フラグメンテーション解析を支援する DAV と LKST Log Tools の開発

平松 雅巳†，杉田 由美子†，藤原 哲‡

†(株)日立製作所　システム開発研究所，‡日立ハイブリッドネットワーク株式会社

{hiramatu,sugita}@sdl.hitachi.co.jp, satoshi-fujiwara@hitachi-hybrid.co.jp

概要

Linux システムにおいて，フラグメンテーションが原因と推察されるファイル I/O 性能の低下が散見される．しかし，現在の Linux においては，ディスクやファイルのフラグメンテーションを解析できるツールがなく，性能劣化との係わり合いを確認する方法がない．そこで我々は，ディスク上のフラグメンテーションの状態を静的に解析するツール DAV と，ファイル I/O の動作状態のトレース情報から処理性能を収集・解析するツール LKST Log Tools を開発した．本稿ではこれら 2 つのツールの説明をし，その有効性を確認する実験により． ext3 ファイルシステムにおいて，どのようにフラグメンテーションが発生し，ファイル I/O 性能がどのような状態にあるかの情報を得られることを確認した．

キーワード : Linux， DAV，LKST，トレーサ，フラグメンテーション，ファイル I/O

1. はじめに

近年 Linux がエンタープライズシステムへ適用されるようになり，日常的に負荷がかかるシステムの運用を続けていくと，大容量ファイルのバックアップ時に，ファイル全体のシーケンシャルリードに時間がかかるようになるなど，フラグメンテーションが原因と考えられるファイル I/O の性能低下事例が散見されるようになった. 今後 Linux のエンタープライズ用途への適用拡大に伴い，システムが扱うファイル数・ファイルサイズが増加し，負荷が増大した時に，同様の事例が増加する危険性がある．システム性能の安定化と信頼性の向上のためには，フラグメンテーションの状態やファイル I/O 性能への影響を知る手段が必要である．

従来の Linux では，fsck により，フラグメンテーションを起こしたファイル数の割合を得ることができる．これに加え，個々のファイルがどの程度のフラグメントに分かれているのかという情報や，フラグメンテーションに関わるカーネル処理時間の情報，またそれらの情報を分かりやすく分析，可視化などを行う手段があれば，管理者がサーバによるディスクのヘルスチェックや，カーネル開発者による改善点の発見が，容易になると考えられる．そこで，性能解析支援ツールとして，ファイルのフラグメンテーション情報を収集し可視化するツール DAV と，Linux カーネルのトレース情報から，ファイル I/O の処理時間を収集・解析するツール LKST Log Tools を開発した．

LKST Log Tools は，カーネル内部の処理単位に性能を取得し，フラグメンテーション時の性能分析を行なうことができるが Linux カーネルを変更する必要がある．これに対し DAV はある瞬間のフラグメンテーション状況を視覚的に捉えることができる．LKST に比べ情報は限定的だが，Linux カーネルを変更する必要はなく，すでに稼動しているサーバにおいても容易に情報を取得できる．

DAV と LKST Log Tools の有効性を確認するため，フラグメンテーションによりファイルのシーケンシャルリード性能の低下が発生する状態を作り，実験を行なった．その結果，フラグメンテーションの発生状況を確認でき，さらにファイル I/O に関わる処理性能の傾向をつかむことができた．また，障害が生じた時に対象サーバで DAV を使ってフラグメンテーション状態情報を得て，そのデータを参考に解析サーバで LKST を使って詳細に解析して原因を究明していく，という連携利用の可能性を得ることができた．

2. 性能解析支援ツールの開発

2.1. DAV(Disk Allocation Viewer)の開発

Linux では，フラグメンテーションの状態の情報として fsck 等でフラグメンテーションを起こしたファイルが，全ファイル中に占めるパーセンテージを取得することができるが，どのようなフラグメンテーションが，どのファイルで起こっているのかといった情報は得られない．例えば，特定ファイルがどれだけ断片化しているかという情報や，ディスク全体の断片化の状態を得ることができない．そこで我々は，Linux に不足しているこれらの機能を補うため DAV を開発した．[1][2]

DAV は, Linux の ext2／ext3 ファイルシステムのフラグメンテーション状況の情報を取得し視覚化するツールであり，下記の機能を持つ．

- ファイルシステムのマウント状態に関わらず，フラグメンテーション情報を取得

● パーティション全体，任意のディレクトリ下の全ファイル，単一ファイルの単位で情報を取得

● 上記情報の取得操作の GUI 化と結果の可視化

また DAV は，大きく分けて次の 2 つのプログラムで構成されている．

(1) dac（Disk Allocation Checker）：フラグメンテーション情報を取得

(2) dav（Disk Allocation Viewer）：(1)の情報を GUI 表示

dac は，指定されたファイルブロックをファイルの構成順に見て行き，ファイル上で連続して配置されているデータブロックが，ディスク上では連続していない状態をフラグメンテーションとして検出する．すなわち次に示した 2 つのパターンをフラグメンテーションとして認識する．

(a) ファイル上では隣り合う二つのデータブロックが，ディスク上では隣り合わずに配置．

(b) 二つのデータブロックが，ファイル上での並びとは逆の順番に配置．

dac は結果データをテキスト形式で出力する．これにより後で情報を再利用することが可能である．

一方で dav は dac の出力データを読み込み，フラグメンテーション状況を GUI で表示する．表示形式は以下の二通りである．

(a) 物理的なブロック番号順にシステムブロックを含めて表示

(b) 確認対象のファイルを構成するブロックだけを，その構成順に表示

dav が提供する GUI 画面の各パーツの概要を図 2-1 に示す．この GUI により，ファイル構成に詳しくないユーザでもフラグメンテーション状態を確認することができる．

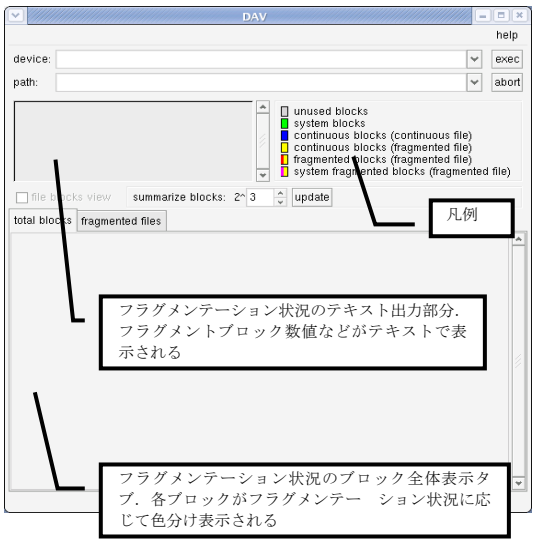

図 2-1 DAV の GUI

2.2. LKST による性能解析ツールの開発

DAV は，フラグメンテーションに関するファイル単位の静的な情報までを解析することができる．一方，発生したフラグメンテーションがどのように OS の性能に影響するかという動的な情報についても，詳細に解析する必要があると考えた．このため，我々が以前から開発しているカーネル内トレーサである LKST（Linux Kernel State Tracer）[3][4][5][6]を強化し，ファイル I/O を含むカーネルの主要な処理の動作解析・性能解析を行う機能（LKST Log Tools）を開発した．

2.2.1. LKST とは

LKST はカーネル内部で発生したイベントの情報を発生順に記録し保持する，カーネル状態トレーサである．LKST は，カーネルの主要な処理に対し，フックポイントと呼ぶ，動的に on/off 可能な関数呼び出し処理を，カーネルに対するパッチの形で追加する．フックポイントでは，その処理に固有のパラメータ(関数の引数やローカル変数など)を引数にして，イベントハンドラと呼ばれる特殊な関数を呼び出す．イベントハンドラでは，渡された引数と，フックポイント固有の ID，呼ばれた時間をメモリに記録する．ユーザはこれらの情報を解析することで，カーネル内部の動作を調べることができる．

2.2.2. LKST Log Tools とは

LKST の仕組みを強化してカーネルの性能評価のための情報収集を行ない，さらに収集した情報を解析するためのツールとして，LKST Log Tools を開発した．[7]

LKST Log Tools では，カーネル内部の処理性能を示す箇所及び変数を記録するため，LKST のフックポイントを新たに追加した．また，直接参照できない情報（リストの長さなど）を計算するイベントハンドラを追加した．これらの機能拡張により，LKST を利用してカーネルの主要な処理の性能情報を収集することが可能になった．

また，LKST Log Tools の一環として，LKST で収集した情報を解析するために，イベント解析ツール lkstla（LKST Log Analyzer）を開発した．lkstla は LKST で取得したログデータを解析し，性能評価を行うためのデータを得るコマンドである．現在 10 項目 25 種類のログデータ解析機能（アナライザ）を開発した．また解析した結果は，時系列表示・統計表示・分布表示の 3 通りで表示することが可能である．次に，ファイル IO の解析に関わる LKST Log Tools 機能について説明する．

2.2.3. ファイル I/O 処理の解析方法

Linux のファイル I/O 処理は，一般的に次の 3 つの層に分けられる．

(1) VFS 層
(2) ブロック I/O（BIO）層
(3) デバイスドライバ層

LKST Log Tools では LKST のフックポイントをこの層の間に挿入し，各層の実行にかかった時間や，各層間でやり取りされる情報を収集した．また，収集した情報を lkstla で解析するため，ファイル I/O 用の 4 種類のアナライザ(syscall, biotime, elvtime, iosector)を開発した．Linux のファイル I/O 処理と, 上記アナライザの関係を図 2-2 を用いて説明する．

(1) syscall アナライザ
syscall アナライザは，LKST のイベントのうち，システムコールの入り口(system call enter)と出口(system call exit)の間の時間を計測する．read/write システムコールの時間を計測することで，ファイル I/O の全ての層の処理時間を解析する．

(2) biotime アナライザ
biotime アナライザは，VFS 層から BIO 層への I/O 要求の受け渡し(generic_make_request)と I/O 要求の処理完了(bio_end_io)の間の時間を計測する．これにより，BIO 層とデバイスドライバ層の処理時間を解析する．

(3) elvtime アナライザ
elvtime アナライザは, BIO 層内部で I/O 要求を IO スケジューラへ渡す処理(__make_request)から IO スケジューラから取り出してデバイスドライバへ渡す処理(elv_next_request)の間の時間を計測する．これにより，IO スケジューラの処理時間を解析する．

(4) iosector アナライザ
I/O 要求がデバイスドライバへ送られる際に，アクセスするセクタが記録される．これにより，どのセクタへいつアクセスしたのかを解析する．

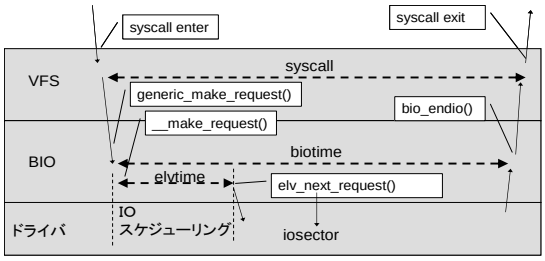

図 2-2 LKST によるファイル I/O 処理の記録ポイント

次節では，この DAV と LKST Log Tools を使った検証を行い，有効性を確認する．

なお，DAV および LKST 及び LKST Log Tools のその他の機能などの情報，LKST 本体のソースコードなどは，参考文献[1]～[3]及び[7]に示したサイトから取得できる．

3. 有効性の確認実験

DAV と LKST Log Tools の有効性を確認するため，フラグメンテーションによりファイルのシーケンシャルリード性能低下が発生する状態を作り，以下の実験を行なった．

実験は IOzone による並列書き込みを行ったときのフラグメンテーションの状態を DAV で確認し，このファイルに対するシーケンシャルリード性能を LKST Log Tools で分析した．

3.1. 実施環境

(1) ハードウェア

実験で用いたハードウェアの仕様を表 3-1 に示す．

表 3-1 ハードウェア仕様

項目	仕様
CPU	Intel Xeon 3.6GHz Dual （HT 有効）
Memory	8GB
SCSI	AHA-3960D U160/m
HDD	U320SCSI/300GB/10025rpm 8 台

(2) ソフトウェア

対象オペレーティングシステムとして Fedora Core 2[8]を用い，カーネルとして Linux2.6.9[9]に性能評価パッチ(LKST パッチ)を当てたものを利用した．評価用のファイルシステムとして ext3 ファイルシステムを用いた．ext3 ファイルシステムの主要なパラメータを表 3-2 に示す．また，IO スケジューラとして, Linux カーネル 2.6 のデフォ

ルトの IO スケジューラである anticipatory scheduler を用いた．

表 3-2 ext3fs の主要パラメータ

項目	値
mount option	rw
journaling mode	ordered data
features	has_journal, resize_inode, filetype, needs_recovery, sparse_super, large_file
Block size	4096

また，今回ファイル I/O のシーケンシャルリード性能を計測するツールとして，プロセスの並列度の設定が可能であり，書き込み・読み込みの I/O スループットの測定が可能な IOzone ver3.233[10]を利用した．

(3) フラグメンテーションの作成

測定対象とするパーティションに対し，16 個の別個のファイルに書き込みを行った．一回の I/O サイズを表すレコードサイズは 1MB，ファイルサイズは 20MB とした．

ファイル作成は，一度に 1，2，4，8，16 プロセスを並列で書き込み処理する．本稿ではそれぞれをパターン A~E と表す．この書き込みパターンを表 3-3 に示す．

表 3-3 ファイル作成パターン

パターン	並列プロセス数	同時作成ファイル数	作成回数
A	1	1	1 6
B	2	2	8
C	4	4	4
D	8	8	2
E	1 6	1 6	1

3.2. DAV を使ったフラグメンテーション状態情報の取得

ファイルの各書き込みパターンにおけるディスク上のフラグメンテーション状態を，DAV を使って確認した．並列度を 1(パターンA)，4（パターンC），16（パターンE）と上げていった時のファイルのフラグメンテーション状態を DAV で解析したものを，図 3-1～図 3-3 に示す．フラグメンテーション状態の凡例を図 3-4 に示す．さらに，パターン A~E において，dac により取得したファイルのフラグメント数の平均を，図 3-5 に示す．

DAV の出力から，書き込みの並列度を上げると，フラグメンテーションが増えている様子が視覚的に確認できる．

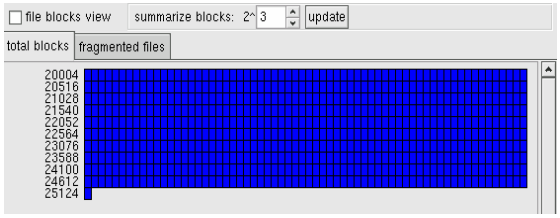

図 3-1 パターン A の状態

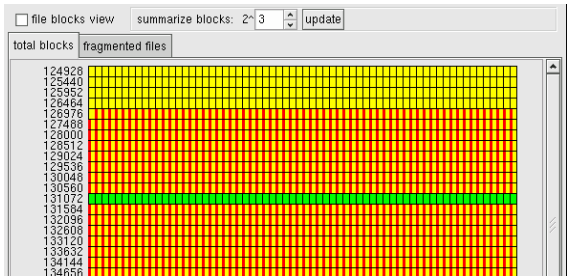

図 3-2 パターン C の状態

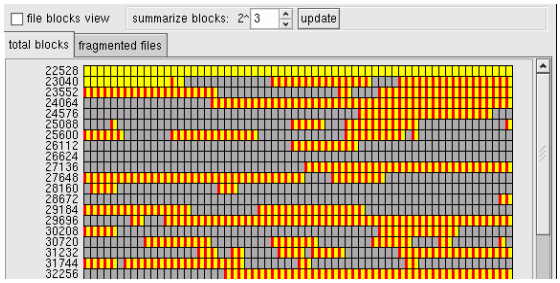

図 3-3 パターン E の状態

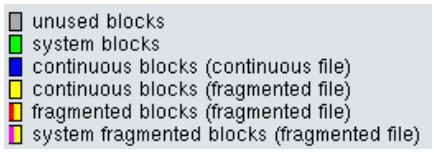

図 3-4 DAV の凡例

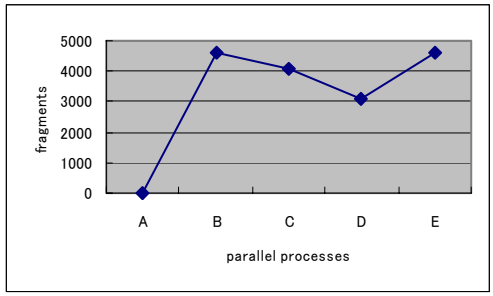

図 3-5 パターンごとのフラグメント数

3.3. LKST を使ったファイル I/O 処理時間の分析

それぞれのパターンで作成したファイルに対し，IOzone により 1 プロセスでシーケンシャルリードを行い，スループットを得る．

IOzone のパラメータとして，1 プロセスでの read スループットを測定するモード(モード 1)を用いる．ファイルへのアクセスサイズは 1 MB とし，ファイル全体（２０MB）を読み出し，それ以外はデフォルト設定とした．図 3-6 に，この測定結果を示す．この結果から，2 並列で書き込んだファイルの場合(パターン B)の読み込みスループットが，1 並列で書き込んだファイルの場合(パターン A)に対して，1/3 以下になっていることが分かる．また，16 並列で書き込んだファイルの場合(パターン E)は約 1/5 程度のスループットに落ち込んでいる．

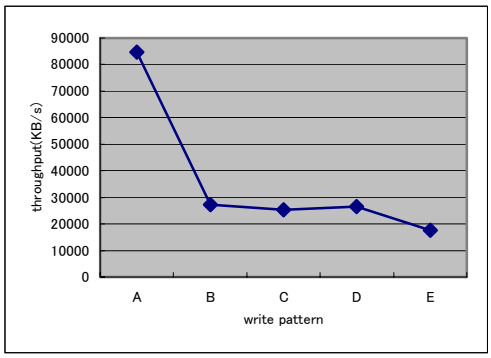

図 3-6 パターンごとのスループット

この測定実験中のカーネルの動作について，LKST を用いて解析した．

LKST を有効にしたカーネルを用いて，シーケンシャルリード中のトレース情報を取得し，ファイル I/O 処理にかかった詳細な処理時間を解析した．解析対象は 2.2.3 で示したシステムコールの時間(syscall)・BIO 処理時間(biotime)・I/O スケジューラの処理時間(elvtime)である．

図 3-7 から図 3-9 に，パターン A, C, E で書き込みを行ったファイルを，1 つのプロセスで読み込む処理を行ったときの LKST による解析結果を示す．図中，X 軸に経過時間を，Y 軸にそれぞれの処理にかかった時間を表している．これらの図から，書き込み処理の並列度が上がるにつれ，システムコール処理時間と BIO 処理時間が遅くなっていくことが確認できた．

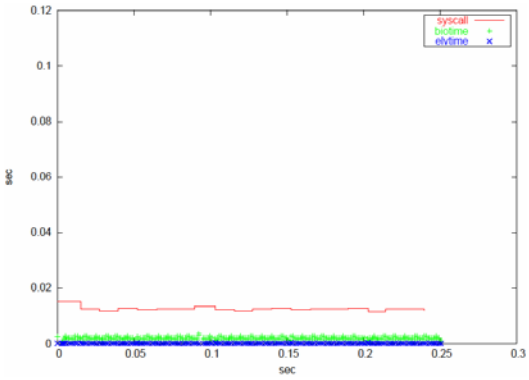

図 3-7 パターン A の処理時間

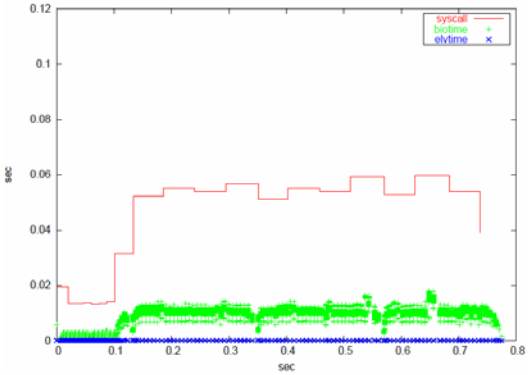

図 3-8 パターン C の処理時間

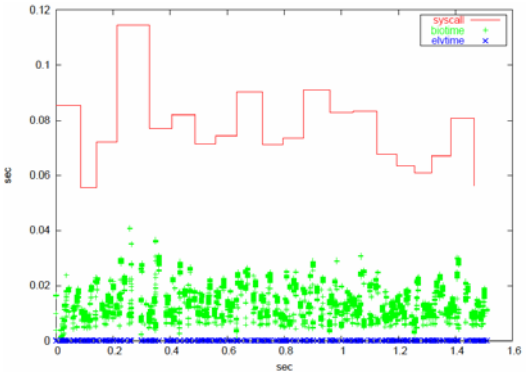

図 3-9 パターン E の処理時間

そこで，このシステムコールの処理時間と BIO の処理時間の関係を次に解析する．図 3-10 にその結果を示す．図中，X 軸は read システムコール処理時間, Y 軸は bio 処理時間である．avg, max, min は，それぞれ，bio 処理時間の平均，最大，最小を示す．

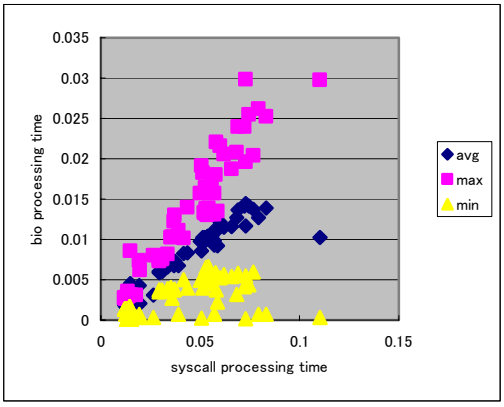

図 3-10　システムコールと BIO の処理時間

このグラフから，時間のかかっている read システムコールの処理ほど，BIO 処理の平均処理時間・最大処理時間が大きくなっているのが分かる．

次に同様に図 3-11 にシステムコール処理時間と，IO スケジューラ処理時間の関係を示す．図中，X 軸は read システムコール処理時間，Y 軸は IO スケジューラ処理時間である．avg, max, min は，それぞれ，IO スケジューラ処理時間の平均，最大，最小を示す．

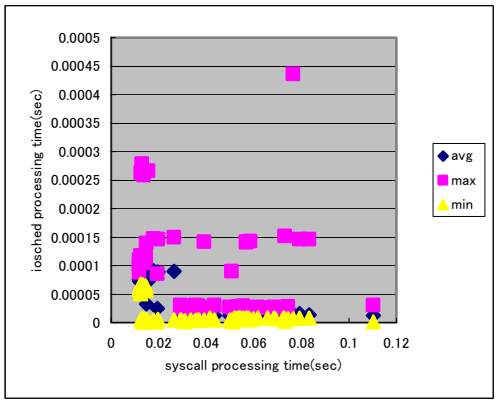

図 3-11　システムコールと IO スケジューラの処理時間

このグラフから，read システムコールの処理時間が長くなっても，IO スケジューラ処理時間はあまり変わらないことが分かる．また，BIO 処理時間が 0.005 秒～0.03 秒程度であるのに対し，IO スケジューラの処理時間はほとんどが 0.00003 秒程度であり，BIO 処理に占める割合は 1%未満であるため，BIO 処理やシステムコール処理に対する影響はほとんどないと考えられる．

2.2.3 節で説明したように，BIO 処理は，IO スケジューラの処理とデバイスの処理で構成されている．IO スケジューラの処理時間が変わらないにもかかわらず，BIO 処理時間が増えていることは，デバイスでの処理に時間がかかっていることを示している．デバイスの処理時間増加の原因として，フラグメンテーションによるディスクのシーク増加が考えられる．次に，このシーク動作について，LKST を用いて解析した．

3.4.　LKST Log Tools を使ったセクタ位置の差分解析

LKST により，アクセスするディスクのセクタ位置とアクセスサイズを記録し，解析を行った．I/O 処理が連続して行われるならば（シークがないならば）一回の I/O 要求でアクセスされるセクタの後尾（セクタ位置＋サイズ）と，続けて起きる I/O 要求でアクセスされるセクタの先頭との差分が 0 になるはずである．このセクタの先頭と後尾の情報は I/O 要求がデバイスドライバに渡される場所（elv_next_request()関数）で記録した．

セクタ位置の差分を図 3-7 から図 3-9 に重ねたものを図 3-12 から図 3-14 に示す．図中，セクタ位置の差分を diff. of sector に，その基準を右 Y 軸に示した．

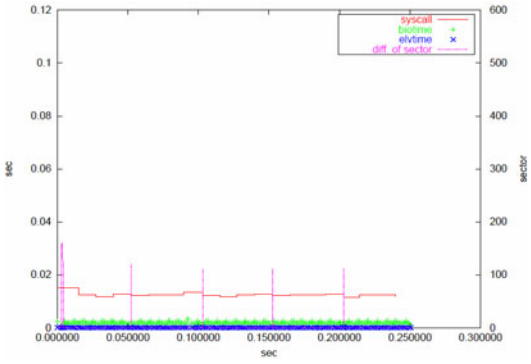

図 3-12 パターン A の処理時間とセクタ差分

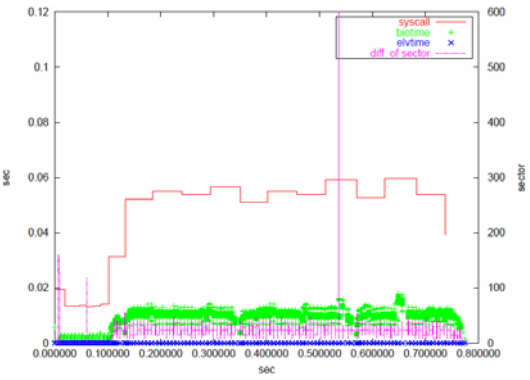

図 3-13　パターン C の処理時間とセクタ差分

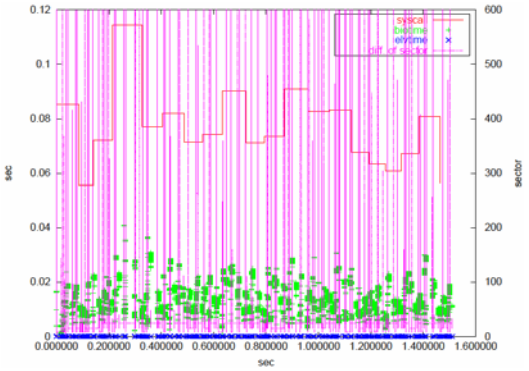

図 3-14 パターン E の処理時間とセクタ差分

この結果から，並列度が上がると，フラグメンテーションが発生している．すなわち，セクタ位置の差分が大きくなっていることがわかる．さらに，セクタ位置の差分の大きさの傾向は，BIO やシステムコールの性能低下の傾向と関連性があることが確認できた．

セクタ位置とブロック位置との間には，（1）式で表される関係がある．

$$P_B = (P_S - S_{MBR}) / R \qquad (1)$$

P_B：ブロック位置
P_S：セクタ位置
S_{MBR}：MBR を含むトラックのセクタ数
R：1 ブロックあたりのセクタ数

ディスクイメージとファイルシステムのパラメータを解析したところ MBR を含むトラックのサイズ（S_{MBR}）は 63 セクタであり，また 1 ブロックのサイズ（R）は 8 セクタであることがわかった．

（1）式に，この情報と，パターン E において LKST Log Tools が取得したセクタ位置（P_S）を代入し，ブロック位置（P_B）を得た．この計算結果と，3.2 において DAV で取得したファイルのブロック配置情報とを重ね合わせたものを，図 3-15 に示す．図中，Y 軸はディスク上のブロック位置を表し，X 軸はファイル上でのアクセス順（LKST Log Tools），あるいはブロックの配置順（DAV）である．

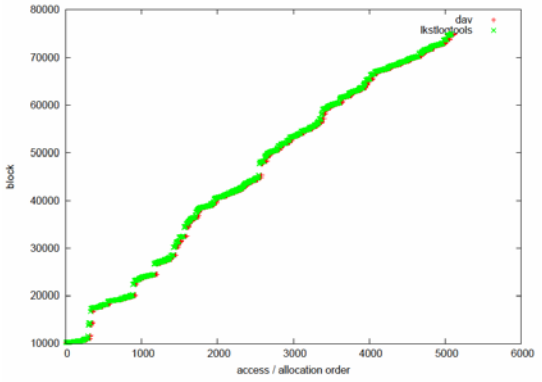

図 3-15 LKST Log Tools と DAV によるファイルのブロック配置情報（パターン E ）

この結果から，LKST Log Tools により取得したセクタ情報は，DAV で取得したブロック配置情報とほぼ同じであることが分かる．すなわち DAV の結果でフラグメンテーション状態がわかれば，今回の LKST の解析と照らし合わせることで，ファイル IO 性能の低下傾向をある程度予測できると考える．実際の利用方法としては，性能劣化が起きた時に，稼動サーバで DAV 情報を取り，性能劣化が起きるようなフラグメンテーションが発生しているかを確認する．フラグメンテーションが発生していた場合には解析用サーバで再現し，DAV の結果をヒントに LKST を用いて詳細な解析を進める，という使い方が挙げられる．

図中では X 軸方向にわずかなずれが生じているのが見て取れる．このずれについては，LKST を拡張しさらに解析を行う予定である．

4.　まとめ

フラグメンテーションによる性能低下を解析するため，ディスク上のフラグメンテーションの状態を静的に解析するツール DAV と，フラグメンテーション時のファイル I/O の動作状態をトレースし，解析するツール LKST Log Tools を開発した．

フラグメンテーションの起きる状況でこれらのツールを使い，並列書き込みを行った場合に急激

にフラグメンテーションが発生することを確認できた．また，フラグメンテーションによるシーケンシャルリード性能の低下の解析を行い，アクセスするセクタの差分が大きくなるとデバイスでの処理時間に影響が出ることを確認できた．さらに，DAV で取得するフラグメンテーション状態と LKST で取得したアクセスするセクタの差分とが同じ情報であることから，DAV の結果でファイル IO 性能の低下傾向をある程度予測できる可能性を得た．

今後さらに DAV と LKST の連携を図り，より詳細な解析を行うツールに強化していきたい．また，可視化・解析だけでなく，デフラグツールによるフラグメンテーションの適切な解消方法や，フラグメンテーションに耐性のあるブロック配置方法やブロック I/O 処理，ファイルのキャッシュアルゴリズムなどについても検討していく必要があると考える．

5. 謝辞

DAV と LKST の開発の一部は「独立行政法人 情報処理推進機構 オープンソースソフトウェア活用基盤整備事業」[11]に係る委託業務の調査・開発に基づくものです．ご支援いただいたことに感謝いたします．

参考文献

[1] http://davtools.sourceforge.net
[2] “「ディスク割り当て評価ツール「Disk Allocation Viewer」の開発」報告書”，http://www.ipa.go.jp/software/open/forum/Contents/DevInfraWG/dav-report.pdf
[3] http://lkst.sourceforge.net
[4] 杉田由美子，“Linux カーネル状態トレーサ LKST を使いこなせ！”, Software Design, 2003 年 4 月号
[5] 武田保真, “Linux サーバ定期健診　LKST によるカーネルチェック”,Software Design, 2003 年 2 月号
[6] 平松雅巳,“LKST（Linux Kernel State Tracer）を用いたカーネル動作の解析”, Linux カーネルシンポジウム in Yokohama Vol.5, http://www.osdl.jp/osdl/docs/ks5-200307/ks5_lkst.pdf
[7] “「LKST(Linux Kernel State Tracer)によるカーネル性能評価ツールの開発」報告書”，http://www.ipa.go.jp/software/open/forum/Contents/DevInfraWG/lkstprospec-0316.pdf
[8] http://fedora.redhat.com/
[9] http://www.kernel.org/
[10] http://iozone.org/
[11] 日本ＯＳＳ推進フォーラム 開発基盤ワーキンググループ http://www.ipa.go.jp/software/open/forum/DevInfraWG.html

商標