HA8000のファイルシステムについて (II): Lustre編

田浦健次朗
情報理工学系研究科・情報基盤センター (兼務)

1　はじめに

2010 年 1 月号の本誌記事 [3] (以下, **「前記事」**) において述べた通り, 2010 年 4 月から, 新しいファイルシステムとして, Lustre ファイルシステム [1] の運用が開始されました.

「/lustre/ユーザ名」というディレクトリ名で, すでにアクセス可能です. 何も申請しなくても, 1 ユーザあたり 10 GB までの領域が割り当てられており, すぐに使い始める事ができます. 正確には,「課金グループに対して 10 GB× そのグループのユーザ数」が割り当てられていますので, 5 人のグループであればグループ全体で 50GB という事になります. それ以上使いたい場合は, 既存の領域を振替える事が可能です (新たな課金は発生しません) ので, 積極的にご利用ください. 詳しくは, HA8000 FAQ[4] 「システム全般, サービス内容」の節などをご覧ください.

現在のファイルシステム構成と領域の割り当て方針をまとめると, 表 1 のようになります.

表 1: ディレクトリと, それを提供するファイルシステム, 運用方針

ディレクトリ	ファイルシステム	quota
/home/ユーザ名	HSFS	申請による
/short/ユーザ名	HSFS	5 日間まで自由
/nfs/all/ユーザ名	NFS	10 GB まで自由
/nfs/{ グループ名,personal}/ユーザ名	NFS	申請による
/lustre/ユーザ名	Lustre	(∗)
/tmp	ローカル (ext3)	1 ジョブ実行中自由

∗:「グループごとに, 10 GB× グループのユーザ数」までは自由.

本稿では導入された Lustre の基本性能と, Lustre で高い IO 性能を得るための方法を紹介します. 結果を理解する上で必要な, ファイルシステムに関する背景知識や, HA8000 のストレージとネットワークの構成などについては前記事も参照ください. また, 性能測定のベンチマークも前記事と同じソフトウェア (paramark[2]) を用い, 同じ内容で実施しています. その意図や詳しい説明などについても適宜, 同記事を参照ください. NFS, HSFS の性能については本稿でも再掲しています.

2　ファイルを作る・開く性能

2.1　1 クライアント

まず, ファイルを新規作成する, および既存ファイルを開くのにかかる時間を単独で取り出してみます. どちらもシステムコールとしては open および close システムコールの経過時間を測定する事に相当します.

新規に作成するには,

```
int fd = open(path, O_CREAT|O_WRONLY|O_TRUNC, 0644);
close(fd);
```

を, 既存ファイルを開くには,

```
int fd = open(path, O_RDONLY);
close(fd);
```

を, それぞれ 1024 回繰り返し, 1 秒間の操作回数を測定します. これを 10 回繰り返し, 95%の信頼区間を求めています (以降の実験でも同様). 前者の場合, 作成されるファイルの名前 (path) は, 元々存在していないファイルの名前です.

表 2: 作成と読み込みの速度 (単位: 回/秒)

	HSFS	NFS	Lustre
作成	13.23 ± 1.10	2745 ± 21	982 ± 255
読み込み	170.38 ± 30.44	2862 ± 53	1282 ± 13

表 2 がその結果です. NFS は単一サーバ, かつサーバ側には状態を保持しないという性質により, open 自身は高速です. Lustre は NFS と同じとまでは行きませんが, NFS の 40%程度の性能を持っています.

2.2 多クライアント

次に, 並列度 (クライアント数) をあげた時の性能を見ることで, ファイルサーバのファイル open/creat のスループット (容量) を測定します. 図 1 は, 横軸にクライアント数, 縦軸に全クライアント合計での回数をとりプロットした物です. HSFS の場合, 作成は 60 回/秒, 読み込みは 1400~1600 回当たりで飽和するようです. NFS の場合, 作成, 読み込みとも 10 並列度程度で一旦 6000~7000 回/秒くらいのスループットに達しますが, それ以上並列度を上げると 3000 回/秒 ~4000 回程度に落ち着きます. Lustre もほぼ同じような曲線を描きます.

大雑把なまとめとして, Lustre の open/close にかかる時間は,

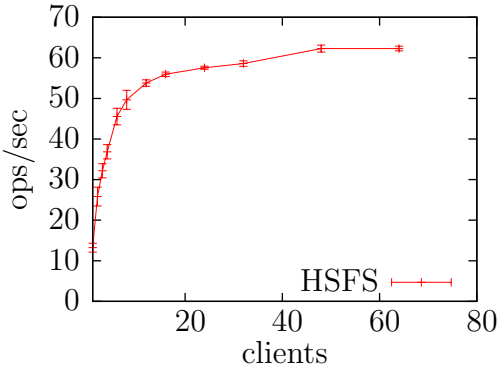
- 一回毎にかかる時間 (遅延) は NFS より大きい (2 倍程度) が,
- 並列度を増した際のスループットは NFS と同等

と考えることができます.

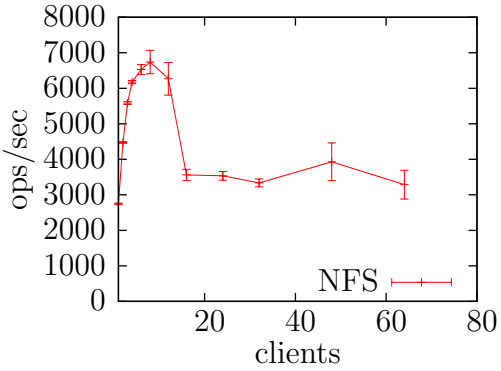
3 Lustre におけるストライピング (ファイル配置) の指定

3.1 基本

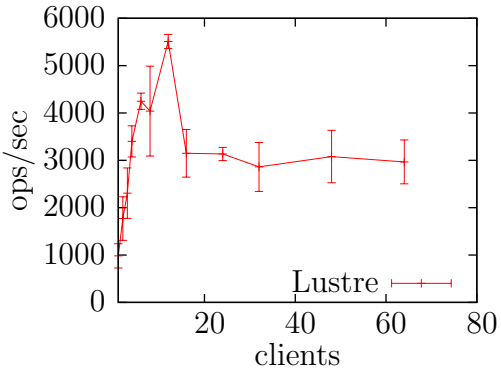
これから read/write 性能を紹介しますが, その前にその性能に大きな影響を与える, ファイルの配置 (サーバやディスクへのデータ分散の仕方) と, その指定方法を紹介します.



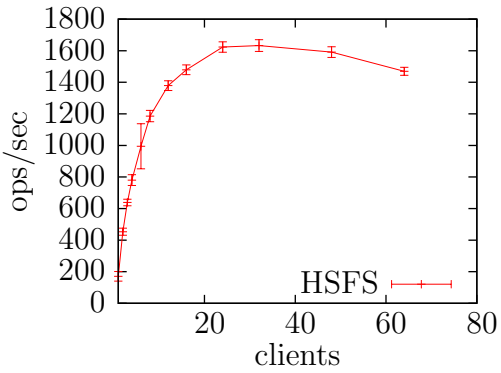
HSFS 作成



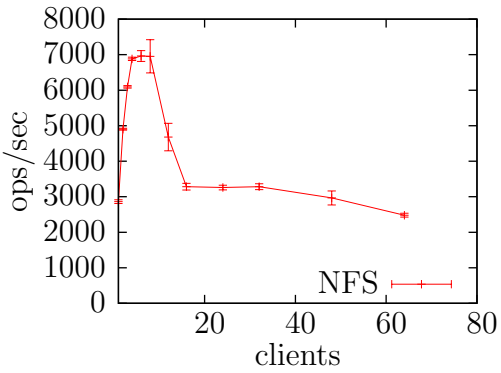
NFS 作成



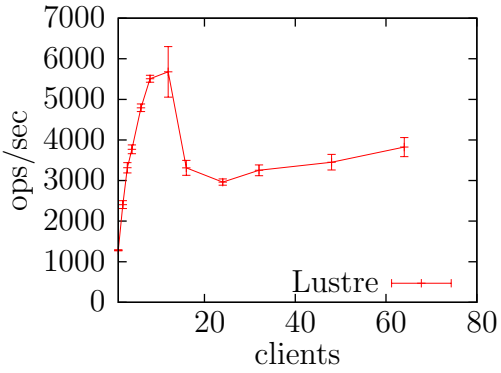
Lustre 作成



HSFS 読み込み



NFS 読み込み



Lustre 読み込み

図 1: 作成, 読み込みのための open/close のスループット

表 3: HA8000 の Lustre ファイルシステムの諸元

ディスクドライブ	1TB
1 OST あたりのディスクドライブ数	8D + 2P (RAID 6)
1 OSS あたりの OST 数	5
OSS 数	6
合計ディスクドライブ数	240D + 60P (240TB)
合計 OST 数	30

Lustre は並列ファイルシステムで, 一つの IO サーバの中の複数のディスクデバイスや, 複数の IO サーバを論理的に一つに束ね, 一つのファイルシステムを構成しつつスループットを高めています. ちょうど, RAID が複数のディスクドライブを一つにまとめて, 性能の高いディスクデバイスを構成しているのと同じ事を, その上のレベルでも行っているわけです.

Lustre の用語では, 一つ一つの IO サーバを **Object Storage Server (OSS)** と呼びます. 一つの OSS が複数のディスクデバイスを持ち, 個々のディスクデバイスを **Object Storage Target (OST)** と呼びます. 個々のディスクデバイス自体が, 物理的には複数のディスクドライブを束ねた RAID である場合もあります. IO スループットは多くの場合, ディスクの性能に律速されるので, このようにしてドライブを束ねて OST を構成し, OST を束ねて OSS を構成し, OSS を束ねて並列ファイルシステム全体を構成します.

HA8000 の構成では, 一つの OST は 1TB のドライブを 8 つ (+RAID 6 のパリティ用ドライブ 2 つ) 用いた RAID 6 ボリュームになっています. OSS につき 5 つの OST を管理し, OSS の台数は 6 台です. したがって合計で $6 \times 5 = 30$ の OST が存在します. 30 の OST は 1-30 までの番号がつけられており, OSS に代わりばんこに割り当てられて (ラウンドロビン分散して) います.

Lustre ファイルシステムにおいて指定できる, ファイルの「配置」とは, ファイルを OST 間にどのように分割して置くか, ということを意味しています. 複数の OST に分散させれば結果として複数のサーバへ分散している事にもなります. 何も設定しなければ一つのファイルは一つの OST に置かれ, 負荷分散はファイル単位で行われることになります. 一つのファイルを複数の OST にまたがって配置することを**ストライピング**と言います. 具体的には,

- どの OST を先頭にするか (オフセット o)
- いくつの OST にまたがってストライピングを行うか (ストライピング幅, ストライプ数 c)
- 何バイトを単位としてストライピングを行うか (ストライピングサイズ s)

が指定できます. 先頭 (オフセット) は指定しなければランダムになります. 通常はこれが無難と言えるでしょう.

lfs コマンドを用いることにより, 上記の 3 つのパラメータをファイルごとに指定することができます. 詳しい説明は, Lustre のマニュアルや lfs コマンドの man ページを参照して下さい. 以下では以下で必要な設定のみを説明します.

例えば,

```
% lfs setstripe -o 5 -c 3 -s 64k data
```

では, data というファイルが作られ, それに対しては OST 番号が 5, 6, 7 の 3 つの OST を用います. それらにファイルの先頭から 64KB ごとにデータを割り当てます. つまり最初の 64KB が OST 5, 次の 64KB が OST 6, 次の 64KB が OST 7, その次の 64KB は再び OST 5, という具合です. OST は OSS に対して代わりばんこに割り当てられているので, これで結果的にそれらのデータは 3 台の OSS に分散している事にもなります.

あるファイルの配置を調べるには, lfs getstripe コマンドを用います.

```
% lfs getstripe data
OBDS:
0: lustre-OST0000_UUID ACTIVE
1: lustre-OST0001_UUID ACTIVE

   ...

28: lustre-OST001c_UUID ACTIVE
29: lustre-OST001d_UUID ACTIVE
data
obdidx  objid objid  group
     5        2630474      0x28234a             0
     6        2628616      0x281c08             0
     7        2628580      0x281be4             0
```

ディレクトリに対して lfs setstripe を発行する事も出来, この場合, そのディレクトリの下に作られるファイルのデフォルトの配置を指定する効果を持ちます.

3.2 HA8000 で期待できる性能

表 3 に示した諸元により, 最大で期待できる性能について, おおよそ以下のような見通しが得られます.

- ストライピングをしなければ一つのファイルに対するアクセス性能は, ディスクデバイス一つ (RAID 6) の性能で決まり, これは 1 クライアントでの実測によりおよそ 100MB/sec 程度. 実際には多数のクライアントが一つのディスクデバイスへアクセスすると最大で 200MB/sec 程度まで伸びますが, 通常簡単に観測できるのは 100MB/sec 程度です.

- 一つの OSS は 5 つの OST を持つため, 一つの OSS の持つディスクに対する読み書き性能は 100MB/sec $\times 5 = 500$MB/sec 程度. これはネットワークの性能 (10Gbps $\approx$ 1.2GB/sec) よりも下回ります. つまりサーバ上にデータがまったくキャッシュされない状態でのスループットは, ディスクがボトルネックになり, 1 OSS 当たり 500MB/sec 程度になります. 繰り返しますが, 非常に並列度を増すとこれ以上になることがありえますが, 通常は見られません.

- OSS のキャッシュ上にあるデータに対してのスループットは, ネットワークの性能 (10Gbps $\approx$ 1GB/sec) に律速され, 1 OSS あたり 1.2GB/sec 程度.

- 6 台の OSS があるため, ファイルシステム全体でのスループットは, すべてがディスクに対するアクセスの場合 3GB/sec 程度, キャッシュ上のデータに対するアクセスの場合 6GB/sec 程度.

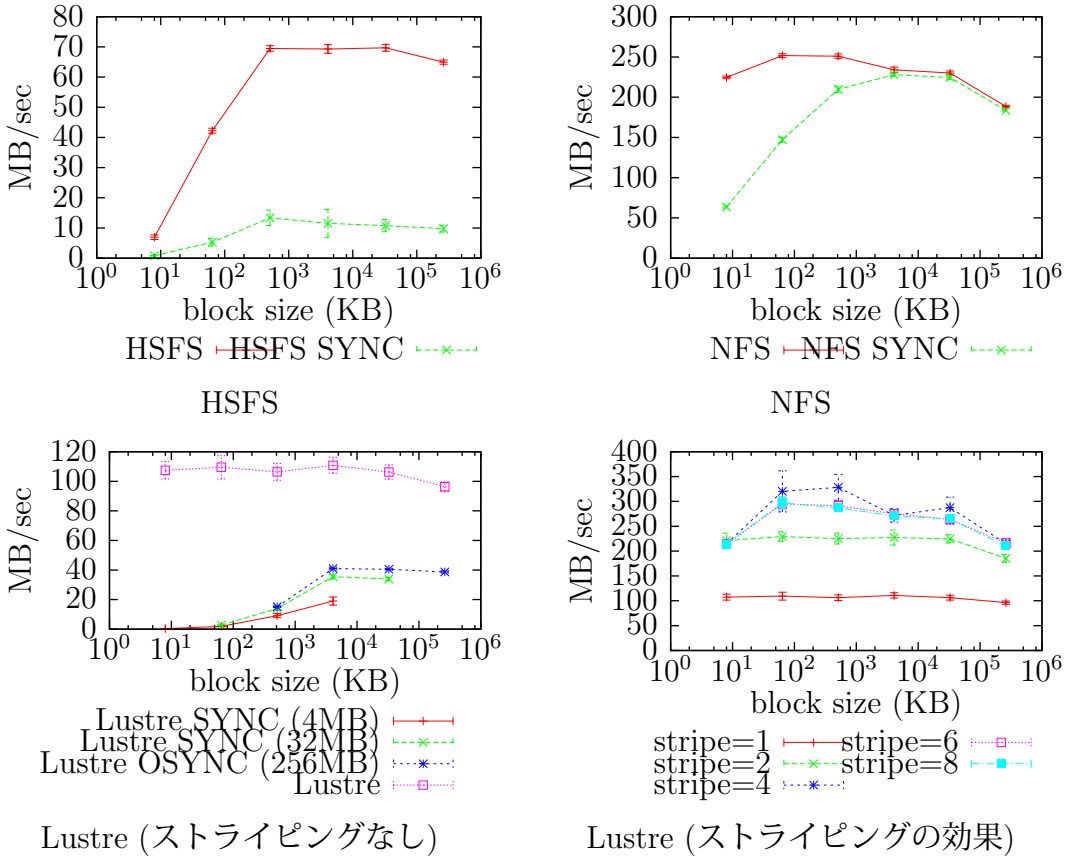
4　書き込み性能

4.1　1 クライアント

1 クライアントが一つのファイル (サイズ: 256MB) に書き込みを行った場合の性能を図 2 に示します. 一度の write システムコールで書き込む量を変えてそれを横軸 (block size) としています. また前節で述べたとおり, Lustre での性能はファイルをいくつのサーバに分散させるか (ストライピング) に依存しますので, それを変えて測定しています. 具体的には, 書き込み先のディレクトリに対して,

```
% lfs setstripe -c ストライプ数 ディレクトリ
```

として, ストライプ数 (のみ) を調節しています.



HSFS　NFS

Lustre (ストライピングなし)　Lustre (ストライピングの効果)

図 2: 1 クライアントによる書き込み

図 2 左下のグラフは Lustre でストライピングを 1 にして, 同期書き込み (open 時に O_SYNC フラグを指定) と通常の書き込みを比較しています. Lustre では書き込みの単位を小さくした同期書き込みが非常に遅いという事が分かります. あまり使うことはないかもしれませんが注意が必要です. 通常の書き込みでは小さなブロックサイズ (8KB) からほぼ一定のスループットが得られます.

右下のグラフは通常の書き込みで, ストライピングを 1, 2, 3, 4, 6, 8 と変えたときのスループットです. Lustre では, ストライピングを行わないと約 100MB/sec 程度. ストライプ数が 3-4 程度までは大体それに比例した性能が得られますが, その後は 300-350 MB/sec 程度で一定になります.

以降はファイルのストライプ数は 4 にして測定をします.

4.2 writeシステムコール以外の書き込み

図4に, Cのストリームライブラリ (fopen/fprintf) と, C++の標準ライブラリでのread/write性能を示します. 前記事にも書きましたが, 性能は,

- 書式付き入出力 (浮動小数点数や整数の表示) のオーバーヘッドの大きさ,
- ライブラリがどのようなサイズでread/writeシステムコールを発行しているか,

に左右されます. 要約としては,

- 用いた言語, ファイルシステムを問わず, doubleの書式付き入出力のオーバーヘッドが大きい. HSFSでg++を, 明示的なバッファサイズ指定なしで使うという場合を例外として, 基本的にはIO性能よりも書式をつける処理が律速になる.
- 通常の文字列のみの出力はLustreとNFSがほぼ同じ性能. HSFSでg++を用いる場合, 明示的なバッファサイズを指定する事が推奨される (その方法は前記事で述べた).

という事になります.

表 4: gcc/g++のライブラリでの書き込み性能

処理系	バッファサイズ	NFS	HSFS	Lustre
gcc	default	136.93	43.73	147.32
gcc	512k	138.53	59.83	155.30
g++	default	131.13	4.00	126.04
g++	512k	132.82	42.96	136.46

書式なし文字列 (`"0123456789abcde\n"`)

処理系	バッファサイズ	NFS	HSFS	Lustre
gcc	default	17.78	11.44	17.88
gcc	512k	17.55	13.73	17.97
g++	default	10.07	2.73	9.81
g++	512k	10.09	8.70	10.31

文字列と書式付きdouble

4.3 多クライアント

図2にクライアント数を増やした際の書き込みスループット (全クライアントの合計) を示します. 書き込みの単位は512KBで, Lustreのストライプ数は4としています. ファイルサイズはHSFS, NFSでは256MB, Lustreでは4GBとしています. Lustreでファイルサイズを256MBとすると下記よりも数倍高い性能になりますが, それはおそらく書き込みがクライアントにキャッシュされている間に, 書き込みが終了しているためと考えられます.

グラフから明らかなように, 並列ファイルシステムの効用が現れていると言えます. NFSはクライアント数が増加しても, サーバが一台にディスクが一ボリュームしかないため, すぐに性能が頭打ちになりますが, Lustreでは24クライアントほどになると2.5GB/sec程度まで向上しています. この並列度で

は, NFS の 10 倍程度出ることになります. 4GB のファイルを多数のクライアントが書き込んでいることから, 基本的にはディスクへの書き込み速度を測っていることになります. それは 3.2 節で述べたとおり, 期待できる性能はおよそ 3GB/sec 程度です.

HSFS も並列ファイルシステムですが, クライアントを増やしても性能が上がらない理由の一つは, クライアント 16 ノード毎に, ファイルサーバーへのアクセスパス (ネットワーク) にボトルネックが存在している事にあります. 詳しくは前記事を参照してください.

5 読み込み性能

5.1 1 クライアント

図 4 に 1 クライアントによる読み込み性能 (横軸は read システムコールを発行する単位) を示します. 1 クライアントによる読み込みで, Lustre が NFS と大体同等かやや上回る性能 (500-600MB/sec) を示している事が分かります.

5.2 多クライアント

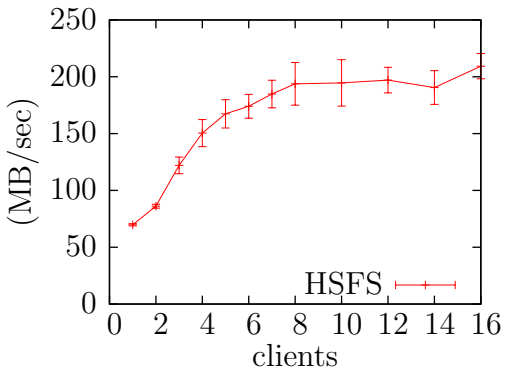
図 2 にクライアント数を増やした際の読み込みスループット (全クライアントの合計) を示します. 書き込みの際と同様, システムコール発行の単位は 512KB で, Lustre のストライプ数は 4 としています. ファイルサイズも書き込み時と同様, HSFS, NFS は 256MB, Lustre は 4GB としています.

ここでも, 並列ファイルシステムの効用が現れていると言えます. NFS が飽和している地点の数字 (約 1.2GB/sec) は, ファイルがサーバのキャッシュに残っている事で, ディスクではなくサーバのネットワーク (10Gbps) が見えているものです. 並列度が大きい場合, Lustre の性能は少なくとも 4GB 程度, つまり NFS の 4 倍まではスケールしていることが分かります. ファイルがサーバのキャッシュにないときはこの差はより大きくなります. HSFS がクライアント数を増やしても伸びない理由は 4.3 節と同様です.

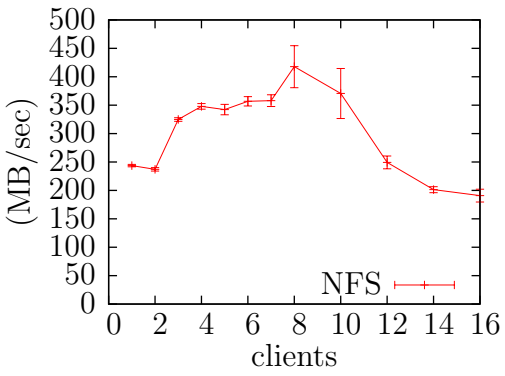
6 まとめ

HA8000 上のファイルシステム, 特に Lustre に関して性能を理解するための整理と, 実験を通した確認を行いました.

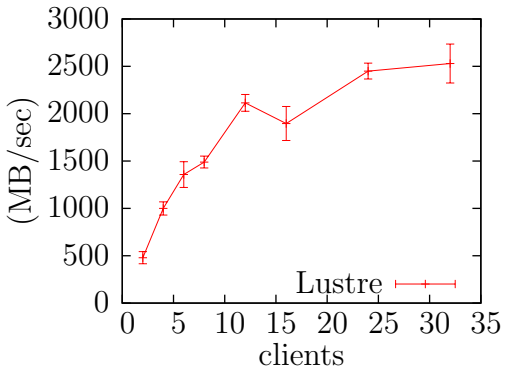
- Lustre の open/close 性能は, 1 クライアント時に NFS の半分程度. つまり, 一つ一つの操作は NFS の倍程度の遅延が生じる.
- しかし, クライアント数が増えたときに可能な open/close 回数 (スループット) は NFS と同程度.
- ストライピングをしなければ 1 クライアイントが一つのファイルに対してアクセスする際の read/write 性能は 100MB/sec 程度. lfs コマンドを用いるとストライピングが可能. ストライピングを行えば 1 ファイルで 300-350MB/sec 程度までは向上する.
- ストライピングを無闇に大きくしても多くの場合あまり意味がない. 300-350MB/sec の性能を得るにはストライプ数 4 程度で十分. これ以上増やしてもあまり意味はない.



HSFS

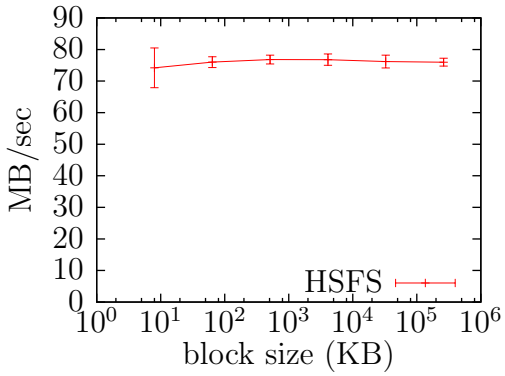


NFS

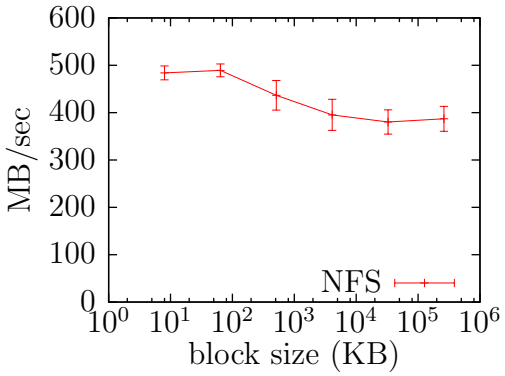


Lustre

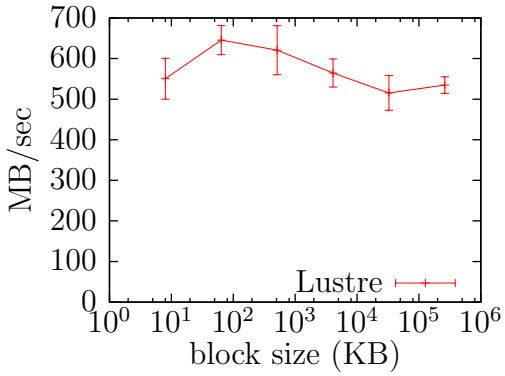
図 3: 多クライアントによる書き込み



HSFS

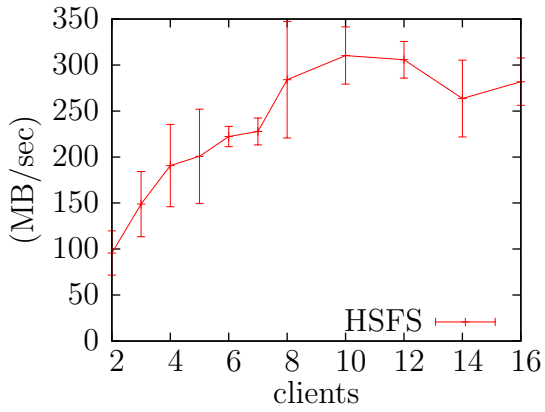


NFS

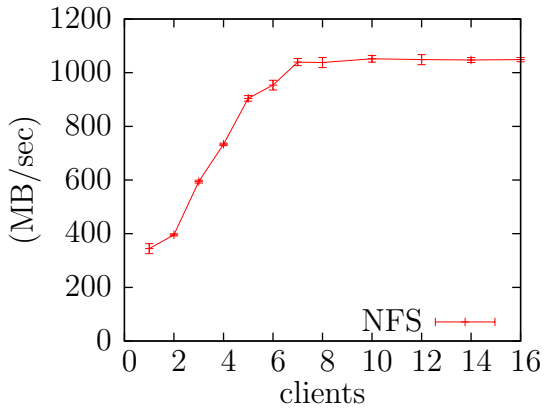


Lustre

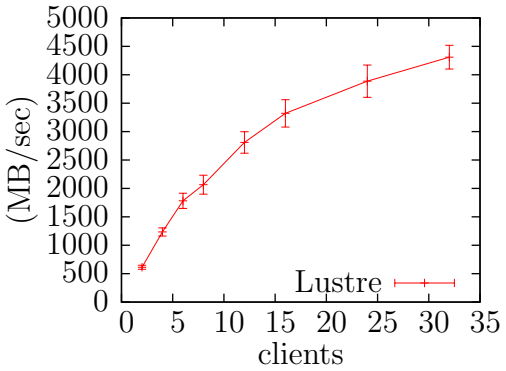
図 4: 1 クライアントによる読み込み



HSFS



NFS



Lustre

図 5: 多クライアントによる読み込み

- 複数クライアントによる write の性能は, 並列ファイルシステムの効用が現れており, クライアント数を増やすだけで 2.5GB/sec 程度が確認された. これは NFS の 10 倍程度.
- 複数クライアントによる read の性能は 4GB/sec 程度が 32 クライアントで確認された. NFS (データがサーバのキャッシュ上にある場合) の 4 倍程度. データがサーバのキャッシュ上になければ差はさらに拡大する.

本稿がファイルシステムの性能理解, ひいては HA8000 環境の有効・快適な利用の一助となれば幸いです.

参考文献

[1] Lustre. http://wiki.lustre.org/.

[2] paramark High Fidelity Parallel File System Benchmark. http://code.google.com/p/paramark/.

[3] 田浦健次朗. HA8000 のファイルシステムについて. In **スーパーコンピューティングニュース**, volume 12. 東京大学情報基盤センタースーパーコンピューティング部門, 1 月 2010 年. http://www.cc.u-tokyo.ac.jp/publication/news/.

[4] 東京大学情報基盤センタースーパーコンピューティング部門. HA8000 クラスタシステム FAQ. http://www.cc.u-tokyo.ac.jp/service/faq/ha8000_faq.html.