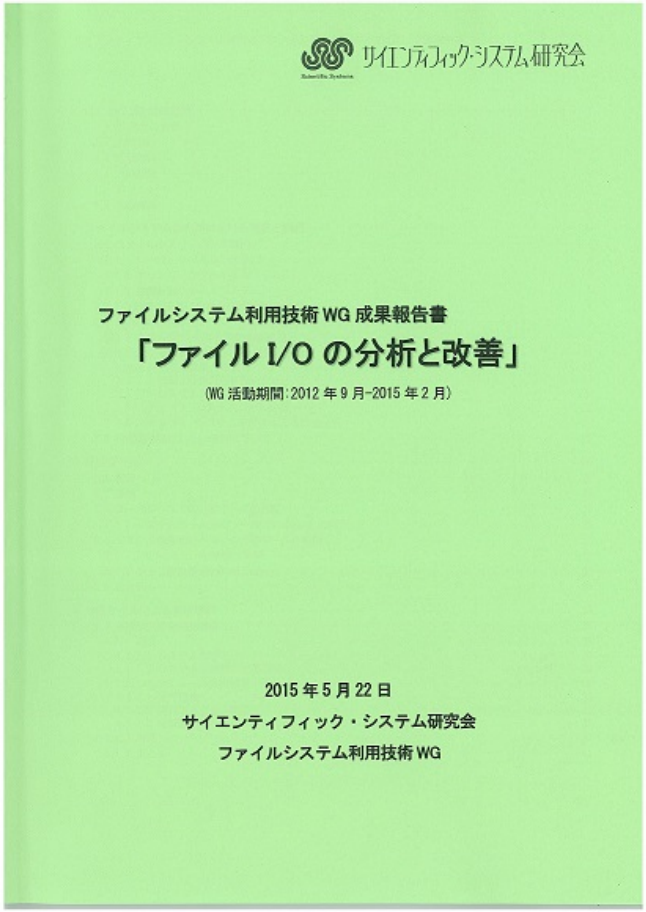
ーSS研ファイルシステム利用技術WG成果報告ー

ファイルI/Oの分析と改善

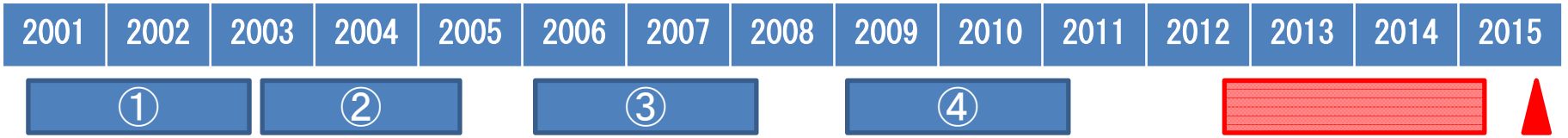


SS研 ファイルシステム利用技術WG まとめ役
国立研究開発法人 宇宙航空研究開発機構 藤田直行
fujita@chofu.jaxa.jp

Agenda

- はじめに
- ファイルシステムの利用における現状技術
- I/O効率改善のための4つのステップ
- 分析ツール
- 分析ツールによる解析事例
- おわりに

■SS研でのストレージに関するWG



①ネットワーク時代の統合ストレージマネージメントWG
(活動期間:2001年5月～2003年4月)

②ストレージを中心としたシステムマネージメントWG
(活動期間:2003年5月～2005年4月)

③データマネジメントを意識したストレージソリューションWG
(活動期間:2006年2月～2008年4月)

④大規模ストレージWG
(活動期間:2009年1月～2011年3月)

■ファイルシステム利用技術WG

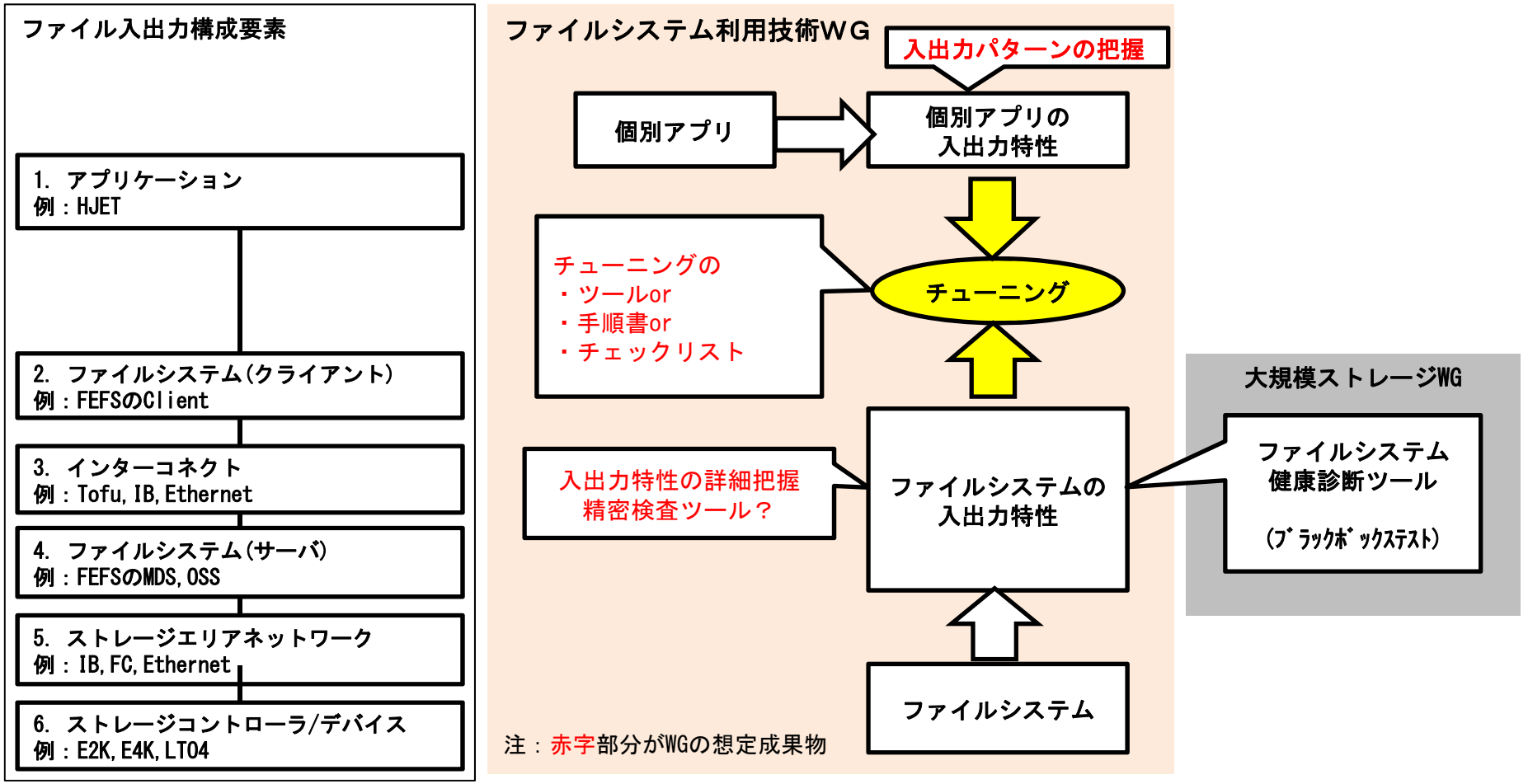
(活動期間:2012年9月～2015年2月)

■ファイルシステム利用技術WGメンバ

		氏名	機関/所属(2015年2月28日現在)
会員	担当幹事	松澤　照男 (2012年9月～2014年5月)	北陸先端科学技術大学院大学
		井口　　寧 (2014年5月～2015年2月)	北陸先端科学技術大学院大学
	推進委員	藤田　直行（まとめ役）	宇宙航空研究開発機構
		安岡　孝一	京都大学人文科学研究所附属東アジア人文情報学研究センター
		松本　好美	高エネルギー加速器研究機構計算科学センター
		八木　雅文	国立天文台
		早戸　良成	東京大学宇宙線研究所附属神岡宇宙素粒子研究施設
		黒川　原佳	理化学研究所
賛助会員（富士通）	推進委員	栄元　雅彦（まとめ役）	富士通(株) テクニカルコンピューティング・ソリューション事業本部
		甲斐　俊彦	富士通(株) 次世代テクニカルコンピューティング開発本部
		酒井憲一郎	富士通(株) 次世代テクニカルコンピューティング開発本部
		宮本　巧輝	富士通(株) 次世代テクニカルコンピューティング開発本部
	オブザーバ	阿部　孝之	富士通(株) テクニカルコンピューティング・ソリューション事業本部
		坂口　吉生	富士通(株) テクニカルコンピューティング・ソリューション事業本部
		塚原　知宏	富士通(株) テクニカルコンピューティング・ソリューション事業本部

■WG発足に際して

ファイルシステム利用技術WGの活動内容イメージ

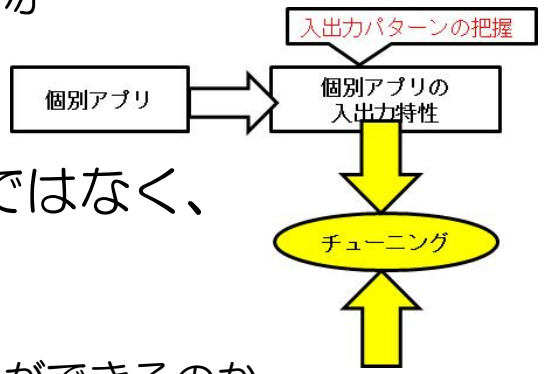


■WGを推進するための意識共有

■WGのポジショニング

- ✓ 利用者視点のツールと管理者視点のツールは別の物
  - 利用者視点のツールは「ソースコードの分析」
  - 管理者視点のツールは「プロファイラ」
- ✓ スライド5の「入出力パターン」を共通化するのではなく、
  - 個別のアプリケーションの特徴を把握して、
  - それが、システムと合っているのか否か、
  - 合っていない場合、どこをチューニングすれば合わせることができるのか

  を、考える



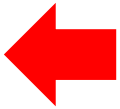
- ✓ 上記ノウハウを、「ツール化」する。できなければ、「手順書化」や「チェックリスト化」

■実現のための施策

- ✓ 個別アプリケーションの入出力特性【利用者が把握すべきもの】
  - 「アプリケーション入出力特性測定ツール(利用者側視点)」をつくる
- ✓ ファイルシステムの入出力特性【管理者が設計・把握するもの】
  - 大規模ストレージWGで作ったファイルシステム健康診断ツールを活用

Agenda

- はじめに
- ファイルシステムの利用における現状技術
- I/O効率改善のための4つのステップ
- 分析ツール
- 分析ツールによる解析事例
- おわりに



ファイルシステムの利用における現状技術

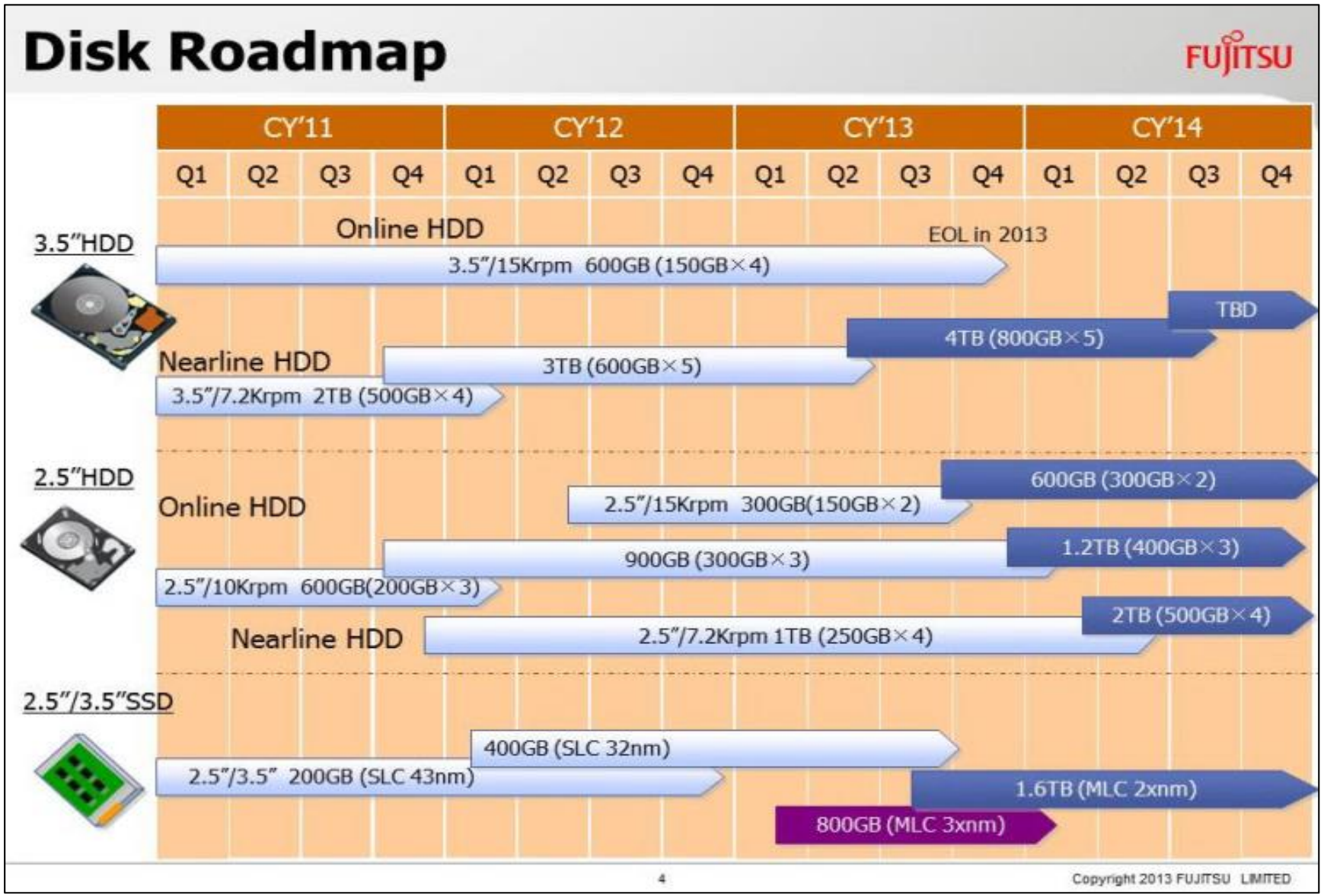
- ストレージ技術
  - ハードディスク動向
  - テープストレージ動向
  - インターフェース動向

  報告書 付録 資料１
- I/Oモニタリング技術
  - ジョブ統計情報
  - I/Oアカウンティング情報
  - ファイルシステム統計情報
  - OS統計情報

  報告書 付録 資料2
- ファイルシステム構成
  - ローカルファイルシステム
  - ネットワークファイルシステム
  - 並列分散ファイルシステム

  報告書 2.1節

■ハードディスク動向

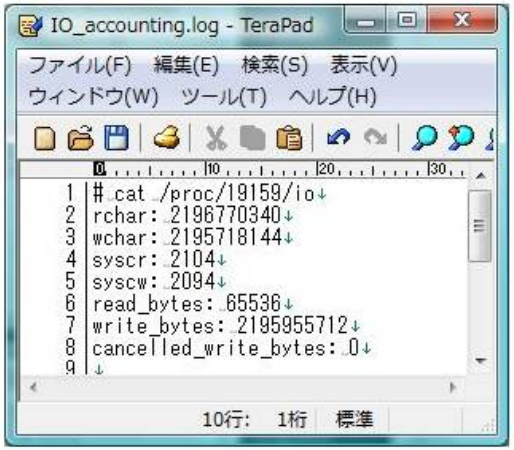

Disk Roadmap
FUJITSU
CY'11
CY'12
CY'13
CY'14
Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
3.5"HDD
Online HDD
EOL in 2013
3.5"/15Krpm 600GB (150GB×4)
TBD
4TB (800GB×5)
Nearline HDD
3TB (600GB×5)
3.5"/7.2Krpm 2TB (500GB×4)
2.5"HDD
Online HDD
600GB (300GB×2)
2.5"/15Krpm 300GB(150GB×2)
1.2TB (400GB×3)
900GB (300GB×3)
2.5"/10Krpm 600GB(200GB×3)
2TB (500GB×4)
Nearline HDD
2.5"/7.2Krpm 1TB (250GB×4)
2.5"/3.5"SSD
400GB (SLC 32nm)
2.5"/3.5" 200GB (SLC 43nm)
1.6TB (MLC 2xnm)
800GB (MLC 3xnm)
4
Copyright 2013 FUJITSU LIMITED

■I/Oアカウンティング情報

IO Accounting 情報

FUJITSU

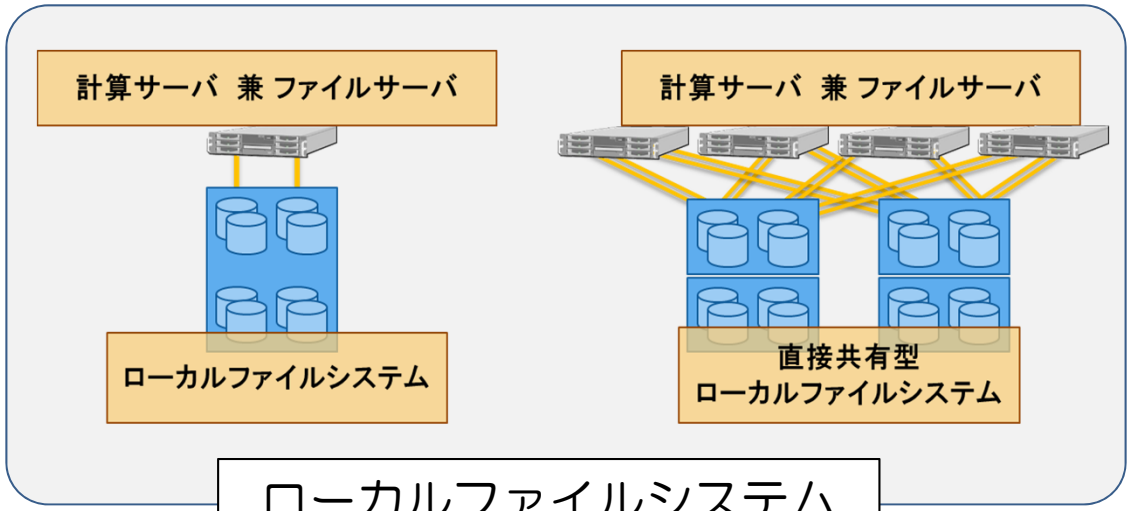
- 動作中プロセス単位のread/write量が採取可能
  - rchar,wchar：read,writeシステムコールの実行バイト数
  - syscr,syscw：read,writeシステムコールの実行回数
  - read_bytes：ブロックデイバスからページキャッシュにロードされたbyte数
  - write_bytes：ページキャッシュ上でdirtyフラグが立ったbyte数
  - cancelled_write_bytes：デバイスに書き込まれる前にcancelされたbyte数
- pidstat -dやdstatが便利



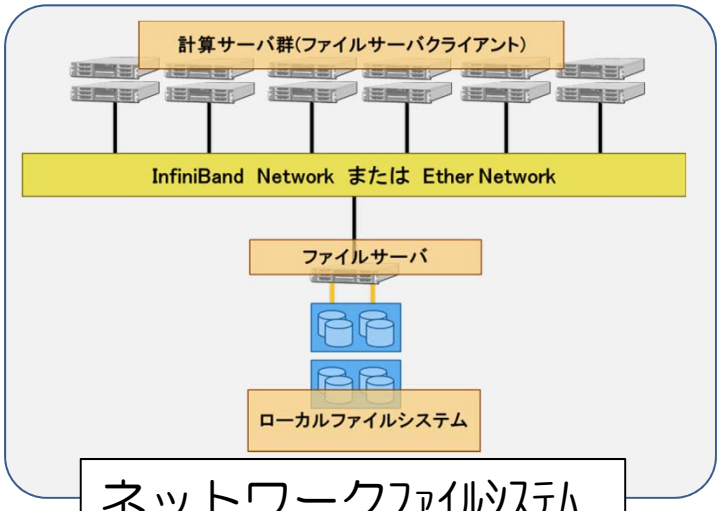
5

Copyright 2013 FUJITSU LIMITED

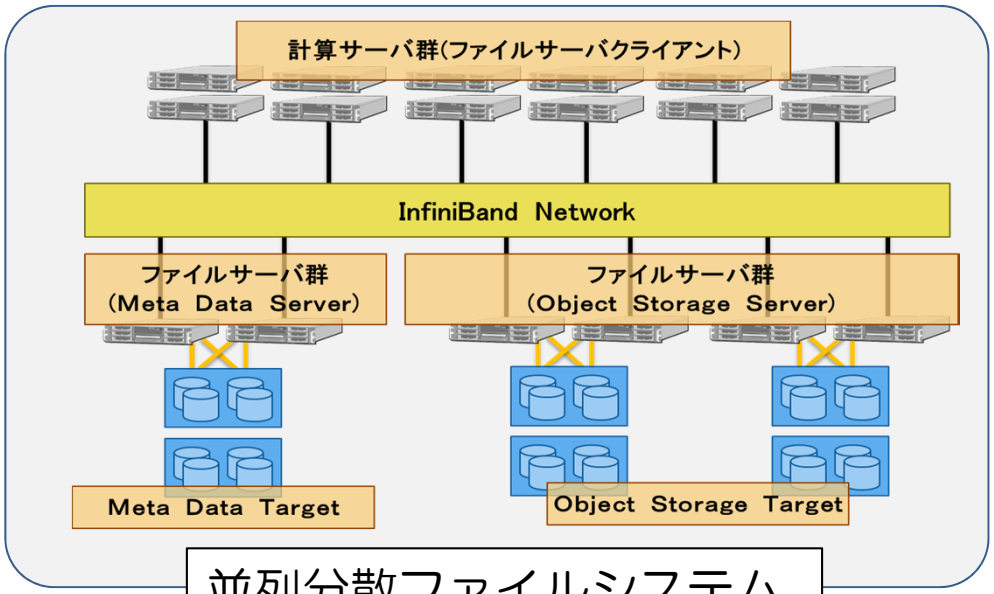
■ファイルシステム構成



ローカルファイルシステム



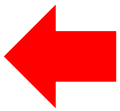
ネットワークファイルシステム



並列分散ファイルシステム

Agenda

- はじめに
- ファイルシステムの利用における現状技術
- I/O効率改善のための4つのステップ
- 分析ツール
- 分析ツールによる解析事例
- おわりに



■I/O効率改善のための４つのステップ

(ステップ1)…I/O比率の簡易確認

(ステップ2)…以下の項目確認

(2-1) 想定I/O量

(2-2) 想定I/O性能

(2-3) システム負荷

(ステップ3)…システムコール分析

(ステップ4)…システム管理者による詳細分析

■報告書の読み方(1/2)

- ✓ 各ステップでは、以下の項目を順番に説明している(報告書P.7に関連の記載あり)。
  - 【作業実施者】
  - 【実施項目】
  - 【確認のポイント】
  - 【手段や手法】
  - 【管理者提供の情報】
- ✓ このうち、【手段や手法】は、
  - 「ジョブ」
  - 「実行方法」
  - 「ファイルシステム」

  の種類によって、そのやり方に違いがあるため、代表的と思われる組み合わせを選んで、具体的な手段や手法を複数記載してある。

■報告書の読み方(2/2)

【報告書】表2.1 想定対象

項目	種類
ジョブ	逐次ジョブ 並列ジョブ
実行方法	インタラクティブ実行 バッチ実行
ファイルシステム	ローカルファイルシステム(L-FS) ネットワークファイルシステム(N-FS) 並列分散ファイルシステム(P-FS)

報告書での記載例：

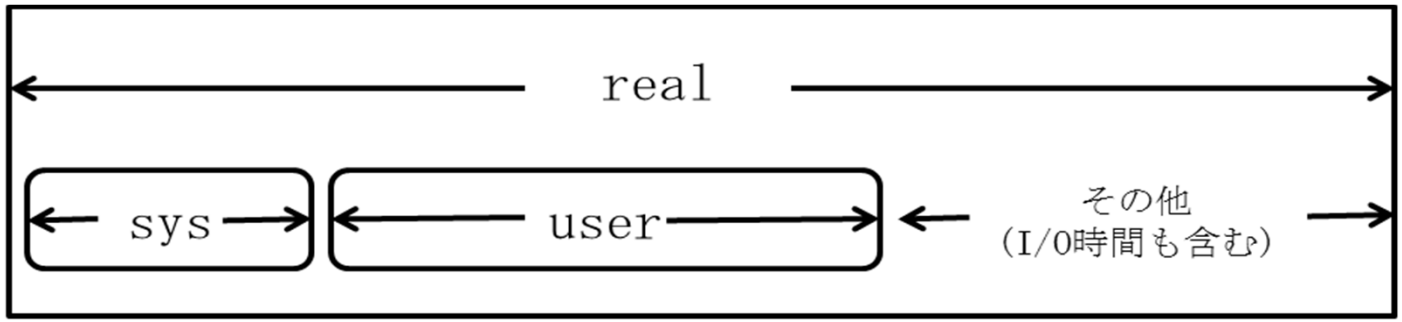
例1)「逐次ジョブ/インタラクティブ実行/L-FS,N-FS,P-FS」
ファイルシステムの種類には依存せず、
逐次ジョブをインタラクティブに実行した場合の
データ採取の手順が紹介されている。

例2)「逐次ジョブ,並列ジョブ/バッチ実行/P-FS」
ジョブが逐次ジョブか並列ジョブかには依存せず、
並列分散ファイルの計算機システム上でバッチでジョブ実行した場合の
データ採取の手順が紹介されている。

■ステップ１…I/O比率の簡易確認

作業実施者	ユーザ
実施項目	CPU利用率の測定
確認のポイント	CPU利用率が目安値以上である
手段や手法の例	timeコマンドからuser÷realを計算
管理者提供の情報	目安値

項目	種類
ジョブ	逐次ジョブ
実行方法	インタラクティブ実行
ファイルシステム	ローカルファイルシステム(L-FS)



real　：　ジョブの開始から終了までの時間(実行時間)
user　：　プログラム自体の処理時間(CPU時間)
sys　：　OS処理時間
その他　：　I/O時間など

【報告書】図2.5 実行時間の内訳

■ステップ１…I/O比率の簡易確認例

作業実施者	ユーザ
実施項目	CPU利用率の測定
確認のポイント	CPU利用率が目安値以上である
手段や手法の例	timeコマンドからuser÷realを計算
管理者提供の情報	目安値 ⇨ 0.9以上

項目	種類
ジョブ	逐次ジョブ
実行方法	インタラクティブ実行
ファイルシステム	ローカルファイルシステム(L-FS)

```
% /usr/bin/time -f " real %E ¥n user %U ¥n sys %S ¥n" ./IOR
-w -r -b 1g -F -e -s 32 -i 2
    省略
Max Write: 138.34 MiB/sec (145.06 MB/sec)
Max Read:  264.51 MiB/sec (277.36 MB/sec)

Run finished: Tue Jan 20 10:34:43 2015
real 26:27.30
user 0.76
sys 180.39
%
```

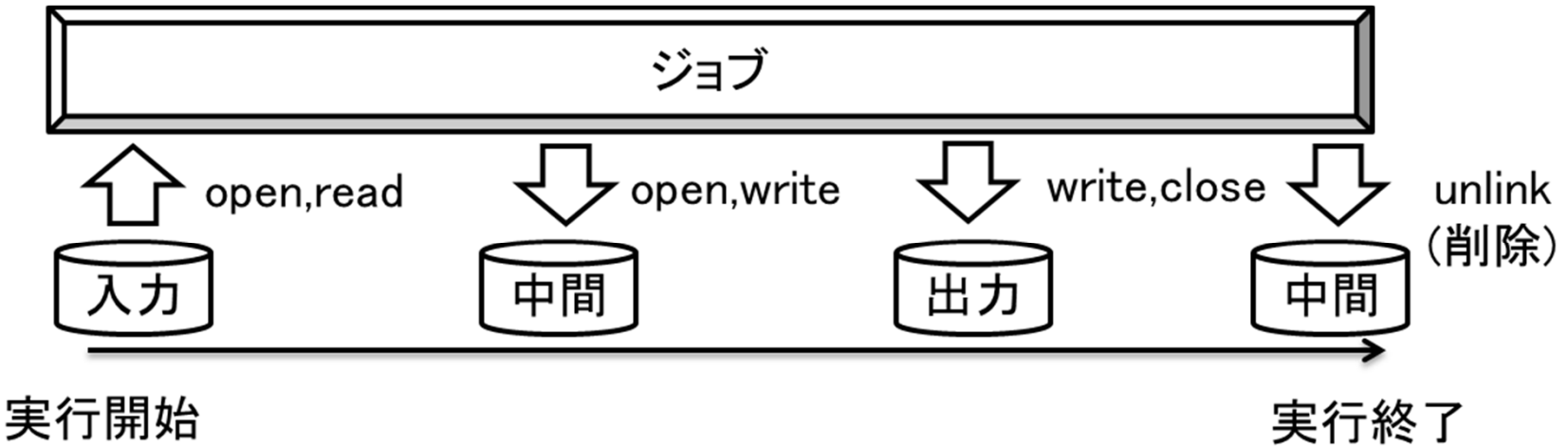
⇨ user 0.76秒 ÷ real 26:27.30 = 0.000478

【報告書】図2.6 timeコマンドにおける実行し時間の内訳の表示方法

■ステップ2-1…想定I/O量の確認

作業実施者	ユーザ
実施項目	I/O量の予測と、実I/O量との比較
確認のポイント	予測したI/O量とほぼ等しい実I/O量である
手段や手法の例	timeコマンドで総I/O量を取得
管理者提供の情報	(ファイルシステムマウント情報、設定パラメータ)

項目	種類
ジョブ	逐次ジョブ
実行方法	インタラクティブ実行
ファイルシステム	ローカルファイルシステム(L-FS)



【報告書】図2.10 ジョブ実行時のI/O例

■ステップ2-1…想定I/O量の確認例

作業実施者	ユーザ
実施項目	I/O量の予測と、実I/O量との比較
確認のポイント	予測したI/O量とほぼ等しい実I/O量である
手段や手法の例	timeコマンドで総I/O量を取得
管理者提供の情報	(ファイルシステムマウント情報、設定パラメータ)

項目	種類
ジョブ	逐次ジョブ
実行方法	インタラクティブ実行
ファイルシステム	ローカルファイルシステム(L-FS)

```
local-single% /usr/bin/time -f "real %E ¥nuser %U ¥nsys %S
¥nio %Okb¥n" ./IOR  -w -r -b 1g -F -e -s 64 -i 2
    省略
Max Write: 138.34 MiB/sec (145.06 MB/sec)
Max Read:  264.51 MiB/sec (277.36 MB/sec)

Run finished: Tue Jan 20 10:34:43 2015real 0:57.72
real 26:27.30
user 0.76
sys 180.39
io 278734376kb
local-single%
```

256GB

【報告書】図2.11　L-FS(ext4)における逐次ジョブ実行時のI/O量

■ステップ2-2･･･想定I/O性能の確認

作業実施者	ユーザ
実施項目	I/O性能の予測と、実I/O性能との比較
確認のポイント	予測したI/O性能とほぼ等しい実I/O性能である
手段や手法の例	iotopコマンドでI/O性能を観測
管理者提供の情報	健康診断ツール等による典型的なI/O性能値

項目	種類
ジョブ	並列ジョブ
実行方法	インタラクティブ実行
ファイルシステム	ネットワークファイルシステム(N-FS)

■ステップ2-2…想定I/O性能の確認例

作業実施者	ユーザ
実施項目	I/O性能の予測と、実I/O性能との比較
確認のポイント	予測したI/O性能とほぼ等しい実I/O性能である
手段や手法の例	iotopコマンドでI/O性能を観測
管理者提供の情報	健康診断ツール等による典型的なI/O性能値 ⇨ 数MB/s以上

項目	種類
ジョブ	並列ジョブ
実行方法	インタラクティブ実行
ファイルシステム	ネットワークファイルシステム(N-FS)

```
nfs-para% /usr/bin/iotop -d 30
Total DISK READ: 0.00 B/s | Total DISK WRITE: 4.66 K/s
  TID  PRIO  USER     DISK READ  DISK WRITE  SWAPIN     IO>    COMMAND
   省略
  TID  PRIO  USER     DISK READ  DISK WRITE  SWAPIN     IO>    COMMAND
19083 be/4 abebe       0.00 B/s  189.26 M/s  0.00 % 85.58 % ./IOR -w~TEST/test
19084 be/4 abebe       0.00 B/s  205.29 M/s  0.00 % 84.67 % ./IOR -w~TEST/test
   省略
Total DISK READ: 11.84 K/s | Total DISK WRITE: 6.65 K/s
  TID  PRIO  USER     DISK READ  DISK WRITE  SWAPIN     IO>    COMMAND
19084 be/4 abebe      54.04 M/s    0.00 B/s  0.00 % 96.27 % ./IOR -w~TEST/test
19083 be/4 abebe      47.93 M/s    0.00 B/s  0.00 % 89.42 % ./IOR -w~TEST/test
   省略
```

【報告書】図2.23　N-FS(NFS)におけるI/O性能の表示の例

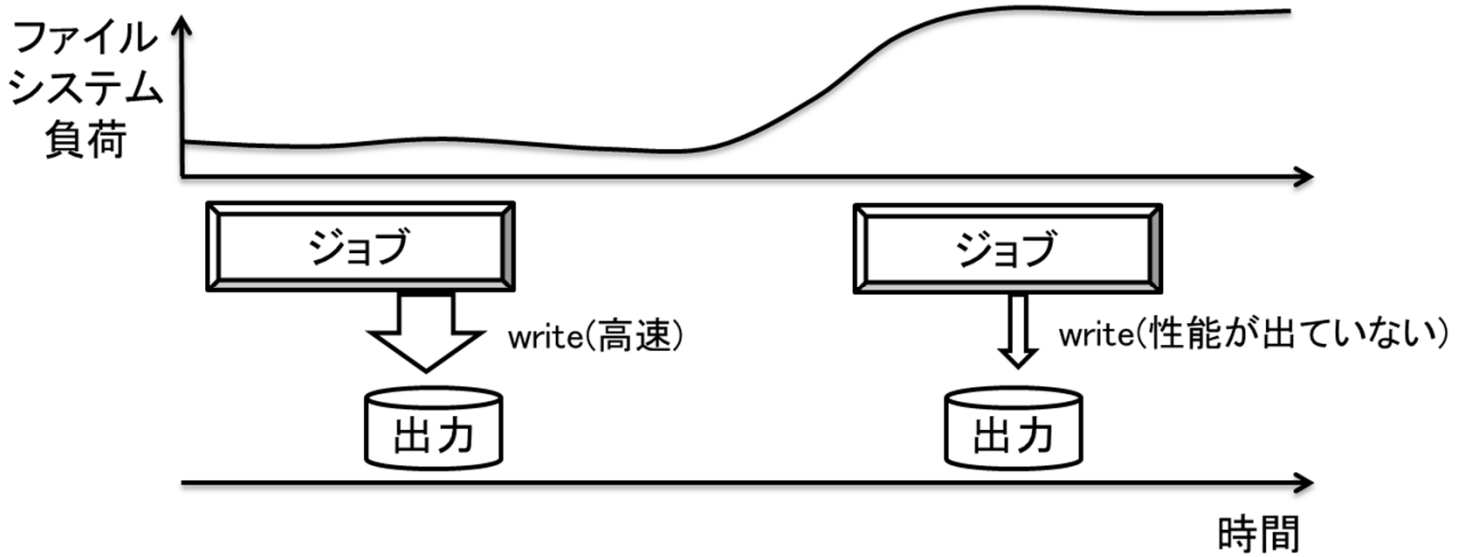
```
nfs% mpiexec -n 2  --hostfile ./hostfile  ./IOR  -w -r -b 1g -F -e -s 32 -i 2
```

【報告書】図2.24　N-FS(NFS)における並列ジョブ実行の例

■ステップ2-3…システム負荷の確認

作業実施者	ユーザ
実施項目	I/O時のシステム状態(負荷、縮退等)の確認
確認のポイント	ファイルシステムが正常である
手段や手法の例	dateコマンドによる時刻取得
管理者提供の情報	ファイルシステム負荷状況・状態の時系列データ

項目	種類
ジョブ	並列ジョブ
実行方法	インタラクティブ実行
ファイルシステム	ネットワークファイルシステム(N-FS)



【報告書】図2.27出力時にファイルシステムの能力が十分に発揮できない状態の例

■ステップ2-3…システム負荷の確認例

作業実施者	ユーザ
実施項目	I/O時のシステム状態(負荷、縮退等)の確認
確認のポイント	ファイルシステムが正常である
手段や手法の例	dateコマンドによる時刻取得
管理者提供の情報	ファイルシステム負荷状況・状態の時系列データ

項目	種類
ジョブ	並列ジョブ
実行方法	インタラクティブ実行
ファイルシステム	ネットワークファイルシステム(N-FS)

```
nfs% /bin/date; mpiexec –n 2  --hostfile ./hostfile   /IOR  -w -r -b 1g -F -e
-s 32 -i 2 ; /bin/date

2015年  1月 20日 火曜日 06:37:13 JST
   省略
Max Write: 110.31 MiB/sec (115.67 MB/sec)
Max Read:  111.45 MiB/sec (116.87 MB/sec)
   省略
2015年  1月 20日 火曜日 06:57:18 JST
```

【報告書】図2.28 dateコマンドを利用した時刻表示の例

■ステップ3…システムコール分析

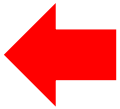
作業実施者	ユーザとシステム管理者の共同作業
実施項目	システムコール分析用コマンドを追加してのジョブ実行
確認のポイント	ｼｽﾃﾑｺｰﾙ発生タイミング、I/O時間、負荷状況等総合的に
手段や手法	IO-Dockを使用
管理者提供の情報	(分析方法・結果を公開し、ユーザのスキルアップに期待)

■ステップ4…システム管理者による詳細分析

作業実施者	システム管理者
実施項目	システム高負荷の原因究明、ジョブ実行時間の特異点分析等
確認のポイント	(同上)
手段や手法	各種ログ情報等の通常通常運用状態との差異を手がかりに調査
管理者提供の情報	ーーー

Agenda

- はじめに
- ファイルシステムの利用における現状技術
- I/O効率改善のための4つのステップ
- 分析ツール
- 分析ツールによる解析事例
- おわりに



IO-Dock(1/2)

1. 機能概要

- ✓ トレースログ(strace,truss)の分析ツール
- ✓ システムコール毎の実行回数、実行時間等を集計

2. 動作環境

- ✓ Red Hat Enterprise Linux 6 ないしは 5

3. 導入方法

1. cd (インストールしたいディレクトリ)
2. tar xfz <(本ツールtar.gzファイル) >/IO-Dock-1.0.0.0.tar.gz

■IO-Dock(2/2)

4. ツール使用の流れ

1. 中間ファイル生成
   (make_im_data.pl の実行)
2. システムコール毎の実行回数、実行時間などの集計
   (analyze_summary.pl の実行)
3. 着目したシステムコールの掘り下げた分析
   (analyze_summary.pl --func write)……”write”
   (analyze_io_dist.pl) ……IO長ヒストグラム
   (analyze_io_oer_time.pl) ……単位時間のIO量、IO回数
   (analyze_syscall_per_time.pl)……単位時間の
   システムコール実行回数

※詳細は、報告書 付録 資料3
『システムコール分析ツール「IO-Dock」使用手引書』
を、ご参照願います。

IO-Dock使用例1(システムコール毎の集計)

```
# ./analyze_summary.pl /work3/tmp/work-starace-20150216.tmpdata
```

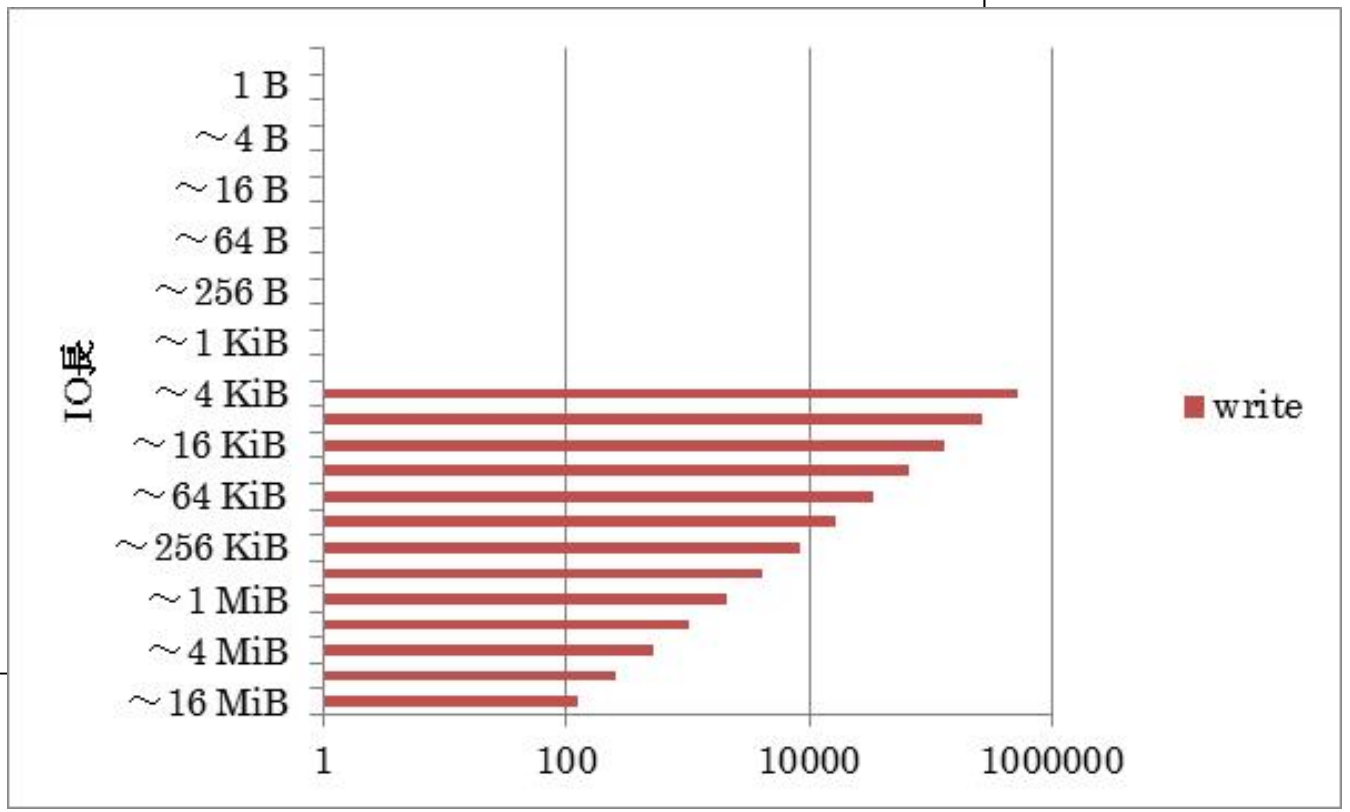
syscall	num	err	total_time[s]	avg_time[s]	min_time[s]	max_time[s]	total_size[B]	avg_size[B]	min_size[B]	max_size[B]
_llseek	293	0	0.010367	0.000035	0.000012	0.000140				
access	2	2	0.000058	0.000029	0.000028	0.000030				
brk	3	0	0.000078	0.000026	0.000025	0.000027				
close	1028	146	0.050469	0.000049	0.000011	0.000298				
execve	1	0	0.000420	0.000420	0.000420	0.000420				
exit_group	1	0	0.000000	0.000000	0.000000	0.000000				
fstat64	6	0	0.000151	0.000025	0.000021	0.000039				
fsync	1168	0	130.845662	0.112025	0.000137	2.595287				
ftruncate	146	0	0.006427	0.000044	0.000016	0.000178				
futex	2	1	0.000045	0.000022	0.000022	0.000023				
getrlimit	1	0	0.000023	0.000023	0.000023	0.000023				
gettimeofday	1191	0	0.038632	0.000032	0.000010	0.000146				
mmap2	15	0	0.000406	0.000027	0.000023	0.000037				
mprotect	4	0	0.000103	0.000026	0.000024	0.000027				
munmap	2	0	0.000063	0.000031	0.000030	0.000033				
open	882	0	0.047239	0.000054	0.000013	0.000400				
read	1048667	0	33.308163	0.000032	0.000009	0.006847	28319172434	27004	7	16777216
rt_sigaction	4	0	0.000089	0.000022	0.000021	0.000024				
rt_sigprocmask	1	0	0.000022	0.000022	0.000022	0.000022				
set_robust_list	1	0	0.000021	0.000021	0.000021	0.000021				
set_thread_area	1	0	0.000024	0.000024	0.000024	0.000024				
set_tid_address	1	0	0.000032	0.000032	0.000032	0.000032				
stat64	439	1	0.017096	0.000039	0.000012	0.000304				
sync	292	0	15.048415	0.051536	0.028979	0.181499				
time	2	0	0.000044	0.000022	0.000022	0.000022				
uname	2	0	0.000077	0.000039	0.000023	0.000054				
unlink	293	0	3.179225	0.010851	0.000029	0.122195				
write	1051582	0	58.256180	0.000055	0.000011	0.020730	27849816254	26483	1	16777216

■IO-Dock使用例2(writeシステムコールの内訳)

```
# ./analyze_summary.pl /work3/tmp/work-starace-20150216.tmpdata --func write
#fname                      num  err  total_time[s]  avg_time[s]  min_time[s]  max_time[s]  total_size[B]  avg_size[B]  min_size[B]  max_size[B]
"/work3/tmp/workfile"  1048370    0      58.150417     0.000055     0.000011     0.020730    27849785344        26564         4096     16777216
"STDOUT"                  1819    0       0.062489     0.000034     0.000012     0.000272          15874            8            1           74
"test.xls"                1393    0       0.043274     0.000031     0.000016     0.000129          15036           10            4          268
```

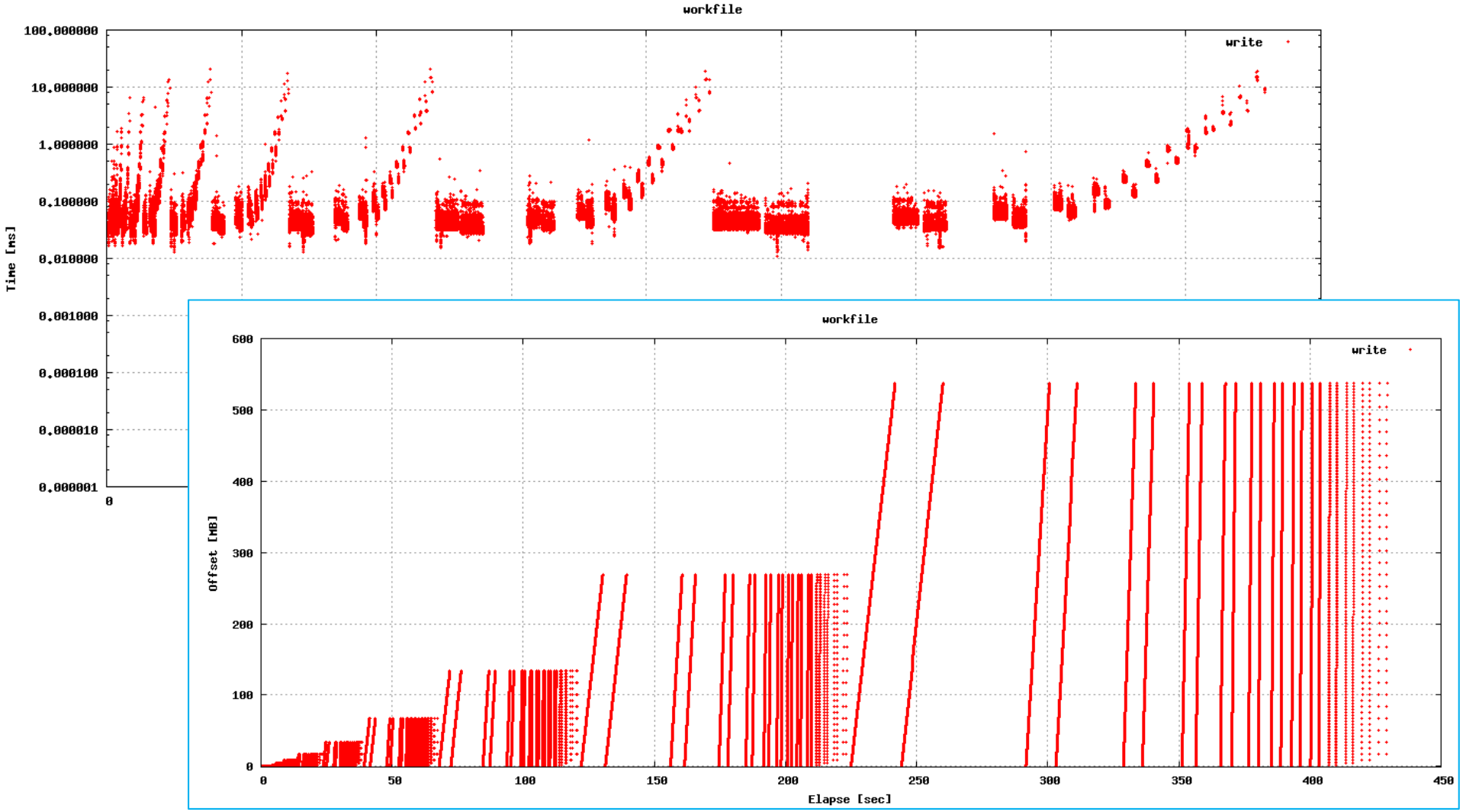
■IO-Dock使用例3(writeのIO長ヒストグラム)

```
# ./analyze_io_dist.pl /work3/tmp/work-starace-20150216.tmpdata -t /work3/tmp/workfile
#read
#size            num
0                0
1                0
2                0
4                0
8                0
16               0
32               0
64               0
128              0
256              0
512              0
1024             0
2048             0
4096             524284
8192             262156
16384            131092
32768            65560
65536            32794
131072           16408
262144           8214
524288           4116
1048576          2066
2097152          1040
4194304          526
8388608          268
16777216         138
```



IO-Dock使用例4(writeの時系列分析[実行時間・offset])

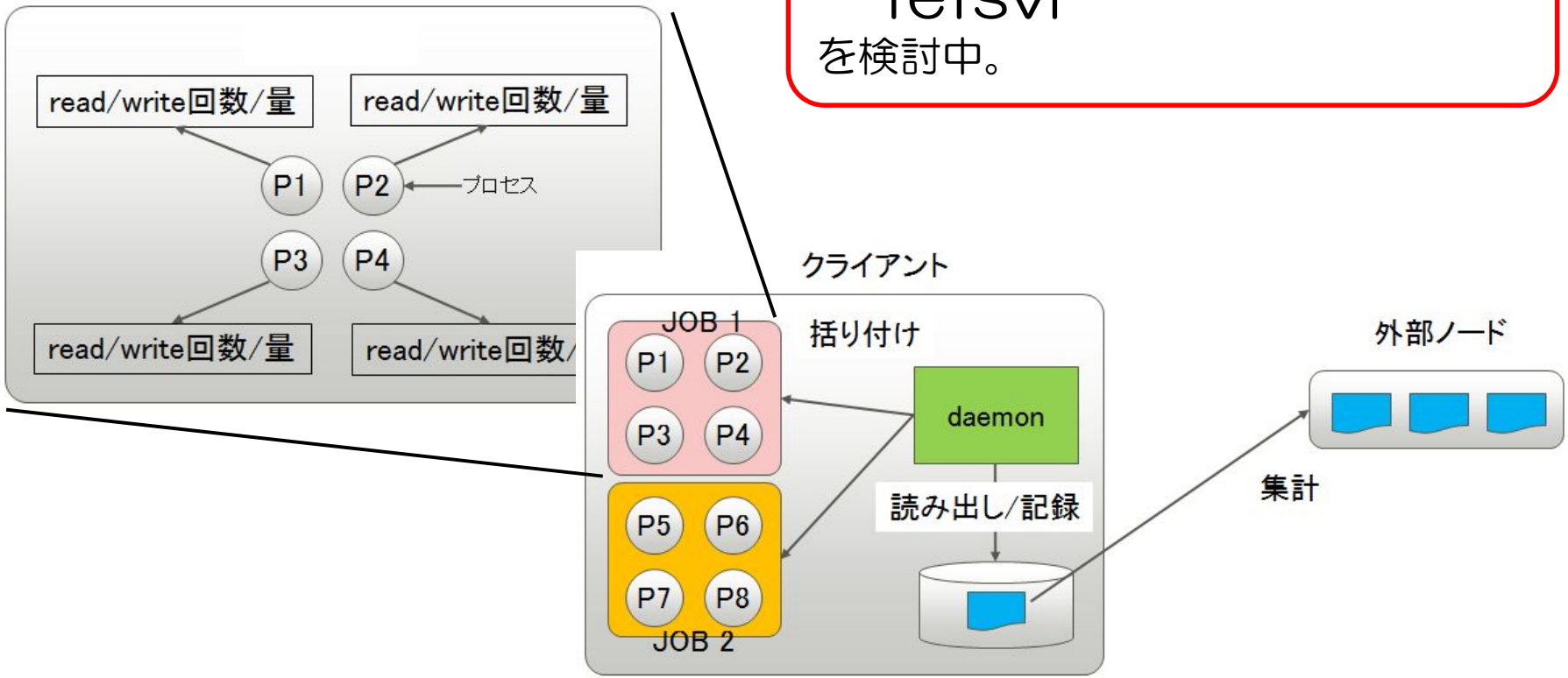
```
# ./analyze_io_per_time.pl /work3/tmp/work-starace-20150216.tmpdata -t /work3/tmp/workfile --offset write  --csv  > iozone_write_offset_per_time.csv
# ./analyze_io_per_time.pl ./iozone_write_offset_per_time.csv --plot TIME --title workfile --outfile ./iozone_write_exec_per_time
# ./analyze_io_per_time.pl ./iozone_write_offset_per_time.csv --plot OFFSET --title workfile --outfile ./iozone_write_offset_per_time
```



システム管理者レベルでの分析ツール

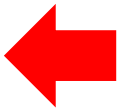
- IO-Dockは、ユーザが自分のアプリケーションのI/O処理内容確認に使用するもの。
- システム管理者は、
  ✓ジョブ毎のI/O発行量
  ✓システムリソースの利用割合
  を把握したい。

FEFSをターゲットにして、この様なシステム管理者のための分析ツール
“fefsvr”
を検討中。



Agenda

- はじめに
- ファイルシステムの利用における現状技術
- I/O効率改善のための4つのステップ
- 分析ツール
- 分析ツールによる解析事例
- おわりに



lseekが多すぎる(1/2)

●MPI化されていない、
かなり昔から利用されている、
データ処理系のプログラム

システムコール回数

syscall	実行回数	error回数
open	4,167	5
openat	165	16
close	6,030	0
unlink	3	0
fstat	7,804	35
fstat64	432	0
stat	1,153	858
stat64	10	0
lstat	3	0
lstat64	6	0
lseek	30,202	0
read	24,704	0
write	1,005	0

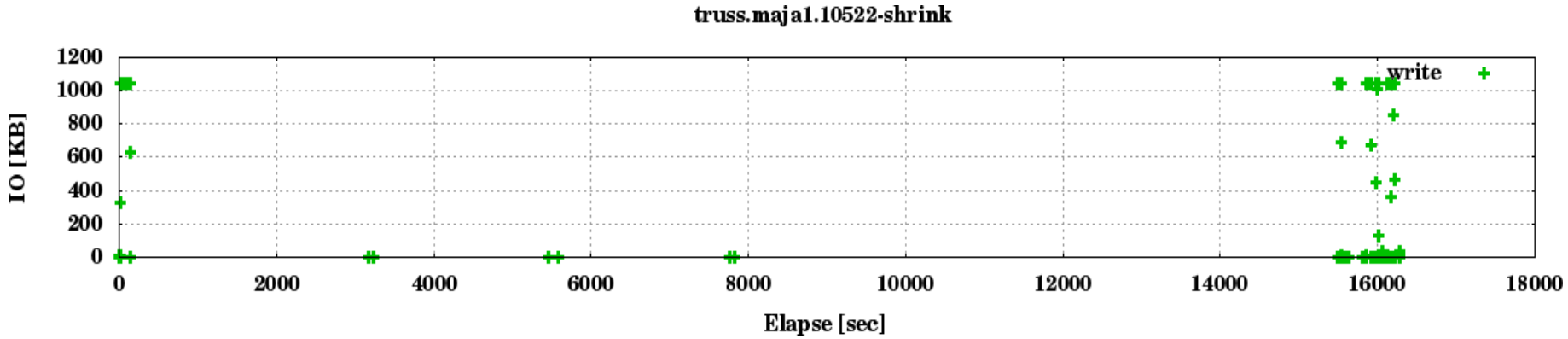
readの回数と量

ファイル名	回数	合計時間[sec]	総IO量[MB]	平均IO量[KiB]
file1	6,616	4.4694	6,935	1,024
file2	6,534	2.1041	3,115	466
file3	3,059	1.3713	1,946	621
file4	1,687	1.2035	1,665	964
file5	147	0.1076	151	1,006
/tmp/file6	134	0.0992	140	1,019
file7	133	0.1231	137	1,006
/tmp/file8	59	0.0426	62	1,018
/tmp/file9	58	0.0421	60	1,012
file10	1,224	0.0067	47	37
file11	1,224	0.0068	47	37

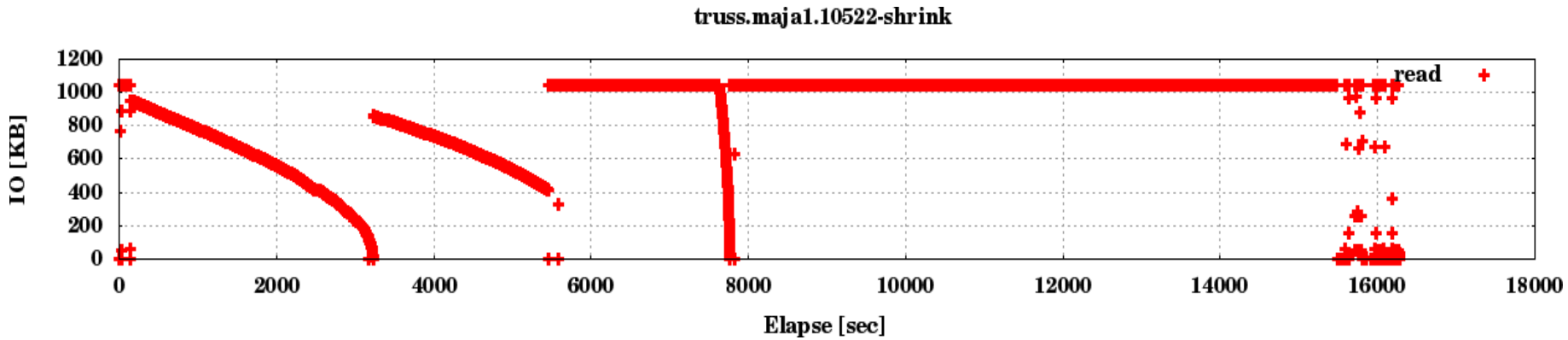
writeの回数と量

ファイル名	回数	合計時間	総IO量[MB]	平均IO量[KiB]
file4	71	0.1288	74	1,014
/tmp/file2	67	0.1579	70	1,019
file1	70	0.1467	67	931
/tmp/file3	59	0.1355	62	1,018
/tmp/file4	58	0.1272	60	1,012
file5	218	0.088	39	177
file6	211	0.0691	33	152

■lseekが多すぎる(2/2)



時系列のwrite量



時系列のread量

read時にlseekし、オフセットを変えつつもファイルの同じ領域を重複して読み込んでいる。この動作は、メインメモリ量が少ない計算機環境であったため、処理データが必要になる度に、ファイルの中にある当該データをreadにより取り出すというコーディングが現在まで残っているものと考えられる。⇒改善箇所発見

■open/closeが多すぎる(1/2)

●全国漢籍データベースの更新プログラム

システムコール回数

システムコール		実行回数	error回数	処理時間	I/O量
メタデータ	open(open64)	1,273,463	318,288	32.1 sec	-
	close	1,590,634	2	33.1 msec	-
	unlink	22	4	854 msec	-
	mkdir	1	0	0.10 msec	-
	rmdir	1	0	0.10 msec	-
	fstat64	636,990	0	25.1 msec	-
	lstat64	1,138,103	7	18.6 sec	-
	stat(stat64)	1,592,675	637,377	5.54 sec	-
ファイルI/O	read	12,047,329	0	110.4 sec	2.7GB
	write	1,283,041	38,273	58.6 sec	1.7GB

ファイル別情報

ファイル	ファイル数	open成功	open失敗
/platform/FJSV,GPUZC-M/lib/libc_psr.so.1	1	318,275	0
/lib/libc.so.1	1	318,272	0
その他共有ライブラリ	8	182	0
/var/ld/ld.config	1	0	318,272
data/FA*/tagged*/*.dat	318,169	318,025	0
その他	123	432	16

■open/closeが多すぎる(2/2)

```
while read F
do echo '<OTDoc><OTFile>'"$F"'</OTFile>'
    cat “$F”
    echo '</OTDoc>'
done
```

⇒

```
nawk '
{
    f=$0;
    printf(“<OTDoc><OTFile>%s</OTFile>¥n”,f);
    while((getline u<f)>0)
        printf(“%s¥n”,u);
    close(f);
    printf(“</OTDoc>¥n”);
}'
```

更新プログラム全体の処理速度が
約3倍高速になった。

実行回数 (改良前)	システムコール名　または、ファイル名	実行回数 (改良後)
1,273,463	open(open64)	200,348
1,590,634	close	200,984
318,275	/platform/FJsV,GPUZC-M/ib/libc?psr.so.1	243
318,272	/lib/libc.so.1	243
318,272	/var/ld/ld.config	243

■lgetxattrの最大処理時間が長すぎる(1/2)

●多波長データ解析システムで時々lsの反応が遅くなるという既知の問題あり。
約7000ファイルを入れたディレクトリに、一時間おきに/bin/ls -ltr する。

表4.3.4.1

ls実行時のI/O関連
システムコールの統計
（速い時）

syscall	実行回数	error回数	処理時間[s]	平均処理時間[s]	最少処理時間[s]	最大処理時間[s]
fcntl	1	0	0.000009	0.000009	0.000009	0.000009
read	14	0	0.000136	0.000010	0.000008	0.000016
write	129	0	0.002934	0.000023	0.000021	0.000038
stat	6902	0	0.071214	0.000010	0.000008	0.000034
open	12	0	0.000347	0.000029	0.000010	0.000214
getdents	12	0	0.000650	0.000054	0.000027	0.000065
close	19	0	0.000187	0.000010	0.000008	0.000019
fstat	13	0	0.000112	0.000009	0.000008	0.000011
statfs	1	0	0.000011	0.000011	0.000011	0.000011
lseek	1	0	0.000009	0.000009	0.000009	0.000009
lgetxattr	20712	20712	0.249096	0.000012	0.000008	0.000055
lstat	13808	0	0.795326	0.000058	0.000010	0.000683

表4.3.4.2

ls実行時のI/O関連
システムコールの統計
（遅い時）

syscall	実行回数	error回数	処理時間[s]	平均処理時間[s]	最少処理時間[s]	最大処理時間[s]
fcntl	1	0	0.000012	0.000012	0.000012	0.000012
read	12	0	0.000117	0.000010	0.000008	0.000016
write	129	0	0.003112	0.000024	0.000022	0.000041
stat	6902	0	0.075346	0.000011	0.000008	0.000961
open	12	0	0.038194	0.003183	0.000009	0.038064
getdents	12	0	0.320007	0.026667	0.000048	0.163230
close	18	0	0.000177	0.000010	0.000008	0.000016
fstat	13	0	0.000144	0.000011	0.000008	0.000042
statfs	1	0	0.000011	0.000011	0.000011	0.000011
lseek	1	0	0.000008	0.000008	0.000008	0.000008
lgetxattr	20712	20712	5.545225	0.000268	0.000011	0.722723
lstat	13808	0	0.314374	0.000023	0.000009	0.113075

■lgetxattrの最大処理時間が長すぎる(2/2)

lgetxattrシステムコールが全部エラーになっている。
→ACL(Access Control List)機能をOSが未サポートにも関わらず、
　noaclオプションを付けずにNFSマウントしていたことが原因⇒修正済み

反応遅延がNFSサーバ側へのACL問い合わせ時に発生していることが判明。
→NFSサーバカーネル周りで割り込み処理の設定調整を実施
⇒約3割の速度向上

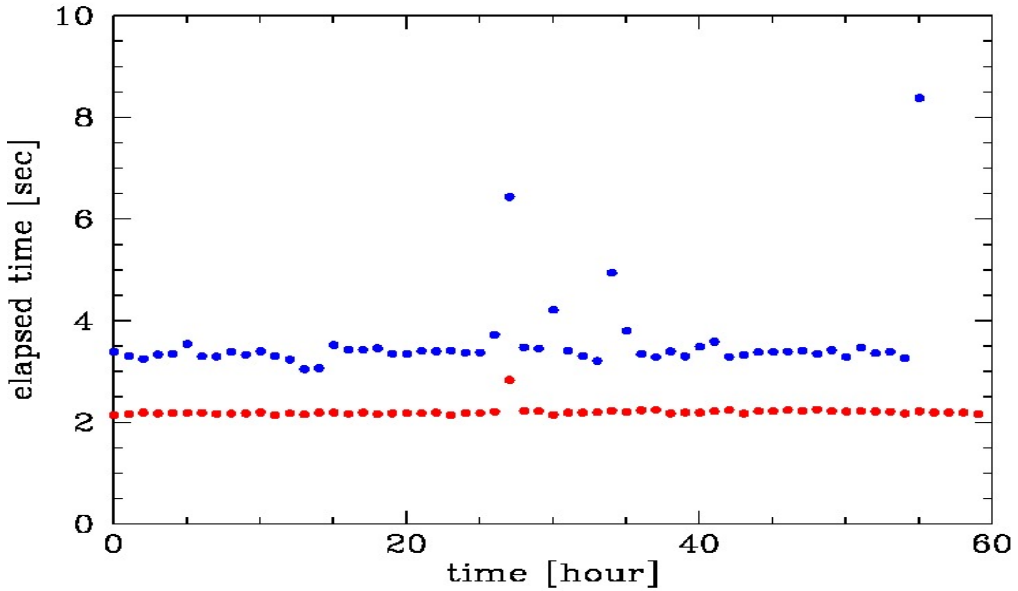


図4.3.10　lsの所要時間。1時間毎の測定結果。改善前(青)と改善後(赤)

■read総量/lseek回数が多すぎる(1/2)

●実験データのファイル形式を変換するプログラム

表4.4.3 システムコール分析結果（リード・ライト処理関連）

システムコール	ファイル	実行回数	処理時間	データ量	IO速度
read (全63ファイル中Top 5)	event_run071482.000010	3,092,219	35.0 sec	25.4 GB	726 MB/s
	/usr/local/sklib_g77/root_v5.28.00b/etc/system.rootrc	7	45 us	24.2 KB	538 MB/s
	/usr/local/sklib_g77/root_v5.28.00b/[illegible]de/RtypesCint.h	3	21 us	4.2 KB	199 MB/s
	/usr/local/sklib_g77/root_v5.28.00b/etc/[illegible]/TSystem/P020_TDCacheSystem.C	6	32 us	1.3 KB	42 MB/s
	/tmp/tmpfpNdvMS	2	9 us	1.2 KB	128 MB/s
write (全9ファイル中Top 5)	rfm_run071482.000010.root	401	329 ms	312 MB	947 MB/s
	/tmp/tmpfpNdvMS	1	15 us	1.2 KB	77 MB/s
	/tmp/tmpfuUpPTV	1	13 us	1.0 KB	77 MB/s
	/tmp/tmpf9tSf1Y	1	14 us	82 B	5.9 MB/s
	/tmp/tmpfuhiHkD	1	21 us	44 B	2.1 MB/s

500MBのファイル

表4.4.4 システムコール分析結果（ディスクI/O処理関連）

システムコール		実行回数	error回数	処理時間	IO量	備考
メタアクセス	open	190	114	27.7 ms	-	
	close	80	0	1.21 ms	-	
	unlink	8	0	0.06 ms	-	
	stat	2,491	124	38.0 ms	-	
ファイルIO	read	3,092,335	0	35.0 sec	25.4 GB	ほとんどが"event_run071482.000010"へのread
	write	427	0	329 ms	312 MB	ほとんどが"rfm_run071482.000010.root"へのwrite
その他	lseek	4,638,810	0	25.6 sec	-	ほとんどが"event_run071482.000010"へのlseek

■read総量/lseek回数が多すぎる(2/2)

データの読み込み方を変更し、バッファリングがうまく働くように調整した。
(OSの更新[RHEL5から6へ]も改善に寄与している)
→read量は50%削減
→lseekにかかる時間も2/3に削減

		修正前				修正後			
システムコール		実行回数	error回数	処理時間	IO量	実行回数	error回数	処理時間	IO量
メタデータ	open	332	244	14.1ms	-	374	286	40.6ms	-
	close	93	0	1.4ms	-	93	0	1.5ms	-
	unlink	8	0	0.1ms	-	8	0	0.3ms	-
ファイルIO	read	3,167,103	0	48.0sec	26.0GB	1,583,929	0	25.7sec	13.0GB
	write	353	0	399.2ms	317.9MB	353	0	386.0ms	317.9MB
その他	lseek	4,750,841	3	46.1sec	-	3,167,360	3	31.0sec	-

■I/Oの特異点が観測される

●SPECsfs相当のベンチマーク用ワークロード

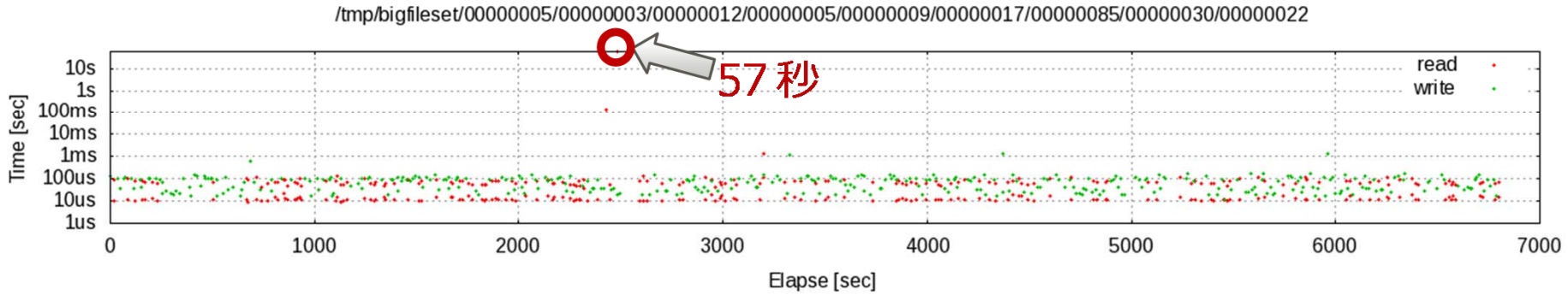


図4.5.4　read特異点

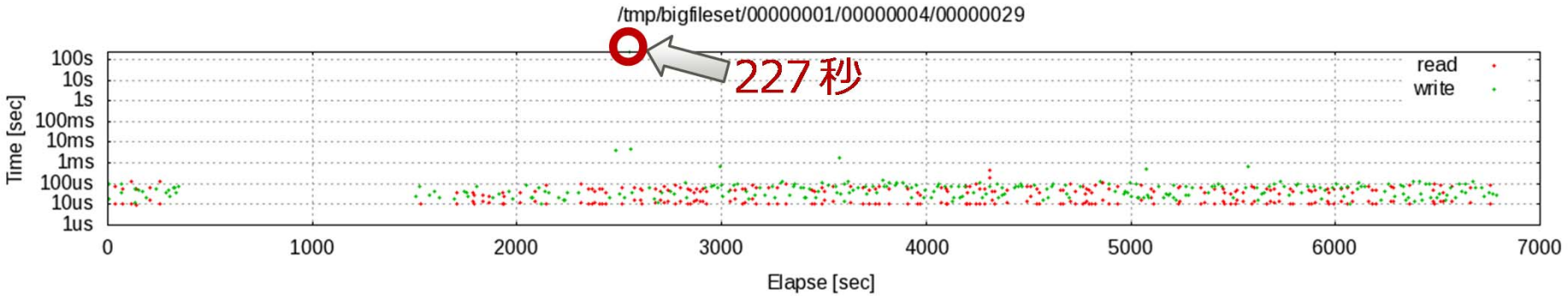
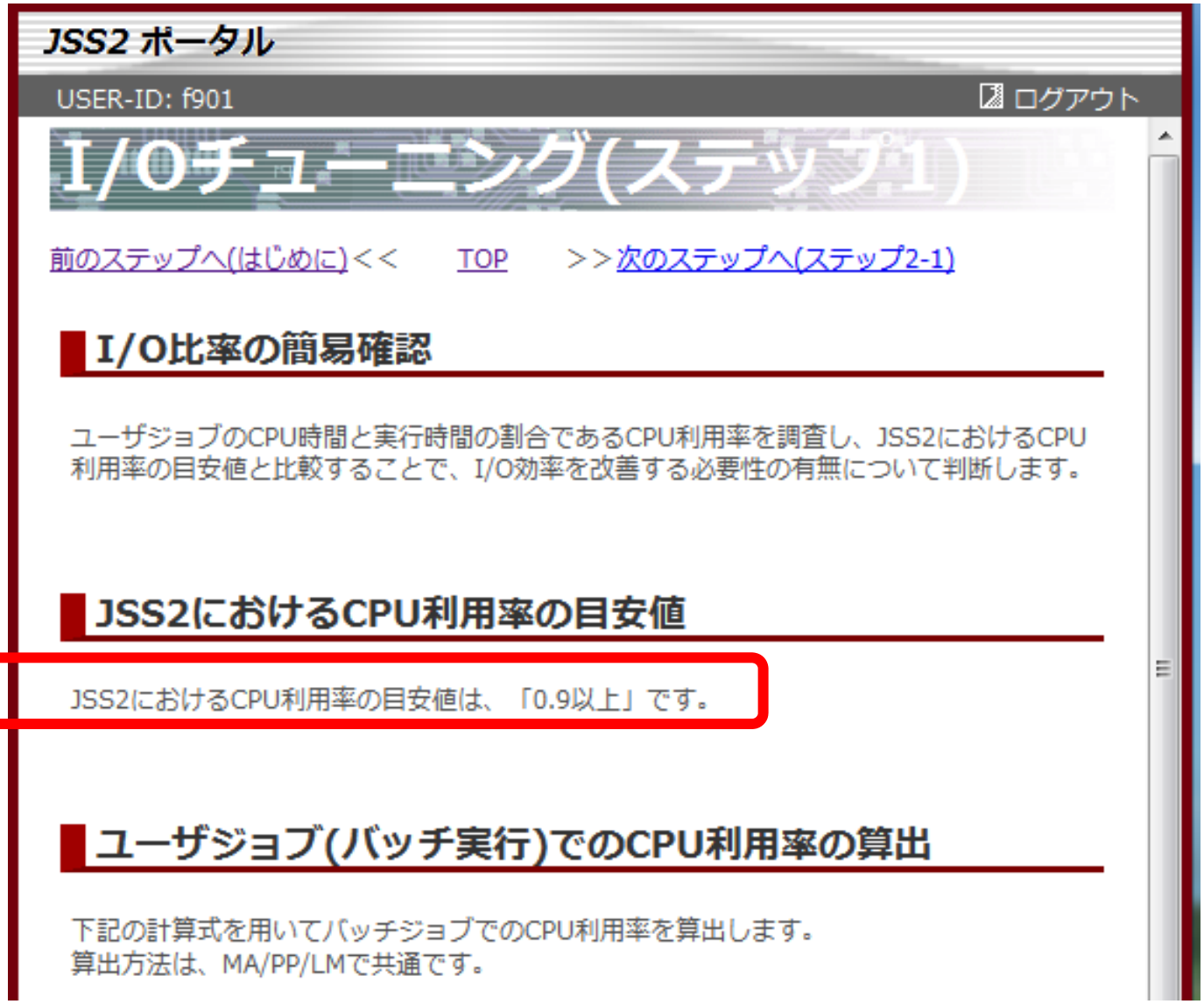


図4.5.5　write特異点

■ユーザへの展開例(マニュアルページトップ画面にリング)

■謝辞
ＳＥの皆さま、
特に、伊藤さん、大迫さん、大日向さん、荒川さん
迅速なコンテンツ整備ありがとうございました。

■ユーザへの展開例(ステップ1= I/O比率)



■ユーザへの展開例(ステップ2-1= I/O量)

JSS2 ポータル

USER-ID: f901 ログアウト

I/Oチューニング(ステップ2-1)

前のステップへ(ステップ1)<< TOP >>次のステップへ(ステップ2-2)

JSS2における想定I/O量の確認

ジョブの実行に先立ち、ジョブの想定しているI/O量を確認します。
ジョブを実行した後に表示されるI/O量と想定I/O量を比較することにより、想定以上にread,writeなどが発生していないことを確認します。

I/O量の見積もり

プログラムが想定するI/O量の見積もりを行います。
ジョブにおけるwrite,readの回数や繰り返し数、並列実行数などによりI/O量は変化します。

例えばI/Oのベンチマークプログラムである、IORを実行した場合の見積もり方法は以下の通りです。

ジョブ内で実行するコマンド
mpiexec -n 2 ./IOR/src/C/IOR -a POSIX -i 3 -B -t 100m -w -r -b 10000m -F -o ./testdir/

確認項目	数	備考
ファイルサイズ	10000MiB	-bの指定値です。
write,read	2	-w,-rを指定することで、write,readを行います。
繰り返し数	3	-iの指定値です。
並列実行数	2	mpiexec -nの指定値です。

この場合、 10000[MiB] x 2 x 3 x 2 = 120000[MiB] = 125829.12 [MB] がジョブの想定するI/O量となります。

ジョブのI/O量の確認方法

■ユーザへの展開例(ステップ2-2= I/O性能)

JSS2 ポータル

USER-ID: f901　　ログアウト

I/Oチューニング(ステップ2-2)

前のステップへ(ステップ2-1)<<　TOP　>>次のステップへ(ステップ2-3)

JSS2におけるI/O性能とプログラムのI/O性能の確認

プログラムの実行時に、プログラムのI/O性能とシステムの典型的な
乖離が大きい場合には、I/O効率改善の可能性があります。

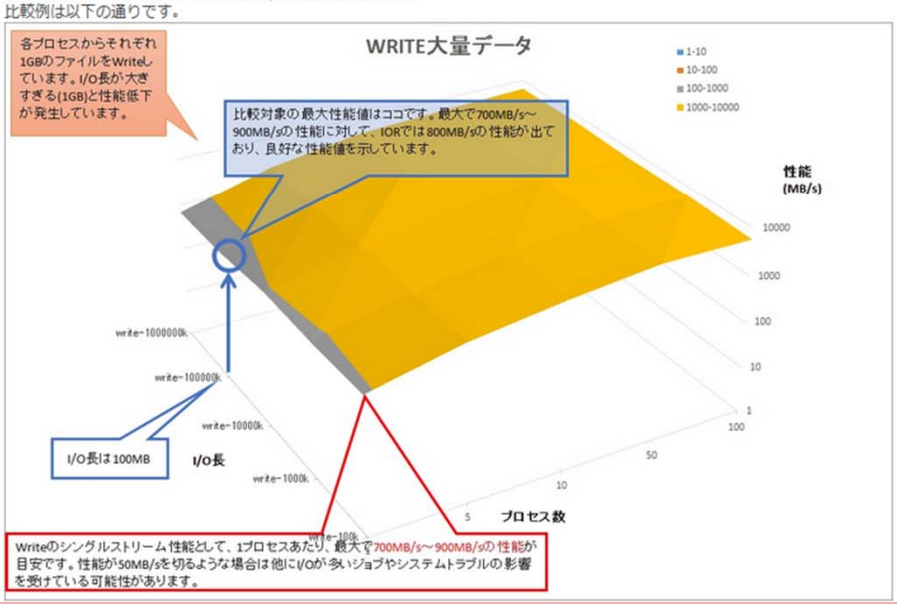
JSS2のI/O性能測定結果

I/O性能の目安として、JSSのI/O性能測定結果を公開します。
I/O性能測定には、健康診断ツールを利用しています。

3. I/O長を確認します。
IORプログラムでは、IORを実行するときのオプション(-t)でI/O長が変更可能です。
以下のように実行していますので、-t で指定している"100m"がI/O長を示しています。

```
mpiexec -n 2 ./IOR/src/C/IOR -a POSIX -i 3 -B -t 100m -w -r -b 10000m -F -o ./testdir/
```

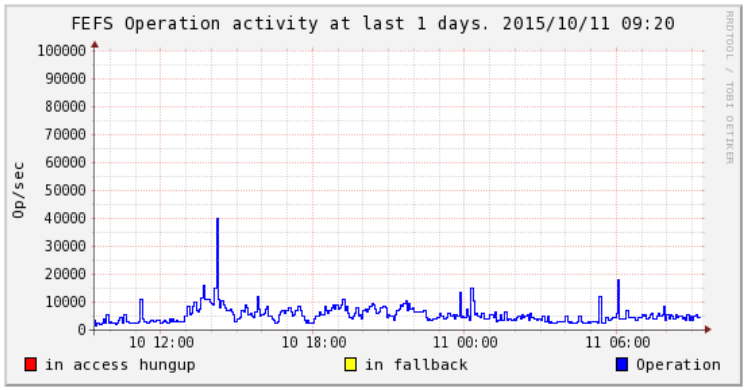
4. 1,2,3の情報から健康診断ツールの結果と比較し、プログラムのI/O性能が妥当かを確認します。
1から、SORA-PPで測定した結果を使用します。
2と3から、測定条件に一番近い比較対象のグラフを選択します。
今回は、10GBのファイルを100MBのI/O長でwriteしていますので、100MBのI/O長で1GBのwriteを測定している「WRITE大量データ」のグラフを選択し、プログラムのI/O性能と比較します。
比較例は以下の通りです。



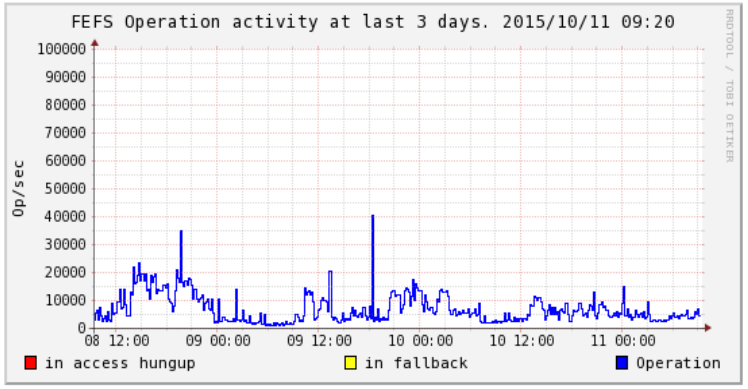
■ユーザへの展開例(ステップ2-3= システム負荷)

SORA-FS操作数

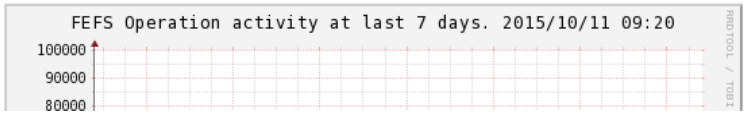
過去1日



過去3日

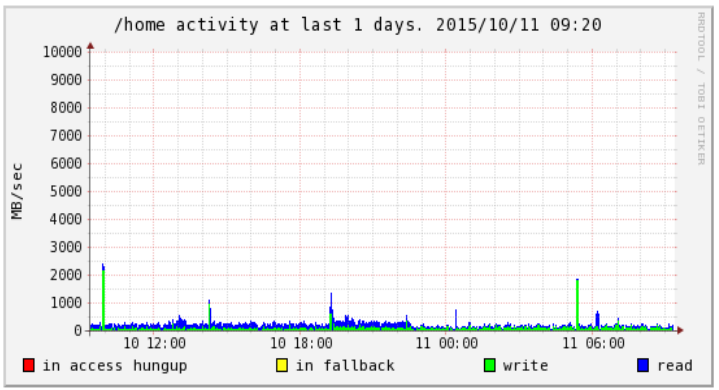


過去7日

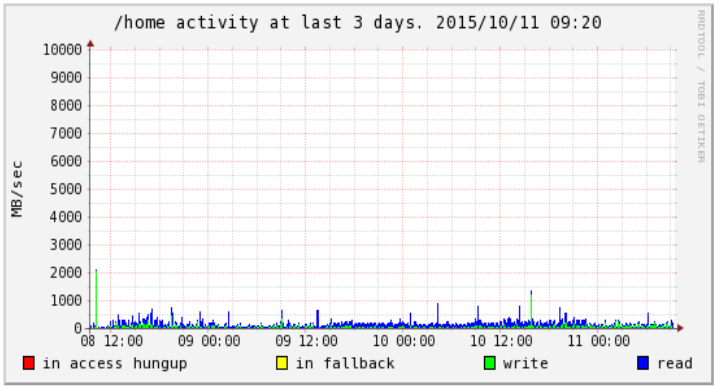


/home I/O量

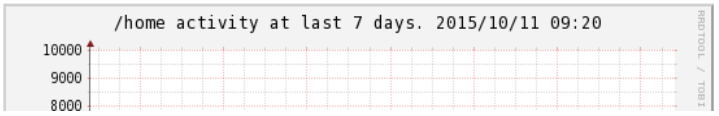
過去1日



過去3日

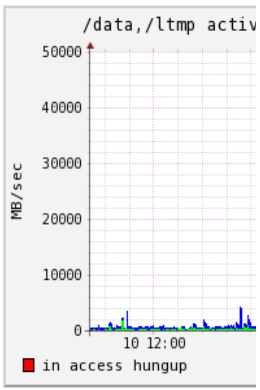


過去7日

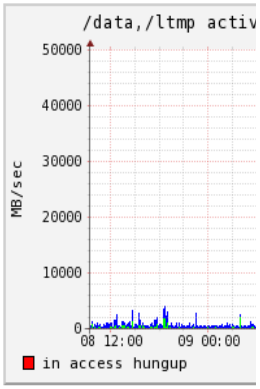


/data,/ltmp I/O量

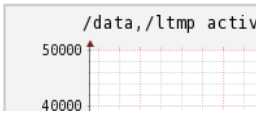
過去1日



過去3日



過去7日



■ユーザへの展開例(ステップ3= システムコール分析)

JSS2 ポータル

USER-ID: f901　　ログアウト

I/Oチューニング(ステップ3)

前のステップへ(ステップ2-3)<<　　TOP

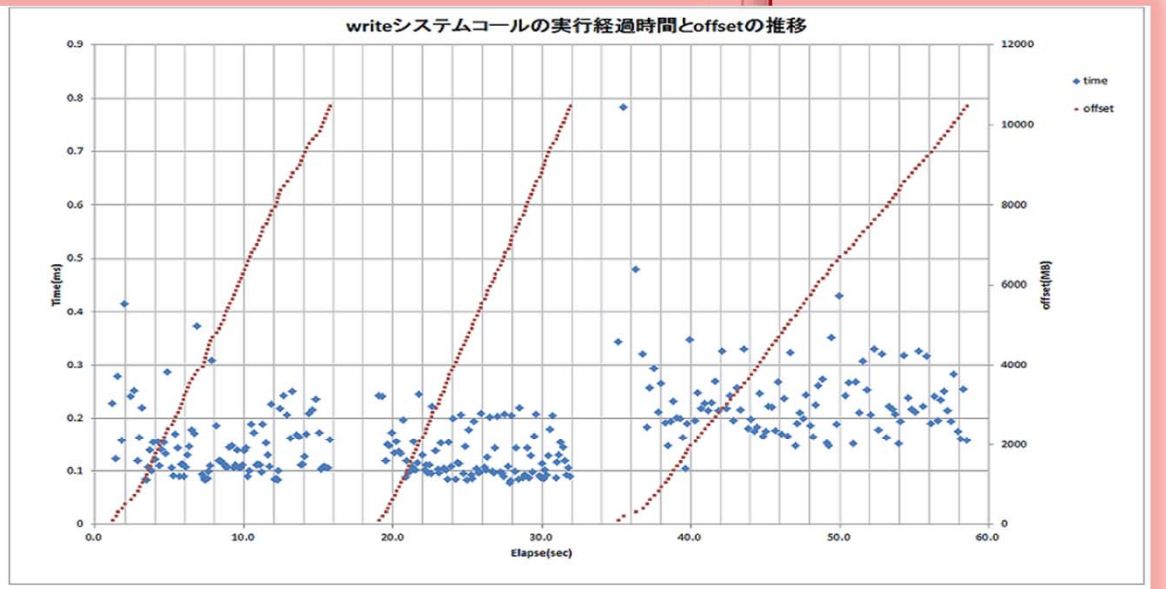
システムコール分析ツール(IO-Dock)について

本ステップでは、ユーザジョブに対してシステム毎の分析を行いますが、
IO-Dockは、ファイルシステム利用技術WGの成果物であり、straceコマ
ズ、ファイルオフセットの項目を集計するツールになります。

まずは、IO-Dockを以下のサイトからダウンロードし、同ページにアッ
www.ssken.gr.jp/MAINSITE/download/wg_report/fs/index.htm

IO-Dockの実行例

以下にユーザプログラムの一例としてIOベンチマークプログラムのIOR
※詳細の手順は、IO-Dockの「使用手引書」および「ファイルI/Oの分析



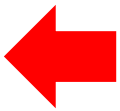
8. 簡単な分析結果

上記7項のグラフ「writeシステムコールとoffsetの推移」を確認すると3回目のwriteは、1回目と2回目に比べて実行時間が長くなっています。
再現性を確認するため、再度同じジョブを実行しましたが、3回目のみ実行時間が長くなるような現象は確認できませんでした。
そのため、「ステップ2-3 システム負荷の確認」を基に当該ジョブを実行した時間帯のファイルシステム負荷状況を確認しましたが、高負荷状態ではありませんでした。
再現性は無く、一時的なファイルシステム負荷も無いことから、ファイルシステム(FEFS)の挙動に目を向け(「はじめに」に記載のJSS2のファイルシステム構成の運用方法を確認)、
2プロセスからのファイルI/OがどのOSTボリュームに書き込まれたのかをシステム管理者の操作 * で確認したところ、2ファイルが同じOSTに書き込まれていました。
以上のことから3回目のwriteは1回目と2回目に比べて実行時間が長くなっていたことが分かりました。
FEFS では、ファイルのデータをOST に分散して格納しますが、分散方式としてラウンドロビン方式とストライプ方式があり、JSS2では、ラウンドロビン方式(デフォルト)を採用しているため、今回のように同一OSTに同時に書き込まれることもあります。

* システム管理者側での調査が必要になる場合もありますので、何か気になる点がありましたらスパコン運用窓口まで、ご相談ください。

Agenda

- はじめに
- ファイルシステムの利用における現状技術
- I/O効率改善のための4つのステップ
- 分析ツール
- 分析ツールによる解析事例
- おわりに



おわりに

- ユーザの視点に立った、モニタリング方法・チューニング指針等をまとめた4つのステップ
  - ステップ1 … I/O比率の簡易確認
  - ステップ2 … 想定I/O量、想定I/O性能、システム負荷の確認
  - ステップ3 … システムコール分析
  - ステップ4 … システム管理者による詳細分析

- システムコール分析ツール”IO-Dock”を用いた実際のチューニング事例

- 今後のI/Oに関する課題
  - システム全体での情報採取技術など、I/Oの測定技術
  - バッチ実行時のI/O性能取得など、チューニング支援
  - MPI-IO、HDF5、KVSなど、並列I/Oの普及

ご清聴ありがとうございました。

■謝辞
WGに参加し活発な意見交換をしていただいた皆様、
SS研事務局の皆様に感謝の意を表します。