2007－HPC－113 (1)
2007／12／7

Sun Fire X4500とGfarmを用いた大規模ストレージの構築

谷村勇輔[†1] 山本直孝[†1] 石橋拓也[†1,†4]
田中良夫[†1] 西川武志[†2]
松岡聡[†2,†3] 関口智嗣[†1]

Sun Fire X4500は4ウェイのx86-64と24TBのストレージを統合したサーバであり，大規模データ処理の基盤となるストレージを構築するのに適したサーバの1つである．特に，Gfarmのファイルアフィニティを考慮したデータ処理モデルとの相性が良いと考えられる．そこで，本研究では20ノードのX4500とGfarmを組み合わせたストレージシステムの構築を検討した．Solaris 10が提供するZFS/RAID-Zの特徴を活用して，Gfarmにおいて十分なスループットが得られるように予備実験を行いながら，ZFSのストレージプールの構成を決定した．同様に，Gfarmのメタデータ操作についても性能が得られるようにメタデータを格納するファイルシステムを検討した．これらの検討を踏まえて，実際に256.5TBのストレージを構築して基本性能を検証し，それを拡張してPB規模のストレージを構築するための課題を明らかにした．

Building A Large-Scale Storage System Using Sun Fire X4500 and Gfarm

Yusuke Tanimura,[†1] Naotaka Yamamoto,[†1]
Takuya Ishibashi,[†1,†4] Yoshio Tanaka,[†1] Takeshi Nishikawa,[†2]
Satoshi Matsuoka[†2,†3] and Satoshi Sekiguchi[†1]

Sun Fire X4500 Server integrates a four-way x86-64 server and 24 TB storage, which may deliver remarkable benefits to the large-scale data processing applications. In particular, the server architecture is supposed to match a data processing model of Gfarm, which uses file-affinity scheduling. In this paper, system integration for building a storage system using 20 nodes of X4500 and Gfarm is discussed. Configuration of the ZFS/RAID-Z storage pool or UFS is determined so that Gfarm achieves significant performance in I/O throughput and metadata operations. According to the discussions and preliminary experiments, a storage system which has 256.5 TB capacity was constructed and the basic performance was measured by benchmarks. The results indicate several issues for building a petabytes-scale storage system with such as the architecture.

1. はじめに

大規模なデータをある程度の期間格納し，データ解析やデータ提供サービスを行う科学技術アプリケーションとしては，高エネルギー加速器の実験や地球科学分野の衛星観測など複数存在する．ビジネス分野においても大容量のデータを扱う事例が増えつつあり，類似のアプリケーションは多い．大規模なデータを格納する方法としては，従来はテープ装置などの利用が主流であったが，HDDの大容量化と低価格化に加え，データアクセスの迅速性が求められることが多くなり，ネットワーク化されたファイルサーバやストレージシステムにデータを保存したいという要求が高まっている．しかし，現状では単一のサーバが持つディスク容量は数十TBに限られるため，数百TB～PB規模の容量を持つストレージの構築には分散ファイルシステムや階層的なストレージ構成を検討する必要がある．

このような状況においてSun Fire X4500[1]は興味深いアーキテクチャを持つ．計算サーバとストレージサーバの両方の特徴を持ち，CPUにはAMD Opteron Dual Coreを2つ搭載し，ストレージとしては500GBのディスクを48台搭載して24TBのストレージ容量を実現する．24TBには冗長構成やシステム領域に使われる容量も含まれるため，実際にアプリケーションで利用できる容量はこれより少なくなるが，仮に16TBのディスクスペースを利用できた場合，64ノードを揃えれば1PBを実現できる．ディスクとメモリ間，ディスクとネットワーク間においても高スループットを達成する一方，RAIDコントローラは搭載されておらず，

†1 産業技術総合研究所 / National Institute of Advanced Industrial Science and Technology (AIST)
†2 東京工業大学 / Tokyo Institute of Technology
†3 国立情報学研究所 / National Institute of Informatics
†4 株式会社 創夢 / SOUM Corporation

ソフトウェア RAID の利用が想定されている．

複数の X4500 を利用したクラスタストレージの事例としては東京工業大学の TSUBAME[2] が挙げられる．TSUBAME では X4500 を Infiniband で接続し，Lustre[3] を用いて大容量，かつ高速にアクセスできる共有ファイルシステムを HPC の並列プログラムを実行するユーザ向けに提供している．ただし，現状の Lustre は Linux 上での動作にのみ対応しており，Solaris 10 や強力なソフトウェア RAID 機能をもつ ZFS に対応していないため，X4500 のアーキテクチャとの適性に議論の余地がある．また，Infiniband の導入には相応の費用がかかり，より安価に大容量のストレージを構築する方法が求められる場合もあると考える．

そこで本研究では，複数の X4500 を統合してクラスタストレージを構築するのに Gfarm を利用した方法を検討する．Gfarm は CPU とディスクを 1 つのセットと考えて，ファイルアクセスの局所性を利用して I/O スループットを得る設計になっており，X4500 との親和性が高いと考えられる．また，ネットワーク越しの I/O が頻繁に発生しないことが前提であるため，ノード間は Gigabit Eternet で問題ない．オープンソースのソフトウェアであるため，ハードウェア以外は無償提供されたソフトウェアだけでシステムを構築できる．

本稿では Gfarm での利用を想定して X4500 のシステム構成を検討し，実際に 20 ノードの X4500 を用いて 256.5TB のストレージを構築し，その性能評価を行った結果を報告する．以降，2 節にて X4500，3 節にて Gfarm の特徴を説明し，それをもとに 4 節にて大規模ストレージの設計を述べる．5 節に性能評価の結果を示し，最後にまとめと今後の課題について述べる．

2. Sun Fire X4500

Sun Fire X4500 は高さ 4U の筐体に 4 ウェイの x86-64 と 48 台の SATA II ディスクからなるストレージを一体化したアーキテクチャをもつ，新しいタイプのストレージサーバである．高速なデータアクセスを行うために，10 の内部 PCI-X バスをもち，カタログ仕様ではディスクとメモリ間のスループットは 2GB/sec，ディスクとネットワーク間のスループットは 1GB/sec である．外部ネットワークとの接続用に Gigabit Eternet のポートを 4 つ備えている．ただし，他の多くのストレージサーバとは異なり，RAID コントローラは搭載されておらず，Solaris ZFS（Zettabyte File System，以降 ZFS と記す.）[4] を利用して RAID-Z と呼ばれるソフトウェア RAID を構成することになる．

ZFS はボリューム管理と RAID 機能，ファイルシステムを 1 つに統合した仮想ストレージプールを提供する．ストレージプールの概念によりデータの格納領域を柔軟に構成可能であり，スナップショットの作成やマウントポイントの設定などファイルシステムの管理が非常に簡素化されている．また，トランザクションファイルシステムやデータの自己修復機能が実装されているため，堅牢なストレージを構築できる．特に，Copy-On-Write のセマンティクスが用いられるため，ディスクの Write Caching を有効にした場合でもデータの破壊が起きないようになっている．

ZFS においてデータの冗長化は RAID-Z によって達成される．RAID-Z は，RAID-5 のようにデータとパリティを同時に書き込むことで冗長性を確保するが，Copy-On-Write の仕組みで書き込むため，Write Hole の問題とパリティの更新が read-modify-write となることによる性能の問題を解決する．このように，ZFS/RAID-Z には NVRAM などの特別なハードウェアを用いなくとも信頼性の高いストレージを提供できる利点がある．

3. Gfarm

Gfarm[5] はネットワーク上に分散するストレージサーバを統合し，仮想的なディレクトリツリーを用いてファイルにアクセスできる分散ファイルシステムを実現する．大規模なデータを並列に処理できるよう，ストレージサーバ上の CPU とディスクを 1 つのセットと考えて，アクセスする予定のファイルが保存されたノードでプログラムを実行することで性能のスケーラビリティを得る．データアクセスにおいて I/O スループットの占める割合が大きく，データアクセスと同時に CPU 処理を要求するアプリケーションの実行に適した設計となっている．

Gfarm バージョン 1（本研究ではバージョン 1 を対象とするため，以降 Gfarm とのみ記す.）では，1 つのメタデータサーバと任意の数のファイルシステムサーバによって Gfarm ファイルシステムを構成する．メタデータサーバはジョブ情報を管理する gfmd とファイルやホスト情報を管理するデータベースサーバに分けられ，後者には PostgreSQL と OpenLDAP が利用可能である．ただし，10 万を越えるファイルを Gfarm ファイルシステムに格納する場合には PostgreSQL の利用が推奨される．バージョン 1.3 以降ではメタデータキャッシュサーバ（gfarm_agent）を起動して，一部のメタデータを複数のクライアントで共有することも可能である．ファイルシステムサーバ（gfsd）はストレージを提供するノード上で動作し，ファイルへの I/O アクセスを受け付ける．Gfarm ファイルシステムへのアクセスは Gfarm API ライブラリを用いてアプリケーションを作成するか，アプリケーションが利用するファイルシステムコールを Gfarm API にマッピングする上位ツールを利用するかのどちらかとなる．後者にはシステムコールフックライブラリと GfarmFS-

表 1　本研究で用いた Sun Fire X4500 のスペック

Hardware	Dual-core Opteron 2.6GHz×2, 16 GB memory, 48 disk drives (HITACHI HDS7250S), Marvell Serial-ATA controller×6, Intel 82546EB Dual port Gigabit Eternet×2.
Software	Solaris 10 (with update3 and recommended patches), PostgreSQL v8.1.9 (with default parameters).

FUSE[6] の 2 つのツールがあり，いずれも既存のアプリケーションを修正せずに Gfarm ファイルシステムにアクセスできる機能を提供する．

4. システムの構築

4.1 概　　要

20 台の X4500 を Gfarm で統合し，次のような検討を行いながら 256.5TB の容量を持つストレージを構築した．本研究で用いた X4500 のスペックは**表 1** の通りであり，ノード間の接続には Gigabit Ethernet を用いた．Gfarm はバージョン 1.4.1（2007 年 3 月 31 日リリース）を用いたが，このバージョンは Solaris 10/x86-64 をサポートしていないため，若干のコードの修正が必要であった．さらに，既存のアプリケーションから Gfarm ファイルシステムにアクセスするため，システムコールフックライブラリと GfarmFS-FUSE の利用を検討した．システムコールフックライブラリに関しては llseek() の動作に問題があったため，コードを修正して利用できるようにした．GfarmFS-FUSE については，OpenSolaris Porject の Fuse on Solaris[7] の開発状況を調査し，2007 年 6 月の時点でまだ利用できないことが明らかとなった．そのため，本システムではシステムコールフックライブラリを用いて Gfarm にアクセスすることとした．

Gfarm，および上位ツールの動作検証に加えて，Gfarm から X4500 のストレージにアクセスした時の性能が十分に得られるようにシステム構成を検討した．第 1 に，Gfarm のファイルシステムサーバは下位のファイルシステムに依存せず，実際にデータを格納する場所（スプール領域）をディレクトリパスで与える仕様である．しかし，Gfarm の I/O スループットは下位のファイルシステムの I/O スループットを反映したものとなるため，ファイルシステムサーバとなる X4500 では ZFS の特徴を踏まえてストレージプールの構成を考える必要があった．第 2 に，Gfarm のメタデータ操作の性能は，バックエンドとして用いるデータベース（DB）サーバのトランザクション処理の性能，およびクライアントとメタデータサーバ間の通信レイテンシに依存する．本研究では前者について，DB サーバが利用するファイルシステムについて検討した．以下にこれらの詳細を述べる．

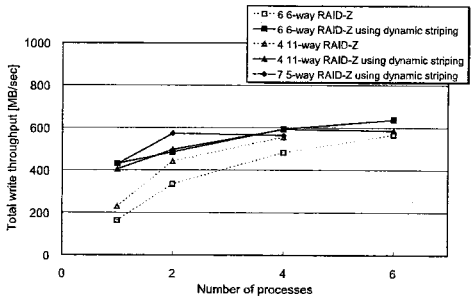

図 1　ZFS/RAID-Z の構成による Write 性能の違い

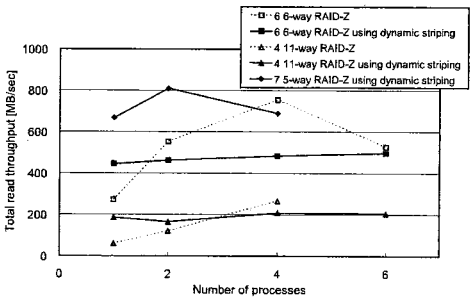

図 2　ZFS/RAID-Z の構成による Read 性能の違い

4.2 Gfarm のスプール領域となるストレージプールの構成の検討

まず最初に，X4500 のディスク構成，特に ZFS/RAID-Z に利用するディスクのレイアウトについて検討を行った．48 台のディスクのうち，2 台は OS をインストールするシステムディスクに用いるため，Gfarm には最大 46 台を割り当てることができる．本実験では，1 つの RAID-Z を構成するディスクの数，RAID-Z の数，最終的に RAID-Z を 1 つのプールとするか否かをパラメータとして 5 つのプール構成を試し，X4500 単体が達成する I/O スループットの合計値を比較した．各プール構成における Write 性能の違いを**図 1** に，Read 性能の違いを**図 2** に示す．“6 6-way RAID-Z” は 6 台のディスクを用いて 1 つの RAID-Z を構成し，それを 6 セット用意する構成である．“Dynamic striping” を利用する場合は複数の RAID-Z を 1 つのプールにまとめる構成となる．“Dynamic striping” を利用しない場合は，ファイルアクセスを行うプロセスが指定された RAID-Z にのみアクセスする形となる．なお，RAID-Z を 4 つ用意する構成は，X4500 が 4 ウェイであるために 4 つのプログラムが同時に異なる RAID-Z にアクセスする場合を想定して用意した．ZFS のパラメータにはデフォルト値を用いた．測定パラメータは “7 5-way RAID-Z using dynamic striping” についてはファイルサイズを 1,000MB，ファイル数を 140 と

表 2 Gfarm 経由の I/O スループット（単位 MB/sec）

	# processes	1	2	4
Write	7 5-way RAID-Z using dynamic striping	430	574	565
	Gfarm	189	307	415
	Gfarm (no checksum)	412	498	541
Read	7 5-way RAID-Z using dynamic striping	668	810	690
	Gfarm	211	374	523
	Gfarm (no checksum)	515	709	609

表 3 Write Caching の有無による合計 I/O スループットの違い（単位 MB/sec）

# processes	1	2	4
Write caching disabled	244	219	218
Write caching enabled	430	574	565

し，それ以外についてはファイルサイズを 10,000MB，ファイル数を 120 とした．

両グラフより，“Dynamic Striping” を利用して複数の RAID-Z を 1 プールにまとめることで，プロセスを 1 つしか起動しない時にも高いスループットが得られることが分かる．“Dynamic striping” を利用する構成のうち，“6 6-way RAID-Z” が Write において，“7 5-way RAID-Z” が Read において良い性能を示した．ただし，“Dyanamic striping” を有効にすると Write は良い値を示すが，Read は逆の傾向を示した．

次に，X4500 の個々の SATA ディスクの Write Caching の有無による I/O スループットの影響を調べた．先ほどと同じベンチマークを用い，“7 5-way RAID-Z using dynamic striping” の構成において比較を行った結果が**表 3** である．これより，Write Caching を無効にすると ZFS の I/O スループットがかなり低下することが分かった．

そして，“7 5-way RAID-Z using dynamic striping” の構成のストレージプールを Gfarm のスプール領域に設定し，システムコールフック経由で Gfarm を介してその領域にアクセスした時の I/O スループットを調べた．**表 2** に示すように，Gfarm を介してアクセスする場合は，4 プロセスによる同時アクセスにおいて約 25%のオーバーヘッドが存在する．このオーバーヘッドの詳細を調べたところ，Gfarm ではチェックサムの計算に時間がかかっていることが分かった．チェックサムの計算を停止すると，オーバーヘッドは Write で 4%，Read で 12%となった．

4.3 メタデータ領域のファイルシステムの検討

メタデータサーバにおいてはメタデータの格納場所，すなわち PostgreSQL の PGDATA の格納場所として UFS と ZFS の利用を検討した．**表 4** にファイルシステムの違いによる pgbench の結果とメタデータ操作の性能試験の結果を示す．pgbench は TPC-B 相当のベンチマークであり，メタデータサーバとして用いる PostgreSQL サーバの性能を測定する．本実験では 10,000 トランザクションを 1 クライアントから実行した．メタデータ操作の性能測定では，O_CREAT での open() と close()，O_RDONLY での open() と close()，unlink() の 3 つのメタデータ操作を指定回数実行し，その実行時間を測定するベンチマークを用いた．open() したファイルに対してデータの書き出しや読み込みは一切行わない．本実験ではそれぞれの操作を 10,000 回（10 ディレクトリ ×1,000 ファイル）ずつ実行した．また，両ベンチマークはメタデータサーバ上で実行した．

表 4 PGDATA の格納場所によるメタデータ操作の性能の違い

	pgbench [tps]	metadata ops. [sec]
UFS (noforcedirectio, 1 disk)	392,751,826	166
UFS (forcedirectio, 1 disk)	149	225
ZFS (1 disk)	91.6	869
ZFS (6 6-way RAID-Z with dynamic striping)	99.9	895
Tuned ZFS (2 disks)	106	838

結果より，“noforcedirectio” オプションを設定した UFS に PGDATA を格納する方法がメタデータ操作を最も高速にすることが分かった．この方法は pgbench の結果にばらつきがあったため表 4 には 3 試行の結果を全て示すが，最低値でも他の方法に比べて良い値である．そして，DB での利用において ZFS は UFS に比べてかなり性能が低いことが分かった．表 4 の Tuned ZFS は文献[8)9)] に従い，pg_xlog を保存するディスクの分離，ARC キャッシュサイズを 100MB に制限，ZFS のパラメータのチューニングを施している．しかし，本実験では Tuned ZFS を用いた結果は UFS を用いた結果よりも低い性能であった．なお，表 4 は PostgreSQL のパラメータにデフォルト値を設定した時の結果である．PosgreSQL の共有バッファを 1000 倍にして同様の実験も行ったが，メタデータ操作の性能に有意な差は見られなかった．

4.4 最終構成

これまでの結果を踏まえて，**図 4** に示すストレージシステムを実際に構築した．1 台がメタデータサーバ，19 台がファイルシステムサーバである．メタデータサーバでは 2 台のディスクをそれぞれ UFS でフォーマットし，メタデータ（PGDATA）とトランザクションのログ（pg_xlog）の格納場所として割り当てた．各

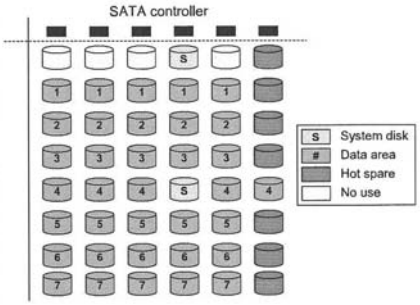

図 3 ファイルシステムサーバのディスクレイアウト

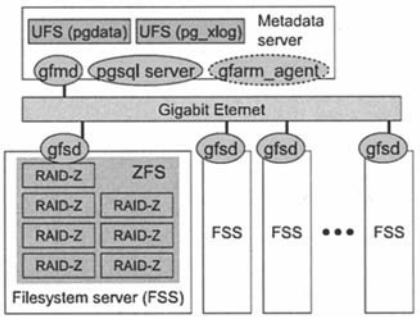

図 4 構築したシステムの全体構成

ファイルシステムサーバでは**図 3** に示す "7 5-way RAID-Z using dynamic striping" の構成を採用し，合計 13.5TB（フォーマット後に実際に利用できる容量は 12.5TB）のディスクスペースを提供した．Gfarm で利用しない 13 台のディスクのうち 2 台はシステム用のディスクとし，11 台はホットスペアとした．実運用を考えると "4 11-way RAID-Z" の構成はスペアの数が少ないために障害に弱く，各 RAID-Z に対して 1 台のディスクをスペアとして設定したり，RAID-6 相当の RAID-Z2 の利用が望ましいと思われる．本システムでは最終的に，Gfarm ファイルシステム全体として 237.5TB の容量を確保した．加えて，メタデータサーバと同じノード上でメタデータキャッシュサーバをマスターモードで起動した．

5. 性能評価

4 節で述べたように構築したストレージの性能評価として，I/O スループットを測定するベンチマークとファイルのメタデータ操作の速度を測定するベンチマークをそれぞれ適用し，ストレージシステム全体の性能を明らかにした．以下に実験方法と結果を述べる．

5.1 I/O スループット

I/O スループットの測定では，10,000MB のファイルを Gfarm ファイルシステムに対して 10 回書き出す（write(2)+fsync(2)）時間と，逆にそれらを読み込む（read(2)）時間を計測するベンチマークを用意し，それを並列に実行することで同時書き込み，同時読み込みにおける合計スループットを得た．**図 5** のグラフは，Gfarm を用いた場合，チェックサムを計算しない Gfarm を用いた場合，"7 5-way RAID-Z using dynamic striping" の構成をとる ZFS の 3 者においてスループットを比較している．4 プロセス以上の実験では 1 ノードあたり 4 プロセスを起動した．例えば，32 プロセスを起動する場合は 8 ノードを使用し，それぞれのノードで 4 つずつプロセスを起動した．チェックサムを計算しない Gfarm を用いた場合と ZFS の結果は，表 2 の 4 プロセス起動時の測定結果を，横軸の並列数に合わせて整数倍した計算値である．

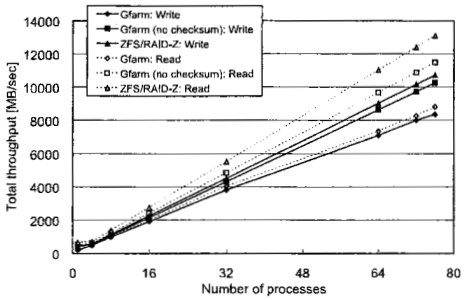

図 5 システム全体での I/O スループット

図 5 より，本システムの合計 I/O スループットは利用ノード数，および起動するプロセス数に従って線形にスケールすることが示された．チェックサムを計算する現状の Gfarm を利用した場合でも 76 プロセスによる同時アクセスにおいて，Write については ZFS の性能の 78%，Read については 67%が得られた．また，ZFS の計算値とは異なり，Gfarm によるアクセスでは Write と Read のスループットにはそれほど違いがなかった．

5.2 メタデータ操作

メタデータ操作の性能測定では 4.3 節で用いたベンチマークを用いた．さらに，5.1 節の実験と同様に，1 ノードあたり 4 プロセスを同時に起動し，各プロセスにおいて 10,000 回（10 ディレクトリ ×1,000 ファイル）のメタデータ操作を行った．**図 6** に示す結果では，Gfarm ファイルシステムにデータをインポートする前の状態とインポートした後の状態を比較している．インポートする前はファイル数は数百以下，ディスクスペースはほとんど使われていないのに対し，データをインポートした後では 1800 万ファイル，173TB のデータを格納し，全ディスクスペースの 73%が使用された状態になっている．

図 6 より，いずれの結果もメタデータ操作は 4 プロセスの時点で限界に達しているのが分かる．さ

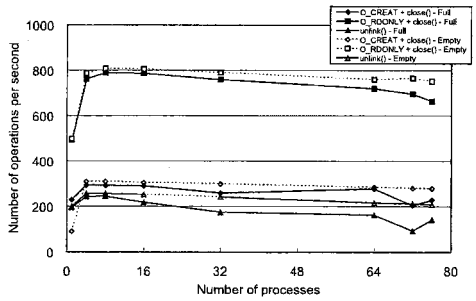

図 6 システム全体でのメタデータ操作の性能

らに，多くのデータが格納されている場合には性能が低下しており，4 プロセス以降の平均低下率は O_CREAT+close() では 11%，O_RDONLY+close() では 5%，unlink() では 24%となった．

5.3 考　　察

本実験により，X4500 と Gfarm を組み合わせたストレージにおいて，ZFS/RAID-Z が X4500 単体の I/O スループットの向上を担い，Gfarm が複数の X4500 のストレージ容量の統合と I/O の合算を提供することができ，Gfarm を利用することによるオーバーヘッドが 20~30%程度であることが明らかとなった．ただし，I/O スループットについては Gfarm 内部においてチェックサムの計算を行わないことで，オーバーヘッドを 4~12%に抑えることが可能である．メタデータ操作の性能はシステムの大規模化に全く対応できていないが，これは Gfarm バージョン 1 が抱える実装の問題に起因しており，バージョン 2 の開発において改善される予定である[10]．

一方，PB 規模のストレージの構築に向けては，性能だけでなくシステムの信頼性についての検討が必要である．ZFS は障害に対してデータの破壊や損失が起きないようにする機能を特徴としており，このような機能を Gfarm のスプール領域に適用できるのは，本システムの利点の 1 つである．しかし，X4500 のノードが停止してしまった場合には，Gfarm のレベルで冗長性が確保されていなければデータにアクセスできなくなる．つまり，ZFS の機能と Gfarm のファイル複製機能を組み合わせて高信頼を達成する仕組みの検討が今後の課題である．また，今回は性能を考慮して，DB サーバの PGDATA の格納に UFS を利用したが，メタデータに影響のある障害の発生を考慮したものではない．Gfarm のメタデータサーバの利用における ZFS の性能の問題を解決し，構成の異なるストレージプールにメタデータを格納する場合のメタデータ操作の性能を検証することも課題である．

6. おわりに

本稿では Sun Fire X4500 と Gfarm を用いて大規模ストレージを構築するための方法について検討した結果を述べ，実際にシステムを構築して I/O スループットやメタデータ操作の性能を明らかにした．今後は PB 規模のストレージの実現に向けて，メタデータ操作の性能の改善や，考察で述べたようなデータの破壊や損失に耐性のある仕組みを ZFS と Gfarm の組み合わせによって検討し，システム全体として高信頼を実現する方法に取り組んでいきたい．

謝辞　本研究は東京工業大学と産業技術総合研究所の共同研究「衛星データ解析アプリケーションによる大規模ストレージシステムの性能評価」による成果である．本研究の遂行にあたり，実験環境の構築にご協力頂き，かつ貴重なコメントを頂いた東京工業大学学術国際情報センター，サン・マイクロシステムズ（株）の皆さまに感謝いたします．

参 考 文 献

1) Sun Fire X4500 Server. http://www.sun.com/servers/x64/x4500/.
2) 松岡聡. TSUBAME の飛翔：ペタスケールへ向けた「みんなのスパコン」構想. 情報処理学会研究報告 HPC-107, pp. 37–42, 2006.
3) Lustre. http://www.lustre.org/.
4) E. Kustarz. ZFS – The Last Word in File Systems. http://www.opensolaris.org/os/community/zfs/
5) O. Tatebe and et al. Grid Datafarm Architecture for Petascale Data Intensive Computing. In *Proceedings of the 2nd IEEE/ACM Symposium on Cluster Computing and the Grid (CCGrid)*, pp. 102–110, 2002.
6) GfarmFS-FUSE. http://datafarm.apgrid.org/software/gfarmfs-fuse.ja.html.
7) Fuse on Solaris. http://opensolaris.org/os/project/fuse/.
8) N. Nadgir. Databases and ZFS. http://blogs.sun.com/realneel/entry/zfs_and_databases, 2006.
9) ZFS Best Practices Guide. http://www.solarisinternals.com/wiki/index.php/ZFS_Best_Practices_Guide.
10) 建部修見ほか. 広域分散ファイルシステム Gfarm v2 の設計と実装. 情報処理学会研究報告 HPC-99, pp. 145–150, 2004.