2007－HPC－111 (19)
2007／8／2

広域分散環境における大規模データ管理のためのノードグルーピング

佐藤 仁 † 松岡 聡†,†† 遠藤 敏夫 †

グリッドファイルシステムでは，1) クライアントからある特定のノードやファイルへのアクセスに時間的局所性がありアクセスが集中する，2) ファイルシステム上のファイルへのアクセスが空間的に遠方に存在するノードへのアクセスとなる，などの要因によりファイルアクセス性能が低下することが問題となる．しかし，効率の良い大規模データ管理を実現するためにどのようにファイルをグリッドファイルシステムを構成する広域に分散したノードに配置すれば良いかの戦略を決定づけるメトリックは明らかではない．我々は，広域に分散した 5 サイトからなる HPC クラスタを連携してファイルシステムを構成し，その上で，ファイルアクセスの行った際の性能を調査した．その結果，リモートファイルアクセス性能はノード間のバンド幅の影響を受けるが，RTT，バンド幅などのネットワークの限定された情報だけではファイルシステムを構成するノードの関係の推定が困難であること，ローカルファイルアクセス性能でもファイルアクセスパターンにより最大 0.1 倍の性能に抑えられてしまうこと，が明らかになった．

Node Grouping for Large-Scale Data Management on the Grid

Hitoshi Sato ,† Satoshi Matsuoka †,†† and Toshio Endo †

In parallel computing environments such as HPC clusters and the Grid, data-intensive applications involve large overhead due to the access concentration on files on commonly shared nodes. A grid filesystem with an automatic data management mechanism is one of the solutions to avoid such performance decrease. However, metrics to achieve efficient large scale data management are not clear for a given real grid environment. We federated 5 geographically distributed HPC clusters using a grid filesystem and experimented its various performance metrics of file access on the filesystem. We observed that, although remote access performance of files is affected by inter-node bandwidth, other factors are in place which makes prediction of performance solely based on limited inter-node information such as RTT or network bandwidth difficult, and that even for local file access, performance difference could be an order of magnitude depending on file access patterns due to access contentions.

1. はじめに

近年，大規模なデータ処理を必要とするデータインテンシブアプリケーションの実行環境としてグリッドの利用が実用的になりつつある．グリッド上でデータ共有を行うには，シングルシステムイメージを提供するためのファイルシステムをベースにするのが望ましい．これは，既存のアプリケーションを修正することなしに利用でき複数のアプリケーション間の連携がファイルベースで行えること，多くのアプリケーション利用者や開発者が慣れ親しんでいること，などの利点があるためである．しかし，このようなファイルシステムをグリッド上で実際に運用しようとするとうまくいかない場合がある．これは，グリッドが大規模で不均質であるため，1) クライアントからある特定のノードやファイルへのアクセスに時間的局所性がありアクセスが集中する，2) ファイルシステム上のファイルへのアクセスが空間的に遠方に存在するノードへのアクセスとなる，などの要因によりファイルアクセス性能が低下するためである．

我々は，今まで，上記の問題を解決するために，ファイルシステム側でアクセスの挙動を解析し必要に応じてファイル複製を積極的に作成することでファイルへのアクセスを分散するグリッドファイルシステム上での自動的なデータ管理機構を提案してきた[1]．しかしながら，効率の良い大規模データ管理を実現するためにどのようにファイルをグリッドファイルシステムを構成する広域に分散したノードに配置すれば良いかの戦略を決定づけるメトリックは明らかではない．本稿では，広域に分散した 5 サイトからなる HPC クラスタを連携してファイルシステムを構成し，その上で，リモートファイルアクセスの行った際の性能を調査した結果について報告する．

† 東京工業大学
Tokyo Institute of Technology
†† 国立情報学研究所
National Institute of Informatics

2. グリッドファイルシステム上でのデータ管理

2.1 ファイルアクセスの自動分散を行うグリッド用分散ファイルシステム

我々が対象とするグリッドファイルシステムの構成を図 1 に記す．Grid Datafarm Architecture[2] の参照実装である Gfarm[3] の上に自動的なデータ管理機構を拡張して実現している．

Gfarm は，ファイルシステムへのアクセスを提供するクライアント，ファイルシステムのメタデータを扱うメタデータサーバ，また，実際にファイル断片を保持する I/O ノードの 3 つの主な構成要素からなる．ユーザや既存のアプリケーションは，クライアントからシングルシステムイメージでファイルへ透過的にアクセスできるが，実際のファイル及びファイル断片はクライアントとは異なるノードのストレージ上に分散されて格納される．ファイルシステム上のファイルへアクセスするために，クライアントは，メタデータからファイルの所在に関する情報を取得し，実際にファイル断片が存在する I/O ノードの中から RTT 及び負荷などの情報を元に，I/O ノードを決定し，実際にファイルへのアクセスを行う．

これらのファイルシステムの構成要素に加え，ファイルシステム側でアクセスの挙動の解析や必要に応じたファイル複製の作成などの自動的なデータ管理を行うためにファイルシステムモニタと I/O ノードモニタが動作する．ファイルシステムモニタはメタデータサーバにおいてクライアントから I/O ノード上へのファイルアクセスのモニタリングを行い，I/O ノードモニタは I/O ノードにおいてファイルシステムモニタの情報やそのノードの CPU の負荷やストレージの利用状況などのリソースに関する情報を元に実際にファイル複製作成の指令を行う．ファイルシステムとこれらのコンポーネントが協調動作することにより，グリッド上でのファイルアクセスの性能向上を実現する．

2.2 広域分散環境でのファイルアクセスの問題点

多くのデータインテンシブアプリケーションでは，write once，read mostly なワークロードを持つため，ファイルへの read アクセスの高速化が重要となる．広域分散環境でのこのようなファイルアクセスの高速化においても核となるアイデアは，「いかにアクセスが必要なデータをアクセス要求元の近くに置くか」，という点である．既存の提案では，ファイルアクセス要求のあるノードから他のノードのストレージ上へのリモートファイルアクセスを避けるために，1) 参照の多いファイルをアクセス要求元の近くのストレージに複製する手法[4)5)] や 2) 計算ジョブをアクセスしたいファイルが存在する計算資源へ投入する手法[2)4)] など，ネットワークへのアクセスを抑えるための戦略が採られる．しかしながら，データインテンシブアプリケーションを対象にした場合，このような手法を単純に適応しただけでは，以下のような問題がある．

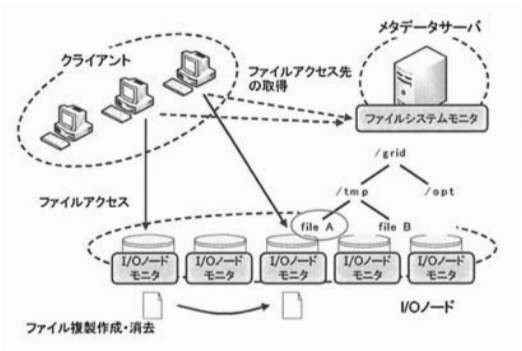

図 1 対象とするグリッドファイルシステムの構成

- ローカルストレージへのキャッシュ作成の問題
  - 大規模なデータを対象にするため，ファイルアクセスの際に同時にキャッシュを行うのに時間がかかる．
  - ローカルストレージの容量に限度があるため，アクセスのあった全てのファイルをキャッシュすることができない．
- ノード間でのファイル複製作成の問題
  - ファイルアクセスの際，適切なファイル複製の選択を行わないと，遠方へのノードのストレージへのリモートファイルアクセスになるケースが発生し，性能が低下する．
- ファイルが存在するノードへのジョブ投入の問題
  - アクセス対象とするファイルが複数存在し，それらが異なるノードのローカルストレージに存在するとき，ファイルのリモートアクセスが発生する．
  - 複数のジョブが同じファイルへアクセスする際，そのファイルの存在するノードにジョブ投入が集中する．
  - 複数のジョブの各々必要なデータが，同じノードのストレージ上に存在する場合，そのノードへのジョブ投入が集中する．

これらの問題点を解決するためには，ファイルアクセスの性能低下を抑えるために，グリッドファイルシステムを構成するノードの性能や接続されたノード同士のネットワークトポロジを考慮したファイル配置を行うことが必要である．近年，ネットワークトポロジの推定[6] やネットワーク上のノードのグルーピング手法[7] が数多く提案されているが，実際の広域分散環境でのグリッドファイルシステムの運用を想定した場合，具体的にどのようなメトリックがファイルアクセス性

能に影響を与えるかは明らかではない.

3. 実　　験

2.2 節で述べたグリッドファイルシステム上での効率の良いデータ配置を実現するための具体的なメトリックを明らかにするために，東京 (hongo, okubo)，神奈川 (suzuk)，千葉 (chiba)，京都 (imade) に存在する計 5 サイトからなる HPC クラスタをグリッドファイルシステムで連携したテストベッドを構築し，ファイルアクセスの行った際の性能を調査した．テストベッドを構成する各 HPC クラスタの性能を表 1 に示す．構築されたグリッドファイルシステムにおいて，メタデータサーバを suzuk クラスタの 1 ノードに割り当て，他のクラスタノードをクライアントと I/O ノードとして割り当てた．この際，構築したファイルシステムの容量は最大で約 85TB(ただし，マシン故障によるノード数の変化やグリッドファイルシステムのスプールとして利用している HDD の利用状況などにより変動する) となった．実験として，空間的に遠方のノードへのファイルアクセスの影響とファイルアクセスの時間的な集中の影響を明らかにするために，

- 1 台のクライアントから他の 1 台の I/O ノードのファイルへ read を行うリモートファイルアクセス
- 複数台のクライアントが一斉に I/O ノード上のファイルへ read を行うファイルアクセス

の 2 種類のシナリオを設定した．ただし，実験で使用した環境は他のユーザとアクティブに共有された環境であった．

3.1 ノード間のリモートファイルアクセスの性能

まず，クラスタノード間のリモートファイルアクセスの性能を調べるために，hongo クラスタの 1 ノードから他のクラスタノードのストレージに存在する 1 つのファイルに対して内容を全て読み取るような read アクセスを 1 回行った際のファイルアクセスの I/O バンド幅を測定した．read アクセスの対象とするファイルは 128MB とした．以降，同一サイト内のクラスタノード間のリモートファイルアクセスに関する実験とサイトをまたがるクラスタノード間のリモートファイルアクセスに関する実験ついて述べる．

3.1.1 同一サイトのクラスタノード間のリモートファイルアクセス

hongo クラスタは表 1 で示したように異なる構成のマシンを含み (hongo(pm),hongo(c2d))，かつ，マシン構成によりノード間のバンド幅が大幅に異なる (表 2)．このため，これらの影響を調べるために，以下の 4 つのシナリオを設定し実験を行った．

(1) hongo(c2d) ノード間のアクセス
(2) hongo(pm) ノード間のアクセス
(3) hongo(c2d) ノードから hongo(pm) ノードへのアクセス
(4) hongo(pm) ノードから hongo(c2d) ノードへのアクセス

この際の各ノード間の RTT は 0.162[ms] で，バンド幅は表 2 のとおりであった．

表 2 hongo クラスタのノード間のネットワークバンド幅 [MB/sec]

src/dst	hongo(c2d)	hongo(pm)
hongo(c2d)	117.3	41.30
hongo(pm)	66.75	65.38

表 3 同一サイト内のノード間のリモートファイルアクセスの I/O バンド幅 [MB/sec]

src/dst	hongo(c2d)	hongo(pm)
hongo(c2d)	54.3	29.0
hongo(pm)	7.55	32.1

表 4 同一サイト内のノード間のリモートファイルアクセスの I/O バンド幅の相対性能

src/dst	hongo(c2d)	hongo(pm)
hongo(c2d)	1.0	0.53
hongo(pm)	0.14	0.59

各シナリオにおけるリモートファイルアクセスの際の I/O バンド幅の結果を表 3 に，また，そのときの hongo(c2d) ノード間の結果を基準とした相対性能を表 4 に示す．結果より，同一サイトのクラスタノード間のファイルアクセスであっても，ノードのマシン構成の違いにより著しく性能が異なることを確認した．例えば，hongo(c2d) ノード間では 54.3[MB/sec] の性能を示すのに対し，hongo(pm) ノード間では 32.1[MB/sec] となり，ノード間のネットワークバンド幅との大きな相関がみられた．一方で，hongo(pm) ノードと hongo(c2d) ノード間のファイルアクセスでは，ファイルアクセスを行う方向によって全く異なる性能を示した．特に，hongo(pm) ノードから hongo(c2d) ノードへのファイルアクセスでは著しく性能が低下し，7.55[MB/sec] と最も性能が良い場合の hongo(c2c) ノード間の結果と比較して 0.14 倍の性能に抑えられていた．この原因は不明で，現在調査中である．

3.1.2 サイトをまたがるクラスタノード間のリモートファイルアクセス

サイトをまたがるクラスタノード間のリモートファイルアクセス性能の違いを調べるために，構築されたテストベット上のクラスタの属するサイト間のネットワークの性能に応じて，以下の 4 つの条件を設定し実験を行った．

(1) 低遅延，高バンド幅な WAN で接続された異なるサイトのクラスタノード間のアクセス (hongo-suzuk)
(2) 高遅延，高バンド幅な WAN で接続された異なるサイトのクラスタノード間のアクセス

表 1　各サイトの HPC クラスタの性能

	hongo		chiba		okubo	suzuk	imade
	hongo(pm)	hongo(c2d)	chiba(pm)	chiba(c2d)			
#nodes	84		128				
	70	14	70	58	14	36	30
#cores	1	2	1	2	2		
CPU	Pentium M 1.8 GHz	Core2Duo 2.33 GHz	Pentium M 1.8 GHz	Core2Duo 2.33 GHz	Core2Duo 2.33GHz		
Memory	1GB	4GB	1GB	4GB	4GB		
OS	Linux 2.6.18						
Network	GBEther						

表 5　hongo クラスタと各クラスタ間のネットワーク性能

src/dst	suzuk	chiba	okubo	imade
bandwidth [MB/sec]	104.5	97.63	10.74	10.53
RTT[ms]	1.66	6.03	2.59	12.5

(hongo-chiba)

(3)　低遅延，低バンド幅な WAN で接続された異なるサイトのクラスタノード間のアクセス (hongo-okubo)

(4)　高遅延，低バンド幅な WAN で接続された異なるサイトのクラスタノード間のアクセス (hongo-imade)

この実験では，hongo クラスタでは hongo(c2d) ノード，chiba クラスタでは chiba(c2d) ノードを用いた．hongo クラスタと各クラスタ間のネットワーク性能を表 5 に示す．各ネットワークではパケットロスがほとんどなく，各クラスタノードの TCP Window Size は 85.3[KB] であった．

各シナリオにおけるリモートファイルアクセスの I/O バンド幅と実験 (1) の hongo-suzuk 間の結果を基準にした相対性能を表 6 に示す．比較として，3.1.1 節の同一サイト内のクラスタノード間のリモートファイルアクセスの実験の最良の結果 (3.1.1 節の実験 (1), LAN best) と最悪の結果 (3.1.1 節の実験 (4), LAN worst) を示す．実験 (1) の低遅延，高バンド幅な WAN で接続されたクラスタノード間の実験で 15.2[MB/sec] の性能を示し，最も良い結果を得た．これは，3.1.1 節の同一サイト内のクラスタノード間のリモートファイルアクセスの実験の最悪の結果 (LAN worst) よりも 2.0 倍高い性能であるが，最良の結果 (LAN best) よりも 0.28 倍低い性能を表している．一方，実験 (4) の高遅延，低バンド幅な WAN で接続されたノード間の実験では，1.20[MB/sec] と大幅な性能低下を示した．また，実験 (1) と (2) では，接続されているクラスタ間のネットワークのバンド幅は同等であったが RTT の違いにより実験 (2) が実験 (1) と比較して 0.32 倍の性能に抑えられていた．

3.2　アクセス集中が発生した状況でのファイルアクセス

1 つのノードへのアクセス集中が発生した状況でのファイルアクセス性能を調べるために，hongo クラスタの 1 ノード (hongo(c2d) ノード) からノードのストレージに存在する 1 つのファイルに対して内容を全て読み取るような read アクセスを行うジョブをスケジューラ (torque) 経由で大量に投入した際の各ジョブのファイルアクセスの際の I/O バンド幅を測定した．実験では，ファイルサイズを 128MB と設定し，投入回数を 64 回とした．以降，ファイルの存在するストレージがジョブ投入ノードのローカルストレージである場合のローカルファイルアクセスの実験と他ノードのストレージである場合のリモートファイルアクセスの実験について述べる．

3.2.1　ローカルファイルアクセス

ローカルファイルアクセスが集中して発生した状況でのファイルアクセスの性能を調べるために，ローカルストレージにサイズは同じだが異なるファイルを複数配置し，ジョブ投入時にアクセス対象となるファイルをランダムに決定することで，I/O ノードに負荷を与えた．各ジョブはファイルの存在するノードに対して集中的に投入した．ノードの HDD の I/O バンド幅は Sequential Read の場合で 3.54[MB/sec] であり，メモリ I/O バンド幅は 3.33[GB/sec] であった．

ローカルファイルアクセス集中時の各ジョブの I/O バンド幅の分布を図 2 に，また，そのときの I/O バンド幅の最大値，最小値，平均値，及び，ファイル数が 1 のときの実験の平均値を基準にした相対性能を表 7 に示す．アクセス対象のファイルの数が増加するにつれ，性能が低下し，最大で 0.1 倍に抑えられているのを確認した．これは，少数のファイルに対するアクセスの場合，アクセス対象となっているファイルがメモリ上のページキャッシュに存在するため，高い性能を示すのに対し，多数の異なるファイルが同時にアクセスされる場合，アクセス対象となっているファイルがメモリ上のページキャッシュに存在しない状況が頻発し，ローカルディスクへのアクセスが発生し，競合状態が起こるためである．

3.2.2　リモートファイルアクセス

リモートファイルアクセスが集中して発生した状況でのファイルアクセスの性能を調べるために，3.1 節と同様に，アクセスの対象となるノードを各クラスタ

表 6 サイトをまたがるノード間のリモートファイルアクセスの I/O バンド幅 [MB/sec] と相対性能

src/dst	suzuk	chiba	okubo	imade	LAN best	LAN worst
bandwidth [MB/sec]	15.2	4.92	4.78	1.20	54.3	7.55
relative perf.	1.0	0.32	0.31	0.079	3.5	0.49

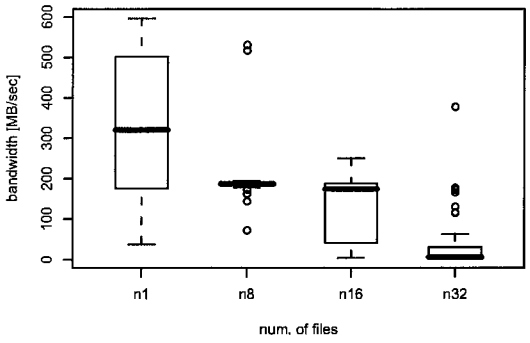

図 2 ローカルファイルアクセス集中時の各ジョブの I/O バンド幅の分布

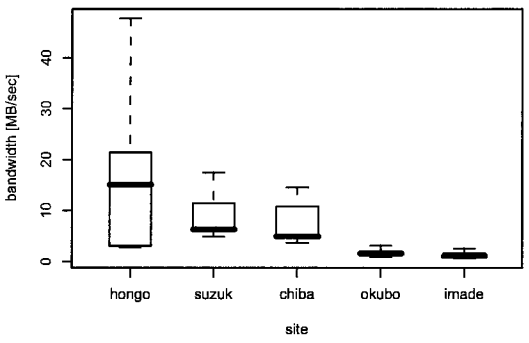

図 3 リモートファイルアクセス集中時の I/O バンド幅の分布

表 7 ローカルファイルアクセス集中時の各ジョブの I/O バンド幅の最大値，最小値，平均値，及び，相対性能

#files	1	8	16	32
max	596.0	531.0	250.0	377.7
min	37.00	72.07	3.992	19.75
avg	322.6	194.9	124.3	33.55
relative perf.	1.0	0.60	0.39	0.10

表 8 リモートファイルアクセス集中時の I/O バンド幅の最大値，最小値，平均値，及び，相対性能

site	hongo	suzuk	chiba	okubo	imade
max	47.7	17.4	14.6	3.13	3.34
min	2.79	4.90	3.66	0.873	0.689
avg	15.6	8.50	7.46	1.71	1.33
relative perf.	1.0	0.29	0.26	0.059	0.046

のノード (hongo, suzuk, chiba, okubo, imade) とし，各ノードのストレージ上にファイルを 1 つ用意し，実験を行った．ただし，hongo クラスタでは hongo(c2d) ノードを使用した．

リモートファイルアクセス集中時の各ジョブの I/O バンド幅の分布を図 3 に，また，そのときの I/O バンド幅の最大値，最小値，平均値，及び，hongo クラスタ間の実験の平均値を基準にした相対性能を表 8 に示す．アクセス対象となるファイルが hongo クラスタに存在する場合，各ジョブのファイルアクセスは同一サイトのクラスタノード間のアクセスとなるが，その性能は平均して 15.6[MB/sec] となり，3.1.1 節の結果と比較して著しい性能低下を確認した．アクセス対象となるファイルがサイト間をまたいだクラスタに存在する場合 (suzuk, chiba, okubo, imade)，大多数のジョブが 3.1.2 節の結果と同等の性能を示した．これは，サイト間のネットワークバンド幅によりファイルアクセスの際の I/O バンド幅により律速されるので，アクセス集中の影響が相殺されるためである．ファイルが suzuk，chiba クラスタに存在する場合の実験では，hongo クラスタと suzuk,chiba の各クラスタが高バンド幅なネットワークで接続されているため，一部のジョブでは 17.4[MB/sec] と高い性能を示したが，低バンド幅なネットワークで接続された okubo, imade クラスタを用いた実験では，最大でも 3.34[MB/sec] 程度の性能に抑えられた．

4. 議　　論

4.1 空間的に遠方のノードへのファイルアクセス

同一サイトのクラスタノード間のリモートファイルアクセスでは，3.1.1 節の結果より，ノード間のファイルアクセス性能にばらつきがあることを確認した．特に，3.1.1 節の実験 (3) と (4) の比較により，ノード間のファイルアクセス性能には非対称性があり，同じノード間のファイルアクセスであってもアクセス元とアクセス先のノードが異なると性能が著しく異なる場合があった．また，3.1.1 節の実験 (4) と 3.1.2 節の実験 (1) との比較では，同一サイトのクラスタノード間

でファイルアクセスを行うよりもサイトをまたいだ別クラスタのノードへアクセスしたほうがファイルアクセス性能が向上した.

サイト間をまたがるクラスタノード間のリモートファイルアクセス性能は, 3.1.2 節の結果より, クラスタの存在するサイト間のネットワークバンド幅に影響を受けたが, 同程度の高バンド幅なネットワークで接続されたクラスタが存在した場合でも遅延の影響により性能のばらつきを確認した.

これらのファイルアクセス性能は実際にアクセスを行うノード間のバンド幅の影響を受けるが, これは, 物理的なネットワーク性能だけでなく, マシン構成の違い, チューニングなどの複合的な要因により決まるため, これらのうちの限定的な情報のみではファイルアクセス性能が向上するようなファイルシステムを構成するノード間の関係を正確に推定することは困難であることがわかった. このため, 実際のファイルアクセスの性能のプロファイルを元に, ファイルシステムを構成するノードの関係を推定する手法などが必要である.

4.2 ファイルアクセスの時間的な集中

リモートファイルアクセスの場合, 3.2.2 節の結果より, アクセス集中による性能低下の主な要因は, ネットワークバンド幅による律速であった. このような状況の場合は, サイトをまたいだクラスタノード間 (特にネットワークバンド幅が低いサイト間) でのキャッシングや複製作成が高い効果を期待できる. 一方で, ローカルファイルアクセスの場合, 3.2.1 節の結果より, アクセス対象となるファイルやそのサイズ, ノードのメモリサイズなどの関係により, 少数のファイルに対して繰り返しアクセスするようなアクセスパターンでは, メモリバンド幅程度の性能が期待できる場合があった. 一方, 複数のファイルに対して時間的に集中して同時アクセスされるようなアクセスパターンでは, ファイルのページキャッシュが効かなくなり, ローカルディスクへのアクセスが発生し競合状態が起こり, 大幅な性能低下 (最大で 0.1 倍) を引き起こすことを確認した. このため, アクセス対象となるファイルやそのサイズ及びノードのメモリサイズを考慮したデータ配置が必要である.

5. おわりに

本稿では, 効率の良い大規模データ管理を実現するためにどのようにファイルをグリッドファイルシステムを構成するノードに配置すれば良いかの戦略を決定づけるメトリックを明らかにするために, 広域に分散した 5 サイトからなる HPC クラスタを連携してファイルシステムを構成し, その上で, リモートファイルアクセスの行った際の性能を調査した. その結果, リモートファイルアクセス性能はノード間のバンド幅の影響を受けるが, RTT, バンド幅などのネットワークの限定された情報だけではファイルシステムを構成するノードの関係の推定が困難であること, ローカルファイルアクセス性能でもファイルアクセスパターンにより最大 0.1 倍の性能に抑えられてしまうこと, が明らかになった.

今後は, 実アプリケーションを構築したテストベッドで動作させ, 実際のアプリケーションのワークロードを想定にしたグリッドファイルシステム上のデータ管理手法を検討していく予定である.

謝辞 本研究の一部は本研究の一部は科学研究費補助金特定領域研究 (18049028) の補助により行われた. 実験環境 (InTrigger) の利用に関しては, 東京大学大学院情報理工学研究科田浦研究室にご協力を頂いた. この場を借りて感謝したい.

参考文献

1) 佐藤仁, 松岡聡 : データインテンシブコンピューティングのためのグリッドファイルシステム上のデータ管理, コンピュータシステムシンポジウム論文集, vol.2006 (ComSys 2006), pp. 29 – 36 (2006).
2) Tatebe,O., Morita,Y., Matsuoka,S., Soda,N. and Sekiguchi,S.: Grid Datafarm Architecture for Petascale Data Intensive Computing, in *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, pp. 102–110 (2002).
3) Gfarm, `http://datafarm.apgrid.org/`.
4) Machida, Y., Takizawa, S., Nakada, H. and Matsuoka., S.: Multi-Replication with Intelligent Staging in Data-Intensive Grid Applications., in *Proceedings of the 7th IEEE/ACM International Conference on Grid Computing*, pp. 88–95 (2006).
5) Lamehamedi,H., Szymanski,B., Shentu,Z. and Deelman, E.: Data Replication Strategies in Grid Environments, in *In Proceedings of the 5th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 02)* (2002).
6) Shirai, T., Saito, H. and Taura, K.: A Fast Topology Inference — A building block for network-aware parallel computing, in *The 16th IEEE International Symposium on High Performance Distributed Computing (HPDC 2007)*, pp. 11–21 (2007).
7) Qiang Xu, J., Subhlok: Automatic clustering of grid nodes, in *The 6th IEEE/ACM International Workshop on Grid Computing* (2005).