大規模仮想ディスクにおける故障時の性能評価

チャイ エリアント　　上原 稔　　森 秀樹

東洋大学大学院工学研究科情報システム専攻

現在では情報技術の進歩によってデータのやり取りが多くなってきた。そのため、データを保管するためのストレージの容量も増えつつある。ところが、ストレージの容量がどんなに大きくなっても、一般に使用されているローカルのストレージの容量は限られている。また、ローカルのストレージのサイズを超える大きなファイルの保存も不可能である。本システムはネットワークを用いて、教育環境などの数百台 PC の開き領域を使用し、安価な高信頼・PB 級の分散ストレージを構築し、無故障時と故障時の性能を評価する。

Performance Evaluation at Failure in a Large-Scale Virtual Disk

Erianto Chai　　Minoru Uehara　　Hideki Mori

Toyo University Graduate School, Dept. of Open Information System

Nowadays, the exchange of data has increased by progress of an information technology. Therefore, the capacity of the storage for keeping data is also increasing. However, even if the capacity of global storage may become large, the capacity of the local storage is limited. Moreover, preservation of the big file whose size is over the size of local storage is also impossible. Using a network, this system constructs a cheap, high trust and PB class decentralized storage by using many hundreds PC of educational environment. Thus, this paper evaluates the performance of large-scale virtual disk at the time of trouble-free, and the time of failure.

1 はじめに

今日、情報が氾濫する情報洪水時代大規模ストレージが必要である。大規模ストレージはヒトゲノム、計算工学など、大量データの研究に扱われる。また、ビデオ、音楽データなど、個人で大量データが使用される。教育環境でも、仮想マシンなどを利用するために大規模ストレージを必要としている。教育環境では信頼性や管理の容易さを重視するため、数 TB の HDD 容量を持つ高価なファイルサーバを導入することが多い。しかし、このようなファイルサーバはより安価なシステムを求める教育現場のニーズと乖離している。ここで、一例をあげる。60TB のファイルサーバ（ETERNUS NR1000F や 336 台の 300GB の HDD など）を定価で見積もると 2.5 億円になるが、120GB の HDD を持つ 500 台の PC からなるシステムは HDD 単価 1 万として 500 万円で済む。そのコスト比は 1:50 になる。信頼性等は重要であるが、これほどコストが違うと異なる選択肢も考えられる。

高価な集中型ストレージのかわりに NFS(Network File System)などの分散型ストレージが採用される。しかし、分散型でも相互運用性が Linux に依存、管理が困難、信頼性が低い、性能が悪いという問題点がある。また既存の分散型ファイルストレージはディスク容量を 100%利用できないという問題点もある。例えば、120GB ずつ集めて 60TB のストレージを構築しても、それぞれのファイルは 120GB を超えることはできない。また、1 つのディレクトリの中でファイルの合計サイズが 120GB を越えることもできない。さらに、仮想ファイルシステムは下位層のファイルシステムに依存するためファイルサイズの上限が 2GB に制約されることもある。このように NFS をはじめとするファイルシステムレベルの分散ストレージでは空き容量を連結して 1 つのストレージにすることができない。

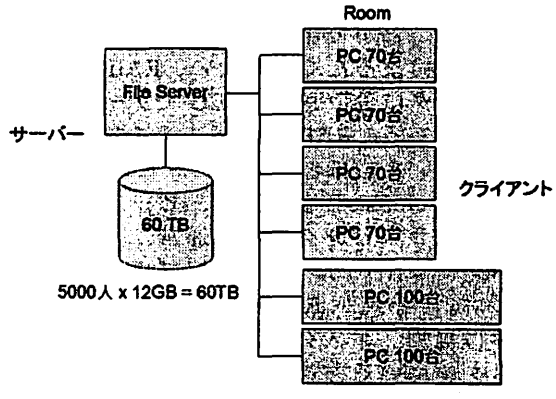

図 1 大学の PC 教室

図 1 に示すように大学ではストレージを利用する人数の平均は 5000 人で、一人当たりに 12GB が必要のため、60TB のファイルサーバが導入された。クライアントには 70 台の PC が 4 室と 100 台の PC が 2 室ある。全ての PC はひとつの値段の高い 60TB 集中型ファイルサーバに接続している。

本研究では数百台の PC から構成される PC 教室の遊休資源（ディスクの空き容量）を連携して一つの大きなストレージを構築する。PC の使い切れない HDD の空き容量を集めて、うまく使えば、高価なストレージの必要性がなくなる。経済の点から見ると

遊休資源を集めた分散型ストレージは、ハードディスク代のみなので、集中型の値段と比較すると 50 倍ぐらい安い。NBD（Network Block Device）などを利用することによってディスクレベルの分散型ストレージの実現ができ、既存の分散型ファイルシステムの問題点を解決することができる。分散型ストレージでは仮想的なディスクを構築することでファイルシステムに依存しない運用が可能となる。また既存の分散ファイルシステムでは、不可能である 1 台のディスク容量を超えるファイルの保存を可能にする。

本研究で開発する大規模ストレージの目的は、PC 教室などの教育環境の数百台の PC で安価な高信頼・大規模ストレージを構築することである。本システムのファイルサーバは 64 ビット Linux で構築され、Linux クライアントからは NFS、Windows クライアントからは CIFS (Common Internet File System)でアクセスされる。また、信頼性をあげるために RAID66 を採用する。

2 関連研究

2.1 RAID [1][8]

今大規模ストレージの信頼性を上げるために RAID（Redundant Arrays of Inexpensive Disks、レイドと読む）を用いる。RAID とは、記憶すべきデータと障害回復のための冗長データを複数のハードディスクドライブに分散して格納することで、パフォーマンス（性能）とフォルトトレラント（耐障害）性を同時に確保するための技法である。冗長データの種類と各ディスクドライブへのばらまき方によって、RAID は 0 から 6 までのレベルがある。

RAID0 は冗長性なしのストライピングである。容量を拡大することができる。RAID1 はミラーリングである。容量はまったく増えない。RAID2 は、各ビットをディスクに分散させ、ECC でデータ誤りを訂正する方式である。RAID2 はパリティ方式に比べて優位性が少ないためほとんど使われない。RAID3 はパリティによって誤りを訂正し、ビットまたはバイト単位でストライピングする。通常、パリティは専用ディスクに保存する。RAID4 はブロック単位でストライピングする点が RAID3 と異なる。RAID3 同様パリティは専用ディスクに記録される。パリティ専用ディスクを用いる方式はそれがボトルネックとなる。RAID5 はパリティをディスクに分散して保存する。ボトルネックが存在しないため性能が高い。RAID 5 システムでは、オペレーターは時々駆動中のドライブを引き間違って、2 つのドライブが同時に失敗するようなことがある。そこで、RAID 5 より信頼性が高い RAID 6 が採用される。RAID6 はパリティデータを 2 重に作成することで、2 重障害に対応することができる。つまり、同時に 2 台のディスクが故障してもデータを失うことなく回復できる。

RAID は実装によりソフトウェア(SW)方式とハードウェア(HW)方式に分類される。現在、主として用いられているのは HW RAID である。CPU の負荷のため SW RAID は HW RAID より性能が低いと考えられてきた。しかし、最近の CPU は HW RAID コントローラより高い性能を持つ。ファイルサーバのように CPU が RAID 演算に専念できるマシンでは SW RAID も有効である。また、ネットワークを越えて RAID を構成する場合、SW RAID は唯一の解である。

2.2 RAID6 [2]

RAID6 ではパリティが 2 二つある（P パリティと Q パリティと名づけ）。P パリティは RAID5 と同じくすべてのディスクのブロックデータの XOR から生成される。RAID6 で使われる Q パリティは P パリティの計算より複雑である。Q パリティの生成には Galois Field 演算によって求められる。GF（Galois Field）は代数学では有限体と呼ぶが、計算機関連の分野では、ガロア体またはガロア域とも呼ぶ。GF は有限数の要素を含んでいる値のセットである。2^8個要素の GF は GF(2^8)と表示され、整数 0 から 2^{8-1}までの要素がある。RAID6 で 2 台のディスクが故障した時に、4 つのケースが考えられる。P と Q ドライブが故障、P とデータドライブが故障、Q とデータドライブが故障、2 台のデータドライブが故障である。

2.3 NBD [3]

仮想ディスクの 1 つに NBD がある。NBD は、Linux カーネルの拡張の一つである。リモート・サーバが提供するブロックデバイスを、ネットワークを経由してローカルのブロックデバイスのように扱うことができる。ブロックレベルで入出力を行うため、任意のファイルシステムを上位層に構築できる。同様のブロックレベル I/O として iSCSI などがある。これらと比較して、NBD は標準で Linux に採用されているため、インストールも容易である。また、プロトコルも単純である実装が容易である。ただし、本論文で提案するシステムが NBD に依存するわけではない。NBD は選択肢の一つに過ぎないことを付け加えておく。

2.4 Samba [4]

Windows のクライアントから Linux や UNIX などのファイルがアクセスできるように Samba を用いることが多い。Samba は、マイクロソフト社のネットワークシステムを実装したフリーソフトウェアである。Linux、Solaris、FreeBSD などの UNIX 系 OS を用いて、Windows のファイルサーバやプリントサービスを提供する。1992 年、Andrew Tridgell によって最初のバージョンが開発され、後に GPL にて公開された。本研究では、Samba を Windows クライアントからの要求を NBD に伝達するために使用する。

2.5 NFS [5]

NFS はあるマシンから他のマシンへと、ネットワークを通じて ディレクトリとファイルを共有することを可能にする。NFS を使うことで、 ユーザやプログラムはリモートシステムのファイルをローカルファイルであるかのようにアクセスすることができる。NFS は UNIX でデファクトスタンダードな分散ファイルシステムである。Linux でも標準でサポートされている。また、Windows でも Service for UNIX(SFU)を用いて NFS を利用することができる。ただし、SFU を用いた運用は Samba を用いた運用ほど容易ではな

い。そのため、クライアント OS によって適切なファイルシステムのプロトコルを選択しなければならない。本研究では、NBD を用いてディスクレベルでネットワーク化するため、NFS によるファイル共有は行わない。ただし、Linux クライアントからの要求を NBD に伝達するために使用する。

2.6 XFS [6][7]

XFS は業界最先端の高性能ファイルシステムとして広く認識されて、システムクラッシュからの迅速な復旧や非常に大規模なディスクシステムのサポートを提供する。XFS は、現在 Linux で利用可能なジャーナルファイルシステムとしては最初のファイルシステムであり、1994 年の後半より実行環境での実績を重ねている。XFS は 64bit ファイルシステムやアロケーショングループという手法を採用することで巨大なファイルシステムを扱うことを可能にした。また、サイズの大きなファイルは書き込み／読み込み領域を確保したり、ディスク上の空き領域を検索するのに時間がかかるといった問題がある。XFS では、遅延アロケーションや B+-Tree によってこれに対処している。XFS で構成したディスクの空き容量が不足した場合、ディスクを増設して論理ボリュームを拡大させ、umount せずに簡単にファイルシステムのリサイズをすることができる。XFS は不要なディスク・アクティビティーを多く発生させない。

3 システム構成

本研究ではディスクレベルで空き容量を連結して 1 つの 69.7TB ストレージを構築する。システムはディスクレベルなので、部分ディスクサイズを越えるファイルの保存も可能である。本研究は 512 台のディスク（1 ディスク=170GB）を 32 グループにして RAID66 を構築する(図 2 に示す)。

図 2 RAID66 の構成図

図 3 に示すように本システムは 64 ビットファイルサーバとディスクサーバがある。ディスクサーバの Linux や Windows などの OS からなる仮想ディスクはディスクの読み込みや書き込みを Java の RMI で機能を提供する。ファイルサーバの方は用意されたディスクに接続して RAID66 を構築する。RAID66 とは、2 階層の RAID6 である。NBD Server はその RAID66 を利用して NBD Client からのアクセスを待つ。そして、NBD Client の起動をした後、XFS でフォーマットする。Windows のクライアントは Samba を介してそのファイルサーバをアクセスする（Linux の場合は NFS）。

NBD プロトコルはセキュリティに欠けるため、ネットワーク上で運用するのは危険である。しかし、我々の方式では 1 台のサーバ内のプロセス間通信として NBD を用いているため安全に運用できる。実際の C/S 間通信はセキュリティを考慮した RMI に基づくプロトコルで実現される。

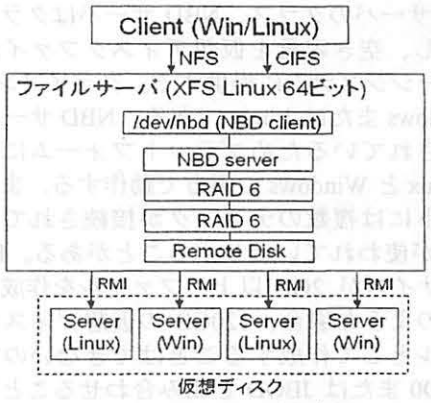

図 3 システム構成図

4 実装

ディスクサーバ側（512 台）では次のようにディスクサーバを起動する。

```
# java DiskServerImpl //localhost/pcn/DiskServerImpl imagen*
```

ここで、imagen は 170GB の仮想ディスクである。

ファイルサーバ側（1 台）では以下のように実行する。

```
# java vlsd.server.RAID66 9000
# nbd-client localhost 9000 /dev/nb0
# mkfs.xfs /dev/nb0
```

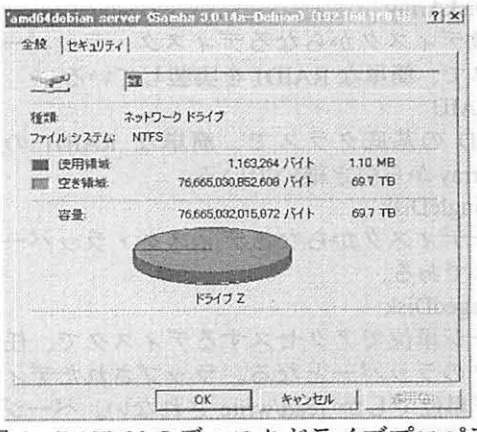

図 4 RAID66 のディスクドライブプロパティ

この後、実行が成功した場合、Windows のネットワークドライブの割り当て機能でストレージを使用できる。図 4 は共有したディスクのディスクドライブプロパティである。69.7TB の容量を確認できる。170GB ディスク 512 台の容量は 85TB であるが、RAID6 ではグループあたり 2 台のパリティを必要とするため、N 台からなる RAID6 は(N-2)/N しかデータを格納できない。RAID66 は 2 階層となるため全体の容量の 82%(=30/32*14/16)しか利用できない。よって、69.7TB となる。なお、今回の試作では容量可変の仮想ディスクを用いている。実際に、69.7TB までデータを保存したわけではない。多くのファイルシステ

ムはフォーマットにおいて一部のディスク領域しか変更しない。
以下は実験で使われたプログラムの説明である。

- NBDServer

NBD サーバのクラス。NBD サーバはクライアントで動作し、空き容量を仮想ディスクファイルとしてストレージシステムに提供する。クライアントの OS は Windows または Linux である。NBD サーバは Java で実装されているためプラットフォームに依存しない。Linux と Windows の両方で動作する。また、クライアントには複数のディスクが接続されていたり、FAT32 が使われていたりすることがある。FAT32 ではそのサイズが 2GB 以上のファイルを作成できない。これらのような場合、120GB の仮想ディスクを単一ファイルとして作成することはできないので、後述の RAID0 または JBOD と組み合わせることで複数のファイルを束ねて仮想ディスクを実現することができる。

- DiskServer

ディスクサーバのインターフェースである。

- DiskServerImpl

ディスクサーバのインプリメンテーションであって、RMI による遠隔ディスクを提供する。

- Disk

仮想ディスクが備えるべきインターフェースである。

- AbstractDisk

抽象的な仮想ディスクのクラスであって、下位クラスで使う定数やメソッドを定める。

- DiskArray

複数ディスクからなるディスク・ラッパーの基底クラスで、簡単な RAID1 を実装している。

- RAID

RAID の基底クラスで、簡単な RAID1 の実装を DiskArray から引き継いでいる。

- SingleDisk

単一ディスクからなるディスク・ラッパーの基底クラスである。

- PagedDisk

ページ単位でアクセスするディスクで、任意のディスクのラッパーとなる。ラップされたディスクはページ単位でしか read/write されない。ページ単位の端数は無視される。

- VariableDisk

可変容量ディスクであって、事前に資源を割り当てず、必要に応じて動的に資源を確保する。作成時に指定したサイズを超えて資源を使用することはないが、論理的なディスクサイズより大きくなることがある。

- RemoteDisk

ディスクサーバをアクセスする遠隔ディスクである。

- RAID0

RAID0 仮想ディスクのクラス。RAID0 は容量を増やすために使われる。後述の JBOD とはストライピングを行う点が異なる。若干性能はよいが、容量は最小サイズのディスクに合わされる。例えば、100GB, 120GB, 160GB を連結しても 100GB×3 にしかならない。純粋に容量増を目的とする場合 JBOD を用いたほうがよい。逆に、RAID0 は性能向上が期待できる場合がある。一部のファイルシステムでは i-node を管理するスーパーブロックが集中して配置される。このようなファイルシステムでは、規模が大きくなると特定のディスクにアクセスが集中する。このような場合、ストライピングは負荷を分散する効果がある。RAID0 はバイト単位でストライピングする。

- RAID5

RAID5 仮想ディスクのクラス。パリティを各ディスクに分散して格納する。パリティを格納するディスクはブロックごとに異なる。

- RAID6

RAID6 の実装で、GF テーブルの生成、ブロック単位ストライピングと分散パリティが行われる。

- RAID66

RAID6 を二段にした RAID である。

- JBOD

JBOD 仮想ディスクのクラス。RAID0 と同様に冗長性がなく、容量増のために用いられる。ストライピングを行わないため容量は単純に総和となる。例えば、100GB, 120GB, 160GB を連結すると 380GB になる。RAID0 には負荷分散の効果があると述べたが、JBOD には負荷を集中させる効果がある。ある程度の規模まではキャッシュが有効に働くため、RAID0 より性能がよくなる可能性がある。

5 評価

5.1 固定長ディスクと可変長ディスクの比較

大容量ディスクを実現するために可変長ディスクが必要である。しかし可変長ディスクの性能が固定長ディスクよりも著しく劣っていては問題である。そこで、ここではフォーマット時間（XFS）を比較することで可変長ディスクの評価をする。

可変長ディスクと固定長ディスクは仮想ディスクであってファイルとして保存される。可変長ディスクは固定ディスクと違い、必要とする分だけイメージファイルに書き込みをする。

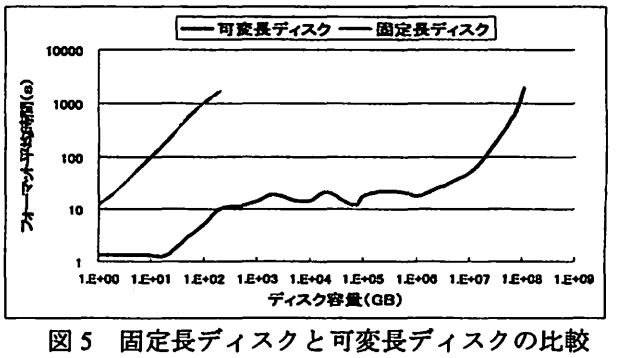

図 5 固定長ディスクと可変長ディスクの比較

図 5 のグラフから見ると、可変長ディスクの方がフォーマットする時間が速い。理由は仮想ディスク

ではディスクのシークする時間が速いためと考えられる。

可変長ディスクを XFS でフォーマットしたときにフォーマットに必要な容量を記録した。表 1 に示すように GB 単位でディスクをフォーマット時に実際に書き込まれたディスクのサイズは何十 MB で、TB 単位になると何百 MB のサイズが必要である。そして、10PB のとき 500MB の実際の物理ディスクが必要で、100PB のとき一気に 3.8GB までいく。

表 1　仮想ディスクのフォーマット

可変長ディスク	
仮想ディスクの容量	フォーマットに必要な容量
1GB	10.45MB
10GB	10.66MB
100GB	50.78MB
1TB	129.39MB
10TB	129.41MB
100TB	132.07MB
1PB	165.45MB
10PB	498.58MB
100PB	3829.71MB

5.2 無故障時の RAID の評価

RAID の比較には文献[8]における Small Reads, Small Writes, Large Reads, と Large Writes を行った。Small Read/Write は同一ディスクに対する読み書きであり、Large Read/Write はグループ全体に及ぶ読み書きである。図 6 から図 9 までは RAID0 に対する RAID5 と RAID6 の相対的な価格スループット比である。RAID5t、RAID6t は RAID5、RAID6 の論理値で RAID5e、RAID6e は RAID5、RAID6 の実験値である。RAID5e と RAID6e のグラフは指数近似の近似曲線で表す。

図 6 に示すように RAID5 と RAID6 の実験値は RAID5 と RAID6 の論理値にほぼ等しい。これはコストパフォーマンスがほぼ等しいということである。RAID5 と RAID6 では単一ディスクへのアクセスでまとまったデータを読み取ることができるため、Small Read では RAID0 と等しい。

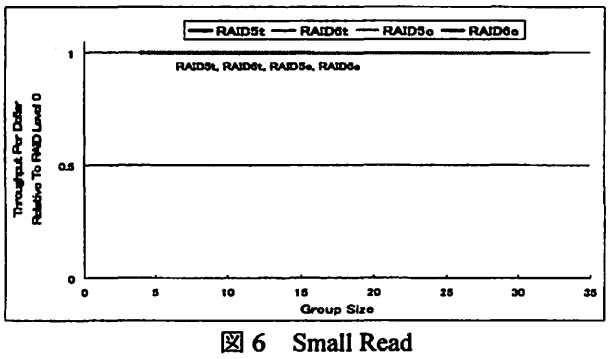

図 6　Small Read

図 7 では RAID5 と RAID6 の実験値は RAID5 と RAID6 の論理値の 4 倍ぐらい速い。これは RAID0 が速いと考えられる。RAID6 ではパリティデータが 2 つあることで、RAID5 よりパリティの計算が複雑になって、パリティの生成で処理時間が大きくなってしまう。それで図 7 で示すように RAID6 は RAID5 の下にあって、RAID5 より遅いとわかる。RAID5 と RAID6 の Small Write では RAID0 と違って、データの読み出し、パリティの読み出し、データの書き込み、パリティの書き込みの 4 つのアクセスが発生する。実験値は論理値と違ってグループが大きいとコストパフォーマンスが少しよくなっていく。

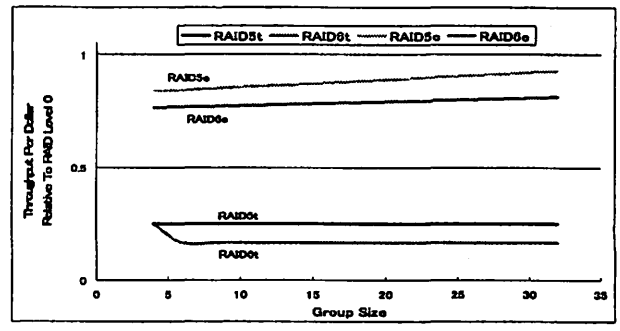

図 7　Small Write

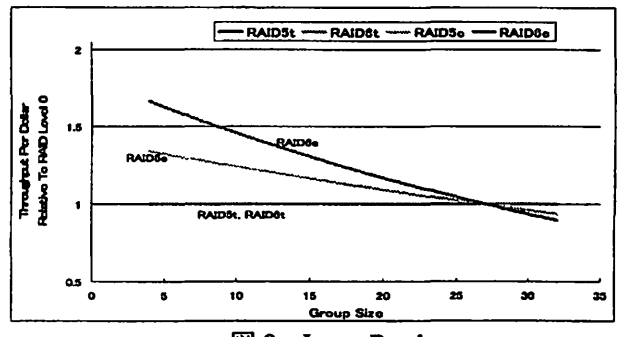

図 8　Large Read

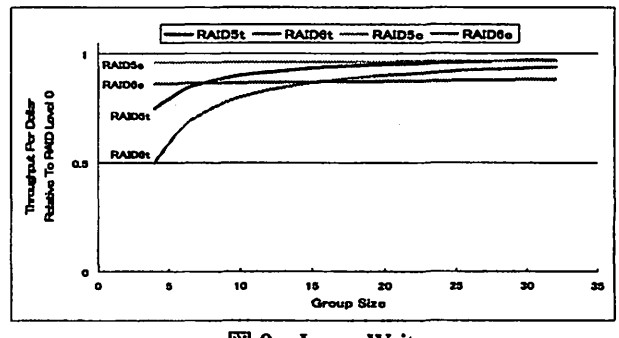

図 9　Large Write

Large Read の実験値（図 8）では小さいグループのときコストパフォーマンスが非常に高いが、大きいグループのとき低くなる。Large Read では RAID6 の読み込み領域が RAID5 より小さいため、コストパフォーマンスが高い。大きいグループではコストパフォーマンスが落ちたのは何らかの計算処理が増えたと考えられる。

図 9 では実験値がほぼ一定となっている。RAID6 の論理値は大きいグループになると RAID6 の実験値を超える。

5.3 故障時の RAID の評価

図 10 から図 13 までは無故障 RAID6 に対する RAID6 の 1 ディスク故障（RAID6e1）と RAID6 の 2 ディスク故障（RAID6e2）の相対的な価格スループット比である。RAID6e1 と RAID6e2 のディスクアクセス数は同じだが、異なる点は RAID6e2 のほうが処理演算がかかる。

Small Read（図 10）ではディスクが 1 つ故障のときコストパフォーマンスが 25%に落ちて、ディスクが 2 つ故障のとき 10%まで低くなる。RAID6e1 と RAID6e2 はグループが大きくなるとコストパフォーマンスが低くなっていく。Small Read ではディスクのアクセスが少なくほとんどの処理時間は演算処理だと考えられる。

図 11 に示すように Small Write では RAID6e1 のスループットが 80%から 60%ぐらいで RAID6e2 のスループットが 70%から 40%ぐらいである。Small Write では Small Read より高いスループットが得られたのはディスクアクセスが遅いからと考えられる。

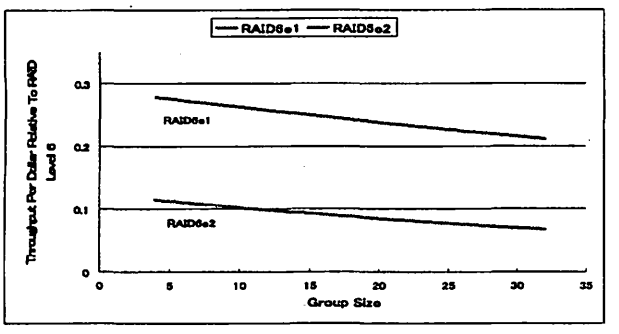

図 10　Small Read

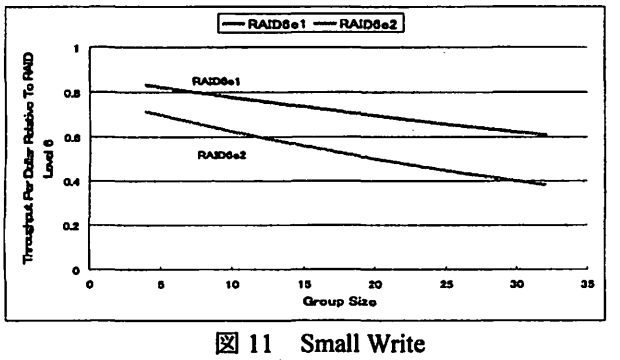

図 11　Small Write

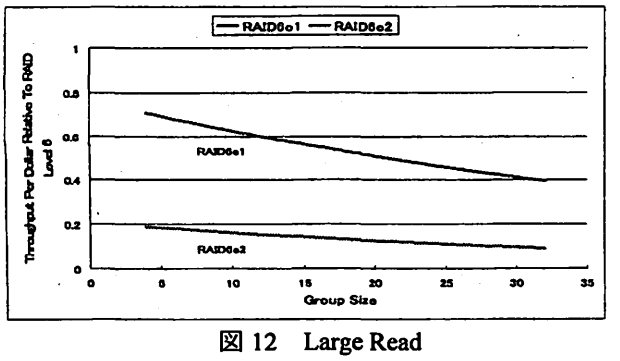

図 12　Large Read

Large Read（図 12）では RAID6e1 と RAID6e2 の違いが 4 倍になっている。Small Read と同じく Large Read では RAID6e1 と RAID6e2 のスループットの差がある。一方、ディスク書き込みを行う処理である Small Write と Large Write（図 13）ではディスク書き込み時間が演算処理よりはるかに遅いためディスク読み込みを行う Small Read と Large Read ほどスループットの差がない。Large Write ではディスク書き込み処理が増えるため大きいグループになるとコストパフォーマンスがよくなる。

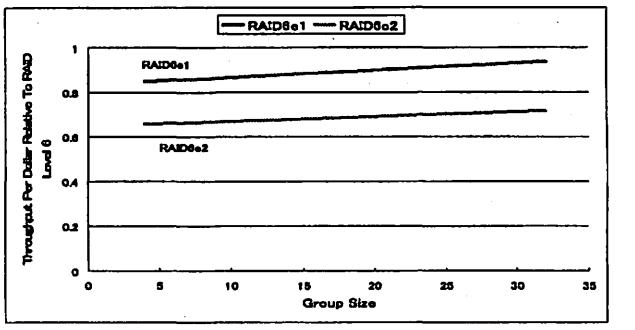

図 13　Large Write

6　まとめ

本研究では RAID66 で 69.7TB の大規模仮想ディスクを構築し、その性能を評価した。評価では理論値と異なる結果が得られたが、これは実証上の問題と考えられる。今後は、性能をあげたり、サーバ遠隔管理システムを作成したり、ボトルネックを解消したりする。

参考文献

[1] 宇野俊夫:「ディスクアレイテクノロジ RAID」, エーアイ出版株式会社, pp. 69-70 (July 2000)

[2] Intelligent RAID 6 Theory Overview and Implementation, http://download.intel.com/design/storage/papers/30812202.pdf

[3] IBM Linux Hint & Tips, http://www-06.ibm.com/jp/linux/developers/techinfo/nbd.pdf

[4] Samba, http://www.samba.org/samba/what_is_samba.html

[5] S. Shepler, et. al. : NFS version 4 Protocol, http://www.ietf.org/rfc/rfc3010.txt

[6] XFS 高性能ジャーナルファイルシステム, http://www.sgi.co.jp/projects/xfs

[7] XFS, http://www.atmarkit.co.jp/flinux/rensai/fs07/fs07a.html

[8] Peter M. Chen, Edward K. Lee, Garth A. Gibson, Randy H. Katz, and David A. Patterson: "RAID: High-Performance, Reliable Secondary Storage," ACM Computing Surveys, Vol. 26, No. 2, pp. 145-185, June 1994