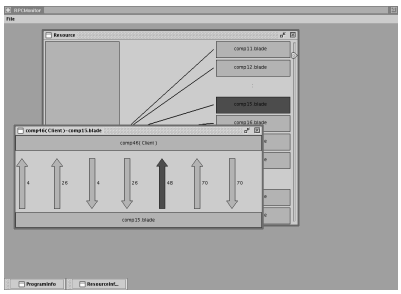

図 2　可視化ツールの動作画面

4. 評 価 実 験

本ツールの評価実験を，24 台の計算ノードと 1 台の管理ノードからなるクラスタ環境上で行なった．ここでは各ノードをグリッド環境上の 1 つの計算機と見立てて動作させている．計算ノードのうち 1 台をクライアントとして，残り 23 台を RPC サーバとして，管理ノードを SQL サーバとして用いた．

4.1 実行時オーバーヘッドの評価

モンテカルロ法を用いて円周率を計算するプログラムを，RPC 実行回数を変えながら動作させてツール利用時の実行時オーバーヘッドを測定した．なお，この評価では RPC を非同期に実行した．表 1 は情報を収集しない場合と収集した場合のプログラム全体の実行時間および実行時オーバーヘッド (情報収集ありの実行時間－情報収集なしの実行時間) を比較したものである．

表 1　実行時間の比較

RPC 実行回数 (回)	情報収集なし (秒)	情報収集あり (秒)	実行時オーバーヘッド (秒)
1	10.93	11.00	0.07
4	14.84	14.97	0.13
16	21.21	21.62	0.41
64	122.27	121.05	-1.22

4.2 有用性評価

行列 A，B，C に対して A × B+C の計算を行なうプログラムを動作させる．このプログラムでは行列積計算を行なう mmul，行列和計算を行なう madd の 2 つの RPC が実行されるが，madd のルーチンにバグを混入させ，特定の引数の場合に無限ループに陥るようにしてある．本実験では，このプログラムにおける問題の原因究明作業にツールを用いた場合と用いない場合での手間の比較を行なった．ここではデバッグ時のコードの追加量と所要時間を手間の目安としている．この評価の結果は表 2 のとおりである．

表 2　ツール使用の有無による手間の比較

	コード追加量 (行)	所要時間 (分)
ツール使用	8	12
ツール不使用	40	40

4.3 考　　察

表 1 からツール利用による実行時オーバーヘッドはプログラム全体の実行時間に比べれば殆ど無視できる程度であるとわかった．また，表 2 からツールを利用することでプログラマの手間は軽減されたと判断できる．

表 1 でオーバーヘッドの値がマイナスになっている部分がある．これは GridRPC を用いたプログラムの実行時間が，実行のたびに変動しやすい為であると考えられる．この傾向は RPC 実行回数が多くなるほど顕著になる．

この他にもツールを利用することに次のような利点が確認できた．

- 提供すべき情報が増えても可視化によって容易に把握ができる．
- ヘッダファイルを外すだけでツール適用前の状態に戻すことができる．

5. お わ り に

本研究では，GridRPC アプリケーションの開発における，プログラマのデバッグおよびチューニングの手間を軽減するための支援ツールの提案，開発および評価実験を行なった．本研究における今後の課題は次のとおりである．

- サーバ側の情報収集機能の実装
- 大規模環境への対応
- 可視化ツールのインタフェースの改善
- 異なるアプリケーションによる評価

参 考 文 献

1) Globus Toolkit: http://www.globus.org/.
2) The Ninf Project: http://ninf.apgrid.org/.
3) 田中良夫, 中田秀基, 朝生正人, 関口智嗣: Ninf-G2: 大規模環境での利用に即した高機能，高性能 GridRPC システム, 情報処理学会研究報告 2003-HPC-95, pp. 89–94 (2003).
4) NetSolve: http://icl.cs.utk.edu/netsolve/.
5) K.Seimour, H.Nakada, S.Matsuoka, Jack Dongarra, C.Lee and H.Casanova: Overview of GridRPC: A Remote Procedure Call API for Grid Computing, *Proceedings of Grid Computing - Grid 2002*, pp. 274–278 (2002).