社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2004－HPC－99 (42)
2004／8／1

グリッド上のスケーラブルな並列レプリケーションフレームワーク

滝澤 真一朗[†] 高宮 安仁[†]
中田 秀基[††,†] 松岡 聡[†,†††]

グリッド環境におけるデータレプリケーション手法として，レプリカ管理とデータ転送の集約およびマルチキャスト転送を利用し，複数サイト間の巨大データの効率的なレプリケートを可能とする，並列レプリケーションフレームワークを提案する．レプリカ情報を集中管理するレプリカ管理サービス，およびデータ要求元に応じて複数レプリカの中から最適なレプリカを選択し，同時並列ファイル転送ツールである Dolly+による高速ファイル転送を行うレプリケーションサービスからなるプロトタイプを実装した．遠隔データサイトに複数クラスタが接続された実際のデータグリッド環境においてプロトタイプの評価を行った結果，ノード数の増加によらずほぼ一定時間内にレプリケーションを完了し，クラスタ内でキャッシュを管理する単純な手法よりも優れたスケーラビリティを示した．

Scalable MultiReplication Framework on The Grid

Shin'ichiro Takizawa,[†] Yasuhito Takamiya,[†]
Hidemoto Nakada[††,†] and Satoshi Matsuoka[†,†††]

We propose a "MultiReplication Framework" for data replication in a data grid environment, where redundant and inefficient long-haul copying of replica data is avoided on local or near-local usage of same sets of data by employing aggregated and efficient multicast-based replication schemes. The replica manager manages the replica creation centrally using an XML-based schema, and when multiple peers requests a replica in a near-simultaneous fashion, this is detected and an organized multi-replication over multiple involved peers are initiated by the use of our multi-replicaiton tool Dolly+. Benchmarks on prototype implementation show that, our scheme scales well in a realistic data grid environment constituting of multiple clusters interconnected to a distant data archiver, maintaining constant replication time even if the number of nodes increase, being superior to simpler schemes such as maintaining a local cache of replicated data within a cluster.

1. はじめに

計算グリッド技術の発展により，物理や化学，バイオなどの分野において，実験や観測で生成されたデータをグリッド環境で解析する取り組みが進められている．ここで扱われるデータセットは数テラから数ペタバイトに及ぶ膨大なものであり，かつアクセスする研究者が地理的に広く分散している．そのため，グリッド環境で大規模データを効率的かつ安全に分散管理する必要があり，データアクセスの分散とデータ保護を目的に，データの複製 (レプリカ) を複数サイトに散在させて管理する手法が研究されている．たとえば，Globus プロジェクトのレプリカカタログ[1),2)] では，ネットワーク上に広く散在するレプリカの情報をデータベースで管理し，レプリカへアクセスするためのインターフェースを提供している．Grid Datafarm[3)] プロジェクトでは，レプリカ管理されたデータへ高速アクセス可能な大規模並列ファイルシステムの研究を行っている．

Grid Datafarm は，アクセスの集中するファイルには適宜アクセス先レプリカを変えることにより効率の良いアクセスを提供している．しかし，クライアントはレプリカ保持サイトと一対一通信を行うため，レプリカ数が少数の場合，アクセスの集中は防げない．一般に同一データの一対一転送に基づく複数サイトへの配布は，複数のコネクションで同一のデータが転送されることとなり，バンド幅の浪費につながる．また，生物の DNA シーケンシングを行う BLAST[4)] では，複数サイト間でのデータベースファイルの更新を，BLAST が実行される可能性のある全てのサイトで行う必要があり，しかもその更新は頻繁に行われる．こ

† 東京工業大学
Tokyo Institute of Technology
†† 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology
††† 国立情報学研究所
National Institute of Informatics

のような場合に，一対一でファイル転送を行っていては，アプリケーションの実行にファイル転送が追いつけず，実行の中止，一時停止が必要となる．BLASTに限らず，一対一でファイル転送を行うと同様の問題が生じるアプリケーションは多く存在する．

そこで本研究では，グリッド環境でのデータレプリケーション手法として，レプリカカタログによるデータの管理と，マルチキャスト転送に基づく高効率なレプリケーションを提供する，並列レプリケーションフレームワークを提案し，ポータビリティの高いXMLベースレプリカ管理と，クラスタ内高速ファイル転送ツールであるDolly+[5]を用いたレプリケーションから成るシステムを試作，その有用性を検証した．試作実装では，レプリケーション手法として，ローカルネットワーク内の複数ノードからのデータ要求を集約し，データ要求数を制限することによりデータサイトへのアクセスを減らす手法を用いた．レプリカ位置情報検索の性能評価と，本レプリケーション手法と単純な一対一転送に基づくレプリケーション手法の比較評価を行った結果，レプリケート先ノード数に対してスケーラブルなレプリケーション手法の実現は確認したものの，キャッシュミス時のレプリカ検索性能低下を改善する必要があることもわかった．

2. クラスタ内ファイル転送ツール: Dolly+

本研究ではローカルネットワーク内でのスケーラブルなデータ転送のため，クラスタ内高速ファイル転送ツールである，Dolly+を[5]☆を使用した．Dolly+での転送方式を以下に示す．

図1はPCクラスタを模式化したものであり，中央の円で表されるネットワークスイッチに複数のクラスタノードが実線で接続されている．Dolly+は，各ノードが仮想的なリング型ネットワーク上に存在するとみなして，そのリング上でTCP/IPを用いたファイル転送を行う．転送元ノードはServer，転送先複数ノードはClientと区別される．転送時には，Serverはファイルを一定サイズの断片に分割し，図のリング型転送路にしたがって連続して送信する．一方，Clientは断片を受信しつつ，以前受信した断片を次のClientへ送信するパイプライン方式でデータのリレーを行う．

Dolly+による転送にかかる時間は，転送するデータサイズをX(MB)，ノード数をN(台)，ノード間のバンド幅をL(Mbps)，遅延をδ(s)とすると，以下の式で与えられる．

$$\frac{X}{L} + N \times \delta$$

ここでδに対してXが十分大きい場合，$N \times \delta$は$\frac{X}{L}$に対して無視できるほど小さくなる．これにより，ファイルサイズが大きいときはノード数に対してほぼ一定

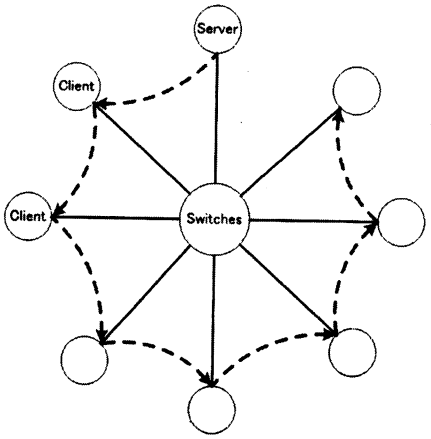

図1 Dolly+仮想リング型ネットワーク図

の時間でのファイル転送が可能と言える．[6]

3. 並列レプリケーションの要件

本研究では，並列レプリケーションの要件として以下の3つを挙げる．

レプリカの位置情報の管理 レプリカ作成にはソースとなるレプリカの位置情報が予め分かっている必要がある．また，作成されたレプリカも後のソースレプリカとして扱うため，レプリカ位置情報の管理が必要である．

マルチキャストファイル転送 一対一のファイル転送の場合，複数サイトが単一レプリカにアクセスすると，ネットワークの競合により全体の転送パフォーマンスが低下する．これを防ぐため，マルチキャストファイル転送が必要である．

ネットワーク監視機能 複数サイトに散在するレプリカの中からソースレプリカとして最適なものを選び出すためには，要求サイト－レプリカ保持サイト間ネットワークの定期的な性能測定が必要である．

4. 並列レプリケーションの設計

並列レプリケーションフレームワークを，大きく「レプリカ管理サービス」,「ファイル転送サービス」の2つのコンポーネントに分割して設計した．

レプリカ管理サービス Replica Location Server(以下RLS)とそのクライアントからなる．RLSではファイルを論理的な名前で管理し，その属性情報と，その物理実体(レプリカ)の位置情報をレプリカカタログというデータベースに記録する．また，ファイルシステムにおけるディレクトリと同様に，複数の論理ファイルの集合を表す論理コ

☆ 高エネルギー加速器研究機構の真鍋氏と松岡研究室の共同開発

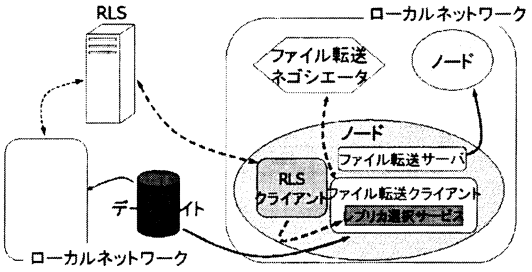

図 2 プロトタイプシステム概要図

レクションをサポートする．RLS のクライアントは RLS に対して論理ファイル・コレクションの登録・削除・属性情報の追加，レプリカの位置情報の検索などの操作を依頼する．

ファイル転送サービス ファイルを要求するノードへ，そのレプリカの中から転送に最適なものを 1 つ選び出しレプリケートを行う．本システムに関与するノードはすべてレプリケート元であるファイルサーバであり，レプリケート先であるファイルクライアントでもある．

各ノードは他ノードからのレプリカ要求を受け取ると，即座には転送を開始せず，その他のノードからの同じレプリカ要求の到着を一定時間待つ．その後，対象のレプリカを要求する全ノードに対してマルチキャスト転送する．これにより，ノード数に対してスケーラブルなレプリケーションを実現できる．レプリカを要求するノードが 1 台のみの場合はこの待ち時間が無駄になるが，ノードが複数存在する際にはその待ち時間によるコストより，マルチキャスト転送による恩恵の方が大きいと考える．

各ノードはレプリカ取得後，後のレプリケーションにおいて転送元として利用するため，レプリカカタログにレプリカ情報を登録する．

また，1 つのファイルの複数のレプリカの中から転送に最適なものを選び出す際には，レプリケート先とレプリケート元間のネットワークの状況を判断して選択する．

以上より構成される並列レプリケーションは以下の流れで動作する．

(1) クライアントは RLS に論理ファイルのレプリカの位置情報を問い合わせる．
(2) 問い合わせの結果，複数のレプリカが存在した場合，その中から転送に最も適したものを選び出す．
(3) 転送サービスを用いてレプリケートを行う．

5. プロトタイプシステムの実装

上記の設計方針に基づいた並列レプリケーションフレームワークを Java 言語を用いて試作した．図 2 にそのシステムの概要を示す．以下に各部分の実装について説明する．

```
<?xml version="1.0" encoding="UTF-8"?>
<query code="12">
    <data name="exp1" collection="/aaa/" xsi:type="file"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    </data>
</query>
```

図 3 クエリ文字列

```
<?xml version="1.0" encoding="UTF-8"?>
<reply code="1">
    <data name="exp1" collection="/aaa/" xsi:type="file"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <locations>
            <location id="pad000.titech.hpcc.jp">
                <info site="pad000.titech.hpcc.jp" path="/tmp/aaa.txt"/>
            </location>
        </locations>
    </data>
</reply>
```

図 4 リプライ文字列

5.1 レプリカ管理サービス

RLS と RLS クライアントからなり，通信プロトコルには，ポータビリティのため，XML ベースのプロトコルを採用した．サーバに対してクライアントが論理ファイルの位置情報を問い合わせる際のクエリとそれに対するサーバのリプライはそれぞれ図 3，図 4 のように表される．

5.2 レプリカカタログ

レプリカカタログでは論理ファイル，論理コレクション，レプリカを保持するノードの情報の 3 種類の情報を管理する．本実装では，レプリカカタログにはネイティブ XML データベースである Xindice[7] を採用した．論理ファイルの情報は論理コレクションの属性情報とし，論理コレクションとノード情報をデータベースのエントリとした．

実装上の工夫として，並列レプリケーションの性質上複数のノードからの同じエントリへのアクセスが頻繁に生じることと，Xindice へのアクセスコストが大きいことが分かっているため，アクセスされたエントリのキャッシュなどを行った．

5.3 ファイル転送サービス

ファイル転送サービスは，各ノード上で動作する転送サーバ・クライアントと，ローカルネットワーク内で唯一のネゴシエータから構成される．転送サーバはローカルネットワーク内のノードに Dolly+でファイルを転送するファイルサーバである．転送クライアントはファイルのダウンロードを，転送元ノードの位置に応じて，以下のように切り替える．

まず，同じネットワーク内のノードからのダウンロードは，そのノード上の転送サーバに転送要求を送り，Dolly+で転送してもらう．一方，ネットワーク外からのダウンロードではネゴシエータに転送要求を送る．

ネゴシエータはネットワーク内の複数のノードが同時にネットワーク外の同一レプリカを要求した際に，そのレプリカを取得するノードを 1 台に限定するサービスである．選ばれた 1 台のノードはそのレプリカを HTTP で取得する．その後，ネットワーク内の同じレプリカを要求していた他ノードに対して，転送サーバにより，ダウンロードしたレプリカを Dolly+を用いてマルチキャスト転送する．このように，ローカルネットワーク外のレプリカを取得する際は 2 段式に転送を行うことになる．

ローカルネットワーク内のレプリカ要求ノードすべてが直接外部のレプリカを取得する場合と比べると，ノード数が少ないときの効果はあまり見込めないが，ノード数が多いときにはそのレプリカへのアクセスを制限することにより，高速な転送が期待できる．

5.4 最適レプリカ決定手法

最適レプリカの決定には，レプリカ要求サイト－レプリカ保持サイト間 RTT が最小であるレプリカ保持サイト上のレプリカを最適とする方法をとった．本来なら，スループットでの比較が望ましいが，実装上の都合より，スループットに高い相関を持つレイテンシを尺度として選んだ．

6. 実験と考察

実装したプロトタイプシステムの性能をレプリカカタログ検索能力，レプリケーション能力の 2 点について評価した．

6.1 レプリカカタログ検索性能

レプリカカタログに登録されているファイル数を 10 から 50 へ 10 単位で変化させ，同時にファイルあたりのレプリカ数も 10 から 50 へ 10 単位で変化させた際の，1 つのファイルのレプリカの全位置情報の検索を 100 回行い，その平均時間を測定した．対象のファイルの位置情報がキャッシュヒットした場合と，キャッシュミスし，データベースから読み込まれた場合の 2 パターンの実験を行った．実験には図 1 に示す構成の Presto III クラスタ 2 ノードを使用した．結果を図 5 と図 6 に示す．図より，キャッシュからの検索の方が，データ量の一番少ない 10 ファイル 10 レプリカでも約 4 倍，一番多い 50 ファイル 50 レプリカで約 16 倍高速であることがわかり，キャッシュ使用の効果が確認できる．

図 5 を見ると，キャッシュから読み込まれた際の検索性能は，ファイル数にはよらず，ファイルあたりのレプリカ数によるものとわかる．これは，キャッシュからデータを取得する際には，どのデータに対しても同じコストで取得できることと，クライアントへ検索結果を返す際，図 4 のような XML 文字列を返すため，レプリカ数が増えると返すデータ量が増えてしまうことによる．一方，図 6 で，データベースから読み込まれた場合はファイル数によっても性能差が見られる．これは，データベースにはコレクション単位でファイル情報を記録するので，ファイル数が増えると取得する情報量も増えるためである．

表 1 東工大 Presto III クラスタ

CPU	Athlon MP 1900+
メモリ	768MB
ネットワーク	100Base-T
OS	Linux Kernel 2.4.19
Java	Sun JDK 1.4.2_03

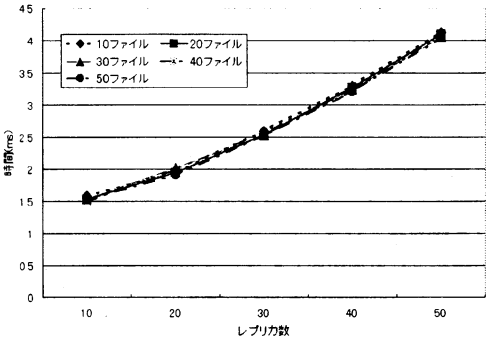

図 5 キャッシュヒット時のレプリカカタログ検索性能

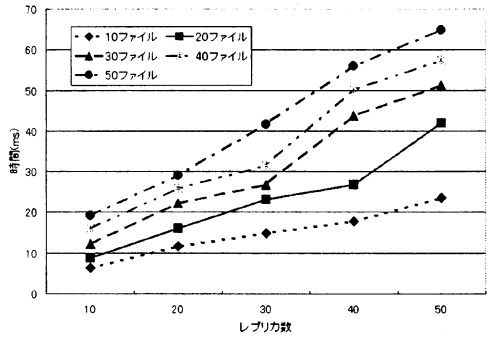

図 6 キャッシュミス時のレプリカカタログ検索性能

6.2 レプリケーション性能

データサイト上に存在する大容量ファイルに対して，ローカルネットワーク内の複数ノードへ同時にレプリケートを行う設定で実験を行った．RLS として表 2 に示す PC を用いた．ローカルネットワーク内ノードとしては表 1 に示す構成の PC クラスタを用いた．レプリケートするデータは Fedora Linux のインストーラ CD イメージファイル (約 660MB) とし，リモートデータサイトとしては Fedora のオーストラリアミラーサーバ (mirror.pacific.net.au) と東工大松岡研 Web サーバ (matsu-www.is.titech.ac.jp) を RLS に登録した．レプリケーションを行う際のネットワー

表 2 RLS 実行 PC

CPU	Mobile PentiumIII 1.13GHz
メモリ	640MB
ネットワーク	100Base-T
OS	Windows XP Professional
Java	Sun JDK 1.4.2_01

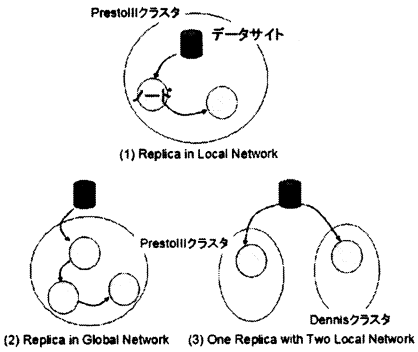

図 7 ネットワーク構成

表 3 筑波大 Dennis クラスタ

CPU	Dual Xeon 2.4GHz
メモリ	1GB
ネットワーク	1000Base-T
OS	Linux Kernel 2.4.20-28.8smp
Java	Sun JDK 1.4.2_03

ク構成として図 7 に示す 3 パターンについて実験を行った.

(1) は対象のレプリカをすでにローカルネットワーク内ノードが保持している場合の構成である. この場合のレプリケーションは, Dolly+による転送 1 回のみ行われる. (2) は対象のレプリカがグローバルデータサイトに存在する場合の構成である. この場合は 1 台の代表ノードが対象のデータサイトから http でレプリカを取得後, 他のノードへ Dolly+で配布する. (3) は (2) と同じ設定だが, 複数ローカルネットワーク内の複数ノードがレプリケーション対象の場合の構成である. ここで, 2 つ目のローカルネットワークとして, 表 3 に示す構成の筑波大学の Dennis クラスタを使用した. このときのレプリケーションは (2) と同様に行われる. 以下に構成 (1)〜(3) での実験結果を示す.

6.2.1 (1): ローカルサイト内レプリカのレプリケーション

ローカルネットワーク内にすでに存在するファイルのレプリケーション性能評価のため, レプリケーション時間について, 本プロトタイプと, 単純に rcp で逐次・並列に転送を行った場合とで比較を行った. 結果を図 8 に示す. 本システムではノード数を変化させても, レプリケーション時間はほぼ一定であり, 効率の

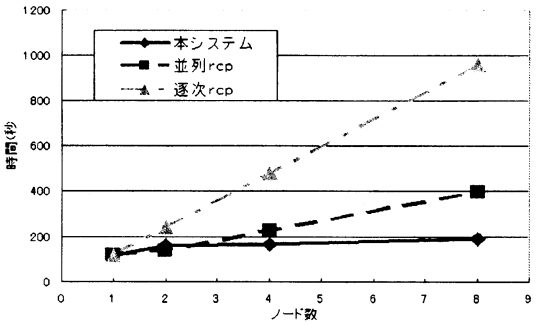

図 8 ローカルサイト内レプリカのレプリケーション

良いスケーラブルなレプリケーション手法を実現していると言える. ここで, 並列 rcp は逐次 rcp と比較するとはるかに高速だが, 本システムと比較すると低速である. これは, レプリカを提供するノードにアクセスが集中することによるネットワークの競合と, ディスクアクセスの集中によると思われる.

6.2.2 (2): リモートサイト上レプリカのレプリケーション

リモートデータサイトから単一ローカルネットワーク内複数ノードへのレプリケーション性能の評価のため, レプリケーション時間について, 単純に逐次・並列 http で転送を行った場合との比較を行った. また, この際, 本プロトタイプシステムの選択に合わせ, http で転送を行う場合には松岡研 Web サーバを使用するよう設定した. 結果は図 9 である. 本システムではノード台数が 2 台以上の場合, 台数を変化させてもレプリケーションにかかる時間はほぼ一定で, この環境でのスケーラビリティが確認できる. だが, 台数が 2, 4 台の時には本システムは並列 http より低速である. これは, RLS へのレプリカ位置情報の検索, 最適サイト決定によるコストと, Dolly+の実行を定期的に行うことによる, 転送が開始されるまでの待ち時間の影響のためと思われる.

6.2.3 (3): 複数ローカルネットワークにおけるレプリケーション

データサイトから複数ネットワークへのレプリケーション性能の評価のため, レプリケーション時間について, 逐次・並列 http での転送と比較を行った. ここでも, 本プロトタイプシステムの選択にあわせ, http での転送に関しては, どちらのクラスタノードも松岡研 Web サーバを使用するよう設定した. 結果を図 10 に示す. この環境においても, 本システムのスケーラビリティが確認できる.

ノード数が 2 台の場合, 並列 http での転送結果より劣っているのは, 先と同じ理由である. また, 本システムの結果について, ノード数が 2 台以上の場合, PrestoIII クラスタの結果と Dennis クラスタの結果と

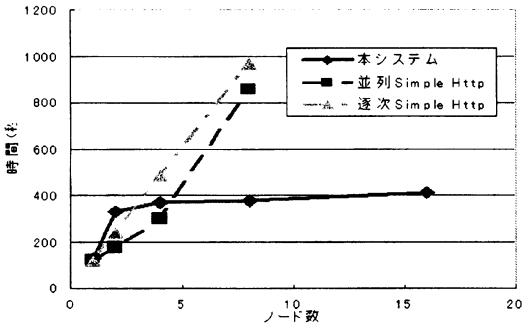

図 9 ローカルサイト内レプリカのレプリケーション

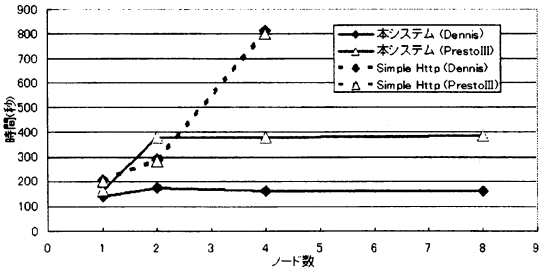

図 10 複数ローカルネットワークにおけるリプリケーション

では大きな性能差が現れているが，これは前者において各ノード間は 100Mbps のネットワークであるのに対して，後者では 1000Mbps なので，クラスタ内での Dolly+転送が高速に行われたことによる．一方，逐次に http で転送を行った結果では 2 者に違いが現れていない．これは，各クラスタからリモートデータサイト (松岡研 Web サーバ) へのアクセスにボトルネックがあるためである．

7. まとめと今後の課題

並列レプリケーションフレームワークの設計と試作，評価を行った．試作システムにおいて，レプリカカタログでエントリをキャッシュすることにより高速なレプリカ位置情報の検索が可能であることを確認した．しかし，ファイルあたりのレプリカ数が増えるとともに通信量が増え，性能が低下することも確認した．また，ローカルネットワーク内におけるレプリケーション手法として Dolly+を用いたマルチキャスト転送を採用することにより，特にノード数が多いほど，従来の一対一レプリケーション手法からの性能向上が確認できた．複数クラスタを用いた実験においてもノード数に対してスケーラブルなレプリケーションが可能であることがわかり，実際のデータグリッド環境においても有用であることが確認できた．

今後の課題としては，以下の項目が挙げられる．

- ユーザやグループのアクセス権限をレプリカカタログで管理する．
- クライアントからの RLS へのアクセスを分散・減少させる．これには，今回レプリケーションに採用したクエリの集約手法と同様な手法を用いる．
- レプリカカタログの再設計．検索実験結果より，キャッシュミスした場合のペナルティは実験時の最大データ量で約 16 倍であった．実際にはレプリカ数はもっと増えるはずで，それに伴い検索時間の増加が見込める．レプリカカタログの再設計，あるいは Xindice 以外のデータベースの採用の検討が必要となる．
- グローバルネットワーク上でのマルチキャストベースのレプリケーション手法の導入．今回の実装では，ローカルネットワーク内ノードのレプリカ要求を集約することにより，データサイトーローカルネットワーク間レプリカ転送量を減らすことに成功した．しかし，管理するネットワーク数の増加に伴い，データサイトへのアクセスは増加する．このアクセス削減のため，マルチキャストベースレプリケーションが要求される．

謝辞 評価実験を行うにあたり，クラスタ資源を提供して頂いた筑波大 HPCS 研究室の方々に深く感謝の意を表します．本研究の一部は平成 15 年度文部科学省科研費特定領域研究 (13224034: Grid における peer-to-peer 大規模データ処理に関する研究) による．

参 考 文 献

1) A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets. Journal of Network and Computer Applications, 23:187-200, 2001
2) Globus project. Getting Started with the Globus Replica Catalog
3) Grid Datafarm: http://datafarm.apgrid.org/
4) BLAST: http://www.ncbi.nlm.nih.gov/BLAST/
5) Dolly+: http://corvus.kek.jp/ manabe/pcf/dolly/
6) Satoshi Matsuoka. You Don't Really Need Big Fat Switches Anymore - Almost. 情報処理学会研究報告, 2003-ARC-154, SWoPP 2003, pp. 157-162, 2003.
7) Apache Xindice: http://xml.apache.org/xindice/