2007－EVA－20 (4)
2007／3／16

グリッドアプリケーションの実行時間を短縮するための計算結果再利用機構の評価

松尾勝則[†] 川崎康博[†] 水谷泰治[††]
伊野文彦[†] 萩原兼一[†]

本稿では，グリッド上で計算結果を再利用するための機構の評価について述べる．この機構は，計算結果をユーザ間で共有することにより計算の実行を回避し，グリッドアプリケーションの実行時間を短縮する．従来のものと比較して，開発した機構は階層的なデータベース構造や計算実行時の再利用などを提供する．評価では，計算結果の保管場所あるいは複製の有無に関していくつかの方針を与え，再利用による時間短縮の効果を調べる．計算結果がWAN上にある場合，およそ8MBの計算結果を16台構成のPCクラスタ上で再利用するために5秒程度を要した．一方，計算結果がLAN上にある場合，ほぼ同じ時間で128MBの計算結果を再利用できた．

Evaluating Data Reuse Framework for Reducing Execution Time of Grid Applications

Katsunori Matsuo,[†] Yasuhiro Kawasaki,[†]
Yasuharu Mizutani,[††] Fumihiko Ino[†] and Kenichi Hagihara[†]

This paper presents an evaluation of a data reuse framework capable of sharing computational results in grids. This framework avoids computation by sharing computational results with users in order to reduce execution time of grid applications. As compared to prior frameworks, the developed framework provides a hierarchical database structure and a runtime data reuse capability. In the evaluation, we investigate time reduction effects of our framework with different policies in terms of locations and replications of computational results. If computational results are available on WAN, it takes about five seconds to reuse 8-MB data on a cluster of 16 PCs. On the other hand, it takes almost the same time to reuse 128-MB data if the results are available on LAN.

1. はじめに

近年，複数の組織間で計算資源を統合し，仮想的な高性能計算機として動作させるグリッド[1]に関する研究が盛んである．例えば，一部の研究者はグリッド上に仮想組織を構築し，組織内で計算資源を共有している．一般に，このような仮想組織は共通の目的を持つ研究者で構成されることが多い．したがって，複数のユーザが同一の計算を繰り返し実行し得る．

このような同一計算の実行を回避するために，一部のグリッドシステムはデータ再利用機構を持つ．例えば，量子化学[2]，最適化[3]，医用画像処理[4]あるいは汎用のワークフローシステム[5]において，再利用機構が有用な機能として実装されている．この機構により，過去に実行されていない計算のみを実行対象にでき，効率のよい運営を支援できる．具体的には，仮想組織内で既実行の計算に対しては，その計算を実行する替わりに，蓄積済の計算結果をユーザに返す．これにより，貴重な計算資源を未踏の計算にのみ使用できるだけでなく，計算を省くことによる時間短縮を実現できる可能性もある．

多くの既存システムは，計算結果を単一のデータベース（DB）サーバに保管し，その複製をグリッド内に配置することにより計算結果を提供している．また，一部のシステム[3]はグリッド技術を用い，複数のDBサーバ上に計算結果を分散保持している．しかし，これらのシステムは，WAN（Wide Area Network）を介してDBサーバを提供しているために，再利用

† 大阪大学大学院情報科学研究科コンピュータサイエンス専攻
Department of Computer Science, Graduate School of Information Science and Technology, Osaka University
†† 大阪工業大学情報科学部情報システム学科
Department of Information Systems, Faculty of Information Science and Technology, Osaka Institute of Technology

時のオーバヘッドが大きい．特に，計算結果のサイズが大きく，DB サーバから計算ノードまでの通信性能（帯域および遅延）が低い場合，再利用による時間短縮は見込めない．

そこで本研究では，計算結果の保管場所および複製の有無に関していくつかの保管方針を与え，各々の方針のもとで再利用による時間短縮の見込みを評価する．また，再利用に要する時間だけでなく，登録に要する時間の観点から各方針の長所および短所を整理する．

以降では，まず典型的な機構における再利用の手順をまとめる（2 章）．次に，本研究で評価対象とするいくつかの保管方針を提案機構とともに示す（3 章）．その後，実環境における計測値を基に，各方針を評価する（4 章）．最後に本稿をまとめる（5 章）．

2. 既存の再利用機構

図 1 に，再利用機構を含めた典型的なグリッドの全体像を示す．この全体像は 5 つの部分からなる．それらは（1）グリッドを構成する計算ノード群（複数のクラスタ），（2）1 式のクラスタを WAN に接続するゲートウェイ，（3）グリッドへ計算を投入するクライアント PC，（4）グリッド資源に計算を割り付けるブローカ，および（5）計算結果を蓄積する DB サーバである．なお，DB サーバの複製により，機構のスケーラビリティを向上できる．

DB サーバは，下記に挙げる 3 種類の属性を計算のインスタンスごとに保管する．

- 属性 1：計算の固有名（プログラム名や関数名）
- 属性 2：計算への入力
- 属性 3：計算結果

これらの属性を持つ DB を基に，計算結果の登録および検索の操作を DB サーバは提供する．

ブローカにおける再利用の手順を以下にまとめる．なお，説明を容易にするために，下記の手順では計算全体（ジョブ）を再利用の単位とする．

（1） 計算の待ち受け．クライアント PC からの計算依頼（ジョブ）を待つ．

（2） 計算結果の検索．ブローカは，ジョブをグリッド資源に割り付ける前に，計算結果の有無を DB サーバに問い合わせる．この際，属性 1 および 2 の値を検索クエリに含めて転送する．

（3） 計算結果の取得．
- 計算結果がある場合．計算結果を DB サーバより受信し，計算の実行を回避する．
- 計算結果がない場合．計算をグリッド資源に割り付け，その実行結果を回収する．

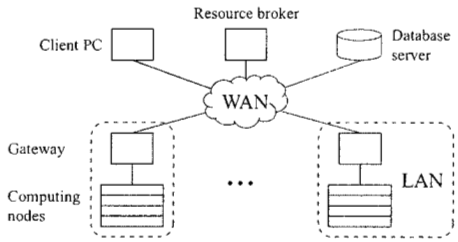


図 1 既存の再利用機構

（4） 計算結果の提示．ユーザへ計算結果を返す．

（5） 計算結果の登録．将来の再利用のために，新規に得た計算結果を DB サーバへ登録する．

なお，ワークフローシステム[5] は，より細かい粒度の再利用を提供している．具体的には，ジョブを構成する一連の計算のうちの一部（プログラム単位）を再利用できる．この場合，DB サーバとやりとりするのはブローカだけではない．例えば，計算ノードが計算結果を要求し，それを入力として未実行部分の計算を開始することもある．特に，グリッドミドルウェアとして Globus Toolkit[6] を用いれば，その一部である GridFTP[7] の第三者転送機能により，ブローカを介することなく計算ノードへ計算結果を転送できる．

3. 提案する再利用機構

提案する再利用機構は，既存機構からの差分として以下の特長を持つ．

- 階層的な DB 構造．本機構は計算結果を分散保持して DB 構造を階層化する．階層の上位に位置するサーバは，メタサーバとして下位の情報を集約保持する．計算ノードが計算結果を検索する場合，下位から上位に向けて段階的に解決することを試みる．これにより計算ノード自身が計算結果を保管し得るため，迅速に再利用できる可能性がある．
- ディレクトリ情報と計算結果の分離．新規に GridFTP サーバを構築し，計算結果をファイルとして提供する（**図 2**）．一方，DB サーバは，計算結果の保管場所（ホスト名およびファイルパス）をディレクトリ情報として提供する．計算結果をその有無に関する情報と分離し，第三者転送と併用することにより，DB サーバへの通信を量的に緩和できる．
- 計算実行時の再利用．本機構は，実行前の段階での再利用に加え，実行時における関数単位の再利用を提供する．後者の場合，プログラム実行時に計算ノードが GridFTP サーバや DB サーバとやりとりする．やりとりの間，計算資源は待ち状態

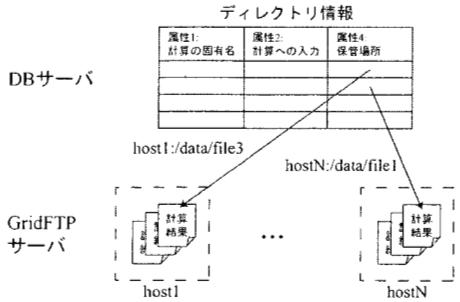


図 2　ディレクトリ情報と計算結果の分割

になるため，迅速な再利用が必要である．

- 登録時の通信量削減．計算結果の登録は属性 1〜3 の値を必要とする．これらのうち，検索の段階で転送済の属性 1 および 2 は，登録の段階で再度転送する必要はない．これらの属性値を DB サーバ側で保存しておけば，登録時には計算結果を転送するだけでよい．

図 3 に，提案する再利用機構の概要を示す．従来，WAN 上に存在した DB サーバは，メタサーバとして機能する．一方，提案機構における DB サーバは，後述する保管方針により，WAN 上のメタサーバ（従来通り），LAN 上のゲートウェイもしくは計算ノードのいずれかが担当する．GridFTP サーバも同様である．

なお，計算実行時の再利用では，計算の大域状態のみを対象とし，計算ノードごとの局所的な再利用は扱わない．ゆえに，計算ノードの代表が DB サーバとやりとりし，そこで得られた計算結果を残りの計算ノードに放送するような再利用状況を対象にする．

3.1　計算結果の保管方針

提案する機構は，3 層からなる階層構造を持つため，各々の階層に計算結果を保管する方針が考えられる（図 3）．なお，説明を容易にするために，ここでは複製はないものとする．

- 方針 1：メタサーバが持つ．他は何も持たない．既存機構そのものである．
- 方針 2：ゲートウェイが持つ．メタサーバはディレクトリ情報を集約して持ち，計算結果を保持しない．計算ノードは何も持たない．
- 方針 3：計算ノードが持つ．メタサーバは方針 2 と同様である．ゲートウェイは，担当する計算ノードのディレクトリ情報のみを集約して持つ．

なお，3 章で述べた通り，計算ノードの検索要求は下位から上位階層へ段階的に解決する．一方，ブローカの検索要求は従来通りメタサーバが解決する．

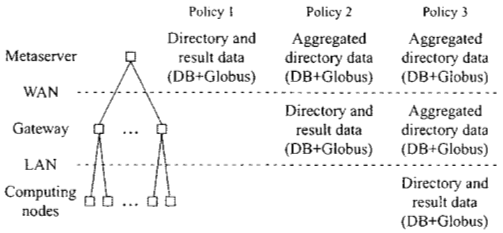


図 3　保管方針の一覧

3.2　方針 1：メタサーバが計算結果を持つ

方針 1 における各操作のオーバヘッドをモデル化する（**表 1**）．まず，LAN および WAN においてサイズ d のデータ転送に要する時間を $L(d)$ および $W(d)$ とおく．また，DB サーバにおける登録時間および検索時間を各々 R および S とする．

仮に，N 台の計算ノードのうちの 1 台がサイズ q の検索クエリをメタサーバに転送し，サイズ r の計算結果を得たとする．このとき，計算ノードは計算結果を得るまでに $L(q)+W(q)+S+W(r)+L(r)$ を要する．その後，計算結果を N 台に放送する際の時間は，2 分木状に転送を繰り返せば $L(r)\log N$ である．これらを合わせ，再利用に要する時間は表 1 の T_2 となる．なお，放送にミリネット[8] などの並列計算用の高速ネットワークを用いれば，$L(r)\log N$ の値を相対的に小さくできる．

一方，計算結果がない場合，その旨を計算結果の替わりに返す必要がある．この通知にサイズ 1 のメッセージを転送すれば，再利用の失敗に起因するペナルティ時間 T_1 は $L(q)+W(q)+S+W(1)+L(1)$ となる．さらに，計算結果の登録時間は，LAN，WAN および DB サーバ上の経過時間の和 $L(r)+W(r)+R$ となる．

最後に，方針 1 における複製は，メタサーバの数を増やすことを意味する．したがって，複製はスケーラビリティを向上できる．さらに，通信性能のよいメタサーバを計算ノードが選択できる場合，$W(d)$ を削減する効果もある．

3.3　方針 2：ゲートウェイが計算結果を持つ

図 4 に，方針 2 における再利用の手順を示す．方針 1 と比較すると，直属のゲートウェイに計算結果がある場合，検索クエリおよび計算結果を WAN 上に流さなくてよい．したがって，この場合，T_2 を $T_2-W(q)-W(r)$ に短縮できる．それ以外の場合，階層化が原因で DB サーバの検索回数が 1 回多い（T_1+S）．さらに，直属でないゲートウェイ上に計算結果がある場合，そのゲートウェイにメタサーバが第三者転送を依頼する必

表 1 計算結果の保管方針

方針	保管場所	操作	計算結果の有無	オーバヘッド
1	メタサーバ	再利用	なし	$T_1 = L(q) + W(q) + S + W(1) + L(1)$
			あり	$T_2 = L(q) + W(q) + S + W(r) + L(r) + L(r)\log N$
		登録	—	$L(r) + W(r) + R$
2	ゲートウェイ	再利用	なし	$T_1 + S$
			LAN 上にあり	$T_2 - W(q) - W(r)$
			WAN 上にあり	$T_2 + W(1) + S$
		登録	—	$L(r) + W(1) + 2R$
3	計算ノード	再利用	なし	$T_1 + 2S$
			計算ノードにあり	$T_2 - W(q) - W(r) - L(q) - L(r)$
			LAN 上にあり	$T_2 - W(q) - W(r) - L(r) + L(1) + S$
			WAN 上にあり	$T_2 + W(1) + 2S$
		登録	—	$L(1) + W(1) + 3R$

q：検索クエリのデータサイズ　　r：検索結果のデータサイズ
$L(d)$：LAN におけるサイズ d のデータ転送時間　　$W(d)$：WAN におけるサイズ d のデータ転送時間
S：DB サーバにおける検索時間　　R：DB サーバにおける登録時間
N：計算ノードの数

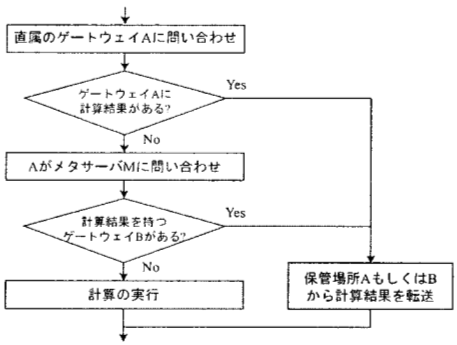


図 4 方針 2（ゲートウェイが計算結果を持つ場合）の再利用手順

要がある．この依頼にサイズ 1 のメッセージを転送すれば，検索時間は $T_2 + W(1) + S$ となる．以上より，増加部分 S が減少部分 $W(q) + W(r)$ よりも十分に短い場合，方針 1 よりも時間短縮の観点で有利である．

一方，登録時には計算結果の替わりにディレクトリ情報を WAN 上に流せばよい．ディレクトリ情報のサイズを 1 とみなせば，方針 1 における $W(r)$ の項は $W(1)$ に削減できる．ただし，DB サーバにおける登録回数が 1 回多いため，登録オーバヘッドは $L(r) + W(1) + 2R$ となる．

方針 2 における複製は，メタサーバあるいはゲートウェイの複製を意味する．前者の効果は方針 1 と同一である．一方，後者の解釈は 2 通りある．まず，LAN 内のゲートウェイを増やす場合，同一 LAN に複数のゲートウェイを設置すること自体が現実的でない．次に，ゲートウェイ間で情報を複製する場合，LAN 内の転送のみで計算結果を再利用できる可能性が高くなる．しかし，複製の増加とともにメタサーバへの問い合わせが不要になり，階層化の存在意義がなくなる．

なお，方針 2 はゲートウェイにおいて DB サーバおよび GridFTP サーバを必要とする．したがって，方針 1 よりも構築に要する作業量は多いが，後述する方針 3 と比較して許容できる範囲であると考える．

3.4 方針 3：計算ノードが計算結果を持つ

図 5 に，方針 3 における通信の様子を示す．方針 2 と比較すると，計算ノード自身に計算結果がある場合（図 5(a)），検索クエリおよび計算結果を転送しなくてよい．したがって，この場合，再利用に要する時間を T_2 から $T_2 - W(q) - W(r) - L(q) - L(r)$ に短縮できる．一方，上記以外の場合，階層化により DB サーバの検索回数が 1 回多い．したがって，方針 2 の各オーバヘッドに S を加えた値となる．ただし，直属のゲートウェイ上に検索結果がある場合（図 5(b)），計算結果の替わりにディレクトリ情報を計算ノードに転送すればよいため，転送時間を方針 2 よりも削減できる．最後に，登録時には LAN 内の転送量を削減できる替わりに DB サーバへの登録回数が増えるため，登録オーバヘッドは $L(1) + W(1) + 3R$ となる．

方針 2 と同様に，方針 3 における複製の効果について考える．計算ノード以外の複製は，方針 1 および 2 で述べた通りである．一方，計算ノードの複製は，方針 2 と同様に解釈が 2 通りある．まず，同一 LAN 内で複製する場合，転送なしで計算結果を再利用できる可能性が高くなる．さらに，すべての計算ノードが同じ計算結果を持つため，放送が不要である．しかし，複製の増加とともに階層化の意義が薄れる．次に，LAN 間で複製する場合，これらの特性はさらに強まる．しかし，グリッド上のすべての計算ノードが同じ計算結果を持つ必要があるため，複製のための通信量が多く，現実的ではない．

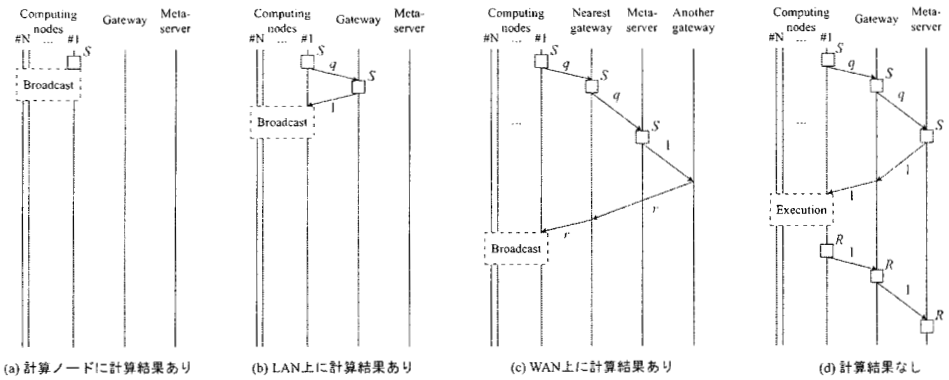


図 5 方針 3 における通信の様子

なお，方針 3 では WAN 上のメタサーバから LAN 内の計算結果を転送できる仕組みが必要である．したがって，再利用と関連のない計算を実行している計算ノードがデータ転送を依頼される可能性がある．この外乱は，計算性能を低下させるだけでなく，再利用に要する時間を長くする．さらに，外部から LAN 内へのアクセスを許可することは，セキュリティの観点で好ましくない．一方，これらの問題は方針 1 および 2 では生じない．最後に，方針 3 ではすべての計算ノード上に DB サーバおよび GridFTP サーバを構築する必要があり，作業量が多い．

4. 評価実験

本章では，実際のグリッド環境において再利用オーバヘッドを計測した結果を示し，各保管方針の優劣について考察する．

4.1 実験環境

図 6 に，実験に用いたグリッド環境を示す．計算ノード群として大阪大学のクラスタ（16 台構成）を用い，メタサーバとして大阪工業大学の PC を用いた．計算ノードは CPU として Xeon 2.8 GHz を持ち，並列計算用のミリネット（2 Gb/s）および TCP/IP 通信用のイーサネット（1 Gb/s）で相互接続されている．なお，大阪大学側のゲートウェイは計算ノードと同じ構成の PC である．大学間を結ぶ WAN の通信性能は，実測で帯域が 25Mb/s および遅延が 13 ミリ秒である．帯域および遅延は，各々scp および traceroute コマンドで計測した．

グリッドミドルウェアとして Globus Toolkit 4[6] を用い，DB としては MySQL[9] を用いた．また，計算結果の転送には GridFTP[7] を用い，クラスタ内のプログラム実行には SCore[10] を用いた．

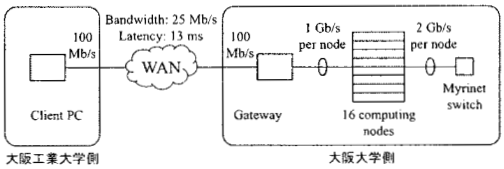


図 6 実験環境

表 2 位置合わせにおける計算時間と計算結果のサイズ

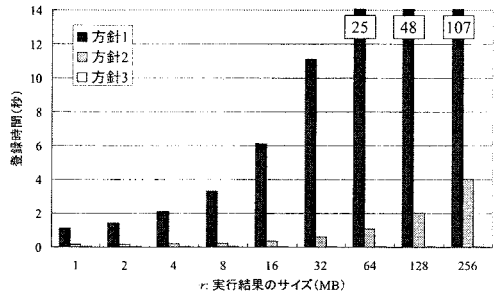
階層	計算時間（秒）	サイズ（KB）
H1	2.4	58
H2	14.6	408
H3	85.1	3072
H4	326.8	3072
H5	91.4	3072

実験では，再利用が有用なアプリケーションとして，非剛体位置合わせ[11] に着目した．このアプリケーションは MPI[12] を用いて並列化されている．再利用のための検索クエリのサイズ q は 52 バイトである．**表 2** に，再利用対象とする計算の実行時間およびその計算結果のサイズ r を示す．なお，表中の階層は計算の進行状況に対応していて，階層ごとの計算を再利用できる可能性がある．詳細は文献 4) を参照されたい．

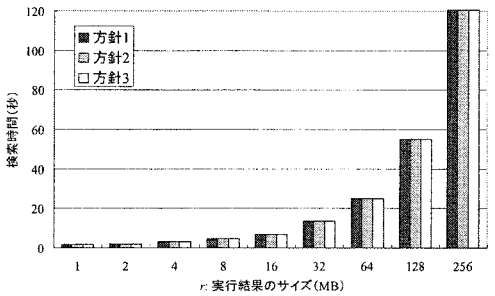
4.2 ペナルティ時間（再利用失敗）の評価

図 7 に，グリッド上に計算結果がない場合の検索時間（ペナルティ時間）を保管方針ごとに示す．実験では，非剛体位置合わせにおける再利用を想定し，検索クエリのサイズ q は 52 バイトとした．

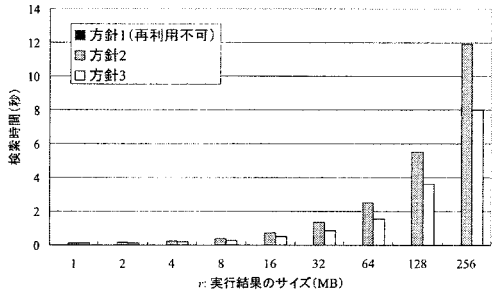
図 7 より，いずれの方針もおよそ 0.3 秒程度でグリッド上に計算結果がないことを計算ノードに知らせている．これらのペナルティ時間は，表 2 に示す計算時間 2.4〜326.8 秒と比較して十分に短い．つまり，再利用対象とする計算と比較してペナルティ時間は短い．



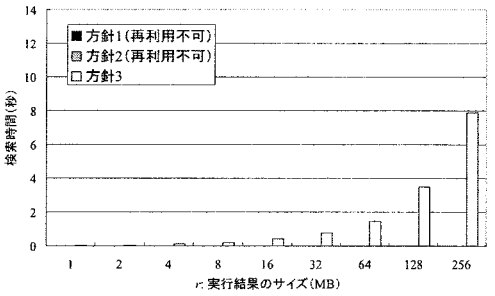
(a) 登録時間



(b) 検索時間（WAN 上で再利用）



(c) 検索時間（LAN 上で再利用）



(d) 検索時間（計算ノードで再利用）

図 8 登録時間および検索時間

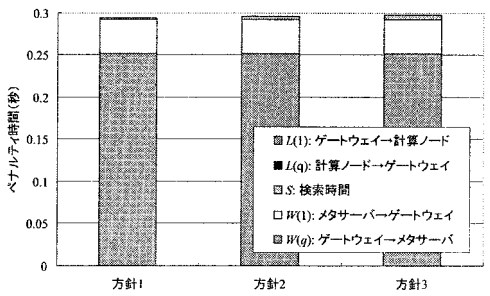


図 7 再利用失敗時のペナルティ時間およびその内訳

一般に，再利用による時間短縮効果は，対象とする計算時間の長さとともに向上することからも，これらのペナルティ時間は十分に小さいと考えられる．まとめると，ペナルティ時間の観点からは，いずれの手法も大きな差異はない．

なお，ペナルティ時間の内訳を分析すると，WAN 上の転送時間 $W(q)+W(1)$ が大半を占める．ペナルティ時間の 83%および 16%を各々$W(q)$ および $W(1)$ が費やしている．結果，表 1 が示すように，階層化により方針 1〜3 の順に DB における検索時間 S が増えたとしても，全体の時間はさほど変わらない．実際に，S はわずか 2 ミリ秒であった．なお，$W(q)+W(1)$ が大半を占めることから，検索クエリのサイズ q を削減する手法[4] がペナルティ時間を抑えるために重要であることも分かる．

4.3 登録時間の評価

図 8(a) に，各保管方針における計算結果の登録時間を示す．4.2 節と同様に，サイズ q が 52 バイトであると仮定し，計算結果のサイズ r を 1〜256MB まで変えながら登録時間を計測した．

図 8(a) より，方針 1 では計算結果が 16MB を超える場合，登録に 10 秒以上を要しているが，他の方針では 256MB の計算結果に対しても 4 秒以下で登録できている．一方，表 1 の解析では，方針 1〜3 の順に，計算結果の転送時間を削減できる替わりに DB サーバ上の登録時間 R が増えることを示している．しかし，実際には R は 0.7 ミリ秒程度であるため，転送時間の削減が登録時間全体の短縮を引き出せた．

なお，3章で述べた通り，提案機構は登録時の通信量を削減している．この削減により転送時間をおよそ210ミリ秒ほど削減でき，r が比較的小さい値のときに有効である．例えば，方針2において r が1MBおよび32MBのとき，それぞれ登録時間の60%および26%を削減できた．

4.4 検索時間の評価

図8(b)〜(d)に，グリッド内に計算結果が存在する場合の各保管方針における検索時間を示す．ここで，検索時間とは計算ノードの代表が検索クエリを送信して全計算ノードが計算結果を受信するまでの時間を指す．図の各々は，メタサーバ，ゲートウェイおよび計算ノード上に計算結果がある場合の結果である．

まず，メタサーバに計算結果がある場合，図8(b)が示すように，どの方針もほぼ同じ検索時間を実現している．この実測結果は，表1の結果と合致する．具体的には，方針2および3は，方針1における検索時間 T_2 よりも各々 $W(1)+S$ および $W(1)+2S$ ほど長い．一方，今回の実験環境では，$W(1)$ は40ミリ秒であり，S は2ミリ秒である．したがって，これらの増大は検索時間の全体と比較して十分に小さく，方針間に大きな差異は生じなかった．

次に，ゲートウェイに計算結果がある場合について述べる（図8(c)）．方針1は，計算ノードに計算結果を保管しないため，この状況は発生し得ない．一方，残りの方針では，r が64MB以下であれば，長くとも数秒で計算結果を再利用できている．ただし，方針2および3の差は r が64MB以下のときに1秒以内であるが，r の増大とともに広がり，256MBのときに4秒に到達している．この原因はゲートウェイおよび計算ノード間の通信量にある．方針3がディレクトリ情報のみを転送する一方，方針2は計算結果を転送するため，この差は r の増大とともに広がる．

最後に，方針3では計算ノードが計算結果を持ち得る（図8(d)）．このときの検索時間は図8(c)とほぼ同じである．つまり，計算結果がLAN内にある場合，保管場所（計算ノードもしくはゲートウェイ）に関わらず，検索時間が変わっていない．この理由は計算結果の放送にある．これらの場合では，放送が検索時間の大半を占める．例えば，r が32MB以上のとき，検索時間の90%以上を放送に費やしている．したがって，計算ノード自身が計算結果を保持したとしても，検索時間は短縮しなかった．すべての計算ノードが計算結果を持てば放送は不要であるため，さらなる時間短縮は複製を必要とする．

なお，5秒程度の検索時間を達成するためには，WAN上に計算結果がある場合は方針に関わらず r が8MB以下である必要がある．一方，LAN上に計算結果がある場合，方針1を除いて r が128MB以下である必要がある．また，方針3において計算ノードに計算結果がある場合，r は128MB以下である必要がある．したがって，時間短縮効果の観点からは，方針2および3に大きな差異はない．しかし，3.4節で述べた通り，方針3はシステム構築のための作業量やセキュリティに課題がある．したがって，これらの観点でバランスのよい方針2がよい．

4.5 非剛体位置合わせへの適用

図9に，非剛体位置合わせにおいて計算結果を再利用したときの実行時間の推測結果を示す．このアプリケーションは5つの階層H1〜H5で構成されているため，各階層における再利用の成否を組み合わせ，方針2の検索時間および登録時間を基に実行時間を推測した．なお，表中の段階L0およびL5は，すべての階層において再利用に失敗する場合および成功する場合に対応する．また，L0における実行時間は，計算結果の登録時間を含んでいる．

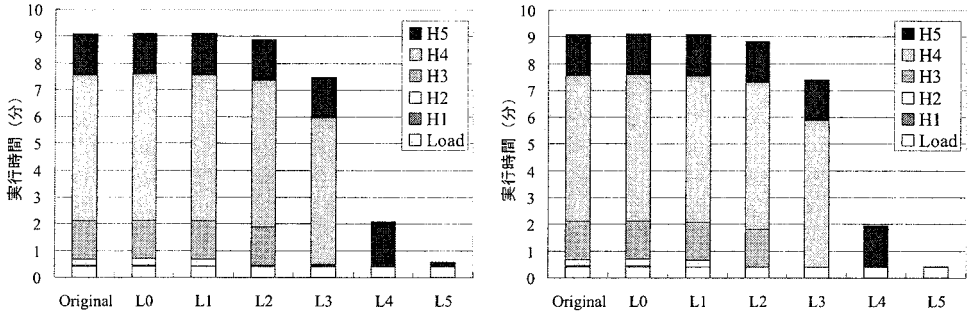
すべての階層において再利用に失敗し，計算結果をその都度登録したとしても（段階L0），実行時間は元のものとほぼ変わらない．実際に，L0におけるペナルティ時間は2秒程度であり，9分の実行時間と比較して十分に小さかった．この理由としては，q および r が3MB以下であり，十分に小さいことが挙げられる．このことは，図9(a)および(b)の結果がほぼ同じであることの原因でもある．つまり，段階L5を除けば，計算結果の保管場所に関わらず，実行時間はほぼ同じである．L5では，WAN上もしくはLAN上での再利用により，9分の実行時間を各々35秒および25秒に短縮できた．

5. ま と め

本稿では，グリッド上で計算結果を再利用するための機構の評価を示した．従来のものと比較して，提案機構は階層的なDB構造や計算実行時の再利用などを提供する．評価では，計算結果の保管場所および複製の有無に関していくつかの方針を与え，再利用による時間短縮の効果を実環境上で調べた．

実験の結果，計算結果をメタサーバ，ゲートウェイあるいは計算ノードに置く方針1〜3に関して下記の3点が分かった．

- ペナルティ時間：方針間に差異がないこと．
- 登録時間：時間の長い方針1を除けば，方針2および3はほぼ同じであること．



(a) WAN 上で再利用　　(b) LAN 上で再利用

図 9　非剛体位置合わせにおける再利用効果の推測（方針 2）

- 検索時間：(1) WAN 上で再利用できる場合，差異がないこと．(2) LAN 上で再利用できる場合，方針 1 が長く，残りはほぼ同じであること．(3) 計算ノード自身が再利用できる場合，放送が原因で (2) の場合と変わらないこと．

結果として，ゲートウェイ上で計算結果を保持し，メタサーバ上でそれらのディレクトリ情報を集約保持する方針 2 が性能および構築コストの観点で望ましいと考える．例えば，方針 2 において計算結果が WAN 上にある場合，およそ 8MB の計算結果を 16 台構成の PC クラスタ上で再利用するために 5 秒程度を要した．一方，計算結果が LAN 上にある場合，ほぼ同じ時間で 128MB の計算結果を再利用できた．

今後の課題としては，再利用の頻度を考慮して計算結果の保管場所を決定することが挙げられる．

謝辞　本研究の一部は，科学研究費補助金基盤研究 (B) (2) (18300009)，特定領域研究 (17032007) および若手研究 (B) (17700060) の補助による．

参 考 文 献

1) Foster, I.: What is the Grid　A Three Point Checklist (2002). http://www-fp.mcs.anl.gov/%7Efoster/Articles/WhatIsTheGrid.pdf.
2) Nishikawa, T. and et al.: Design and Implementation of Intelligent Scheduler for Gaussian Portal on Quantum Chemistry Grid, *Proc. 3rd Int'l Conf. Computational Science (ICCS'03), Part III*, pp.244–253 (2003).
3) 廣安知之他：Grid 環境における評価部に個体データベースを用いた遺伝的アルゴリズムの提案，情報処理学会研究報告，2002-HPC-91, pp.79–84 (2002).
4) Ino, F. and et al.: Minimizing Data Size for Efficient Data Reuse in Grid-Enabled Medical Applications, *Proc. 7th Int'l Symp. Biological and Medical Data Analysis (ISBMDA'06)*, pp.195–206 (2006).
5) Deelman, E. and et al.: Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems, *Scientific Programming*, Vol.13, No.3, pp.219–237 (2005).
6) Foster, I.: Globus Toolkit Version 4: Software for Service-Oriented Systems, *Proc. 2nd IFIP Int'l Conf. Network and Parallel Computing (NPC'05)*, pp.2–13 (2005).
7) Allcock, W.: GridFTP: Protocol Extensions to FTP for the Grid, *Global Grid Forum Recommendation GFD.20* (2003).
8) Boden, N.J. and et al.: Myrinet: A Gigabit-per-Second Local Area Network, *IEEE Micro*, Vol.15, No.1, pp.29–36 (1995).
9) MySQL: The world's most popular open source database. http://www.mysql.com/.
10) PC Cluster Consortium: SCore Cluster System. http://www.pccluster.org/.
11) Ino, F. and et al.: A Data Distributed Parallel Algorithm for Nonrigid Image Registration, *Parallel Computing*, Vol.31, No.1, pp.19–43 (2005).
12) Message Passing Interface Forum: MPI: A Message-Passing Interface Standard, *Int'l J. Supercomputer Applications and High Performance Computing*, Vol.8, No.3/4, pp.159–416 (1994).