令和 4 年度 特別研究報告書

仮想化環境における
物理ディスクのセクタ位置を考慮した
I/O スケジューリング

龍谷大学 理工学部 情報メディア学科

学籍番号 ： T190539
氏名 ： 堀江 健斗

指導教員 ： 三好 力 教授，芝 公仁 助教

内容梗概

近年，計算機の消費電力の削減や設置スペースの削減などから仮想化が利用されるようになっている．しかし，計算機と比較して仮想計算機はI/O処理の性能が低い．ディスクへのアクセスを効率的に行うために，Linuxカーネルでは I/O スケジューリングを行っているが，仮想化環境の場合，十分な効果が得られない．本論文では，仮想化環境における物理ディスクを考慮した I/O スケジューリングを提案する．本手法を用いることにより，物理ディスクでのシーク時間を減らすことが可能になる．

目 次

1　はじめに

近年，計算機の消費電力の削減や設置スペースの削減，サーバ構築，運用の効率化を行うためにサーバ仮想化が利用されるようになっている．しかし，計算機と比較して仮想計算機での I/O 処理の性能は低いという問題がある．ディスクへのアクセスを効率的に行うために，Linux カーネルでは I/O スケジューリングを行っているが，仮想化環境の場合，十分な効果が得られない．

我々は，仮想化環境における物理ディスクのセクタ位置を考慮した I/O スケジューリングを提案する．本 I/O スケジューリングでは，一つの仮装計算機が I/O 処理を行う際に，物理ディスクでのシーク時間を減らすことを目的としている．シーク時間を減らすことで結果としてスループットの向上が期待できる．

以下，本論文では，2 章で関連研究について述べる．3 章で仮想化環境における I/O スケジューリングの問題点について述べ，4 章で問題を解決するための提案機構の構成を，5 章で提案機構の動作について述べる．6 章で評価を行い，提案システムの有用性を示す．

2　関連研究

2.1　仮想化

仮想化とは，コンピュータシステムを構成するリソース (CPU，メモリ，入出力装置など) を元の構成から独立させて，分割あるいは統合する形で仮想的に構成する技術である．OS(Operating System) を稼働させるハードウェアプラットフォームの仮想化では，ハイパーバイザ上に仮想マシンを構築し，その上でゲスト OS を稼働させる．

仮想化の実現方法の一つに KVM(Kernel-based Virtual Machine)+QEMU[1] がある．KVM とは，Linux カーネルに組み込まれたカーネルモジュールである．KVM を使うことで Linux をハイパーバイザ化させることができる．QEMU とはエミュレータ型の仮想化ソフトで CPU やメモリ，I/O デバイスなどを模擬する．KVM+QEMU の仮想化環境を図 1 に示す．KVM+QEMU では，QEMU が様々なデバイスをエミュレーションを行い，KVM が CPU の仮想化支援機能を利用することによって，動作を高速化させている．QEMU は Linux 上のプロセスとして動作しており，ゲスト OS から I/O デバイスへのアクセスを行うと，I/O デバイスの役割をしている QEMU が動作する．

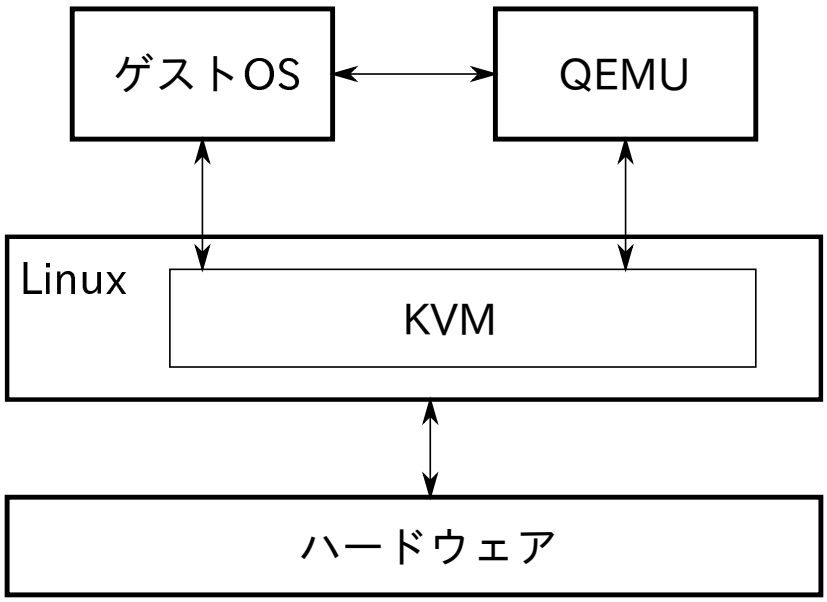

図 1　KVM+QEMU による仮想化環境

2.2 I/O スケジューラ

I/O 処理は，ユーザ空間のプロセスがカーネル空間の各機能を呼び出すことで行われる．図 2 に示すように，カーネルは階層的に各機能を呼び出して処理を行い，ディスクへのアクセスを行う．I/O スケジューラとは，カーネル内にある機能の一つでファイルシステムから送られてきた I/O 要求を処理してデバイスドライバに送っている．I/O 処理を行う際に，一番のボトルネックになっているのはディスクである．I/O スケジューラはディスクへの読み書きを効率的に行うために，上位層によって発行された I/O 要求に対して I/O スケジューリングを行っている．I/O スケジューラは複数あり，それぞれ異なるポリシーで動作する．ただし，同時に複数の I/O スケジューラを使用することはできない．I/O スケジューラは I/O 要求を一定期間キューに溜めて置き，マージ処理とソート処理を行う．マージ処理では，隣接したセクタ領域を持つ I/O 要求を一つの I/O 要求に統合する．ソート処理では，I/O スケジューラのポリシーにしたがって，I/O 要求の並び替えを行う．現在 Linux は None，MQ-deadline，Kyber，BFQ(Budget Fair Queueing) の 4 つの I/O スケジューラを持っている．None スケジューラは，ソー

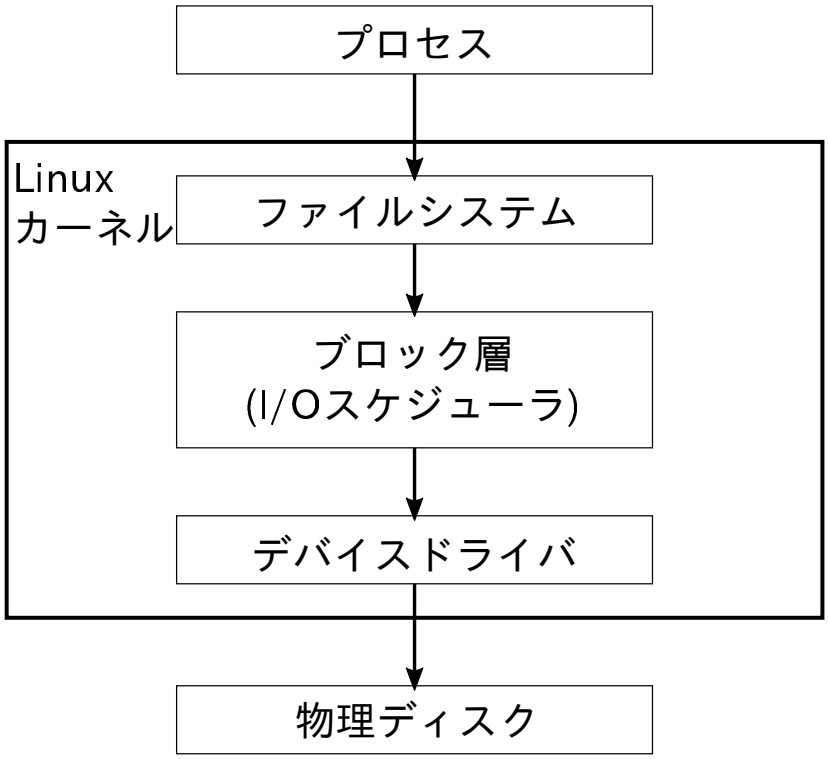

図 2　I/O 処理の概要

ト処理を行わないスケジューラである．ランダムアクセスが高速なデバイスに向いている．MQ-deadline スケジューラはレイテンシを一定に保つように I/O 要求をスケジューリングする．書き込み要求よりも読み込み要求を優先して処理する．様々な用途で使え，まんべんなく使いやすいスケジューラである．BFQ スケジューラは複数のプロセスが平等にブロックデバイスを使えるようにスケジューリングを行う．また，レイテンシの最小化を目的にしている．転送速度が遅いデバイスに向いている．反対に，転送速度が速いデバイスや低速な CPU の場合は，I/O スケジューリングでの処理が大きいため向いていない．Kyber スケジューラは読み取りと書き込みでそれぞれレイテンシを設定し，設定したレイテンシに間に合うように I/O 要求のスケジューリングを行う．SSD(Solid State Drive) などの低レイテンシデバイスに向いている．

I/O スケジューラの動作を図 3 に示す．I/O スケジューラはソフトウェアキューとハードウェアキューという 2 つのキューを持っている．ソフトウェアキューでは I/O スケジューリングを行う．ハードウェアキューではスケジューリングした I/O 要求をデバイスドライバへ送っている．キューの数は I/O スケジューラによって異なる．複数のキューを使うことにより，ロックが競合しないようにしている [2]．I/O スケジューラは I/O 要求をキューに溜めておく plug モードと，I/O 要求をデバイスドライバに送る unplug モードがある．上位層から I/O スケジューラに I/O 要求が送られてくると，I/O スケジューラは plug モードになりソフトウェアキューに I/O 要求を溜めて I/O ス

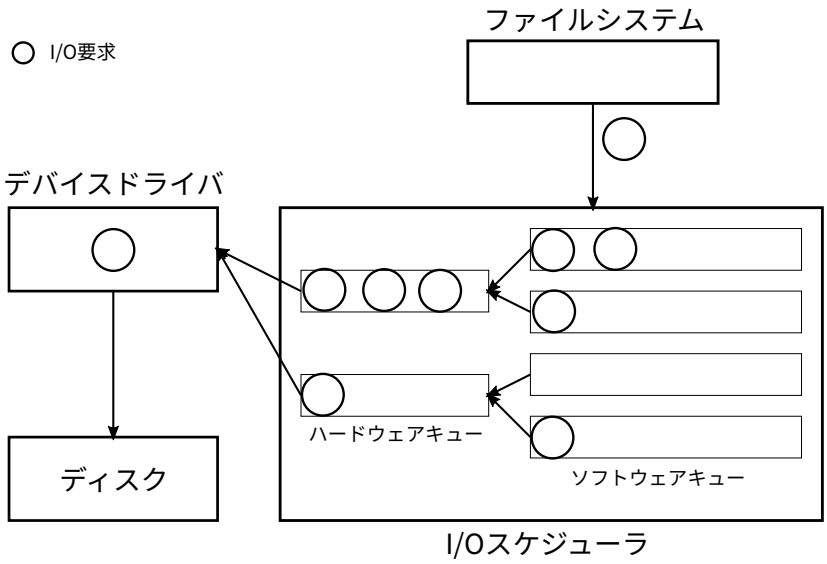

図 3　I/O スケジューラ動作

ケジューリングを行う．I/O スケジューラは I/O 要求がある程度ソフトウェアキューに溜まると unplug モードになり，ハードウェアキューに I/O 要求を送る．その後，I/O スケジューラはハードウェアキューにある I/O 要求をデバイスドライバに送る．

2.3 類似研究

類似研究では以下のような手法が提案されている．新居ら [3] の研究では，複数の仮想計算機が動作している状態において，それぞれの仮想計算機が持つディスクイメージファイル間のシーク時間を減らすことによって I/O 処理の高速化を実現するなどの試みが行われている．

3　仮装計算機におけるI/Oスケジューリングの問題点

2章で説明したように，Linuxカーネル内のI/Oスケジューラでは，ディスクでの読み書きを効率的に行うためにI/Oスケジューリングを行っている．HDD(Hard Disk Drive)のようにプラッタを回転させ，磁気ヘッドでデータの読み書きを行う記憶装置の場合，シーク時間やシーク距離を低減することで，結果としてスループットの向上やレイテンシの低減などといったことを可能にしている．しかし，仮想化環境におけるゲストカーネルのI/Oスケジューリングは，仮想ディスクに対して行われるために，実際にデータが保存されている物理ディスクに対して，十分な効果が得られないという問題がある．

ゲストOSに提供される仮想ディスクの物理ディスク内での状態を図4に示す．仮想ディスクの内容はホスト計算機のディスクイメージファイルに保持される．ディスクイメージファイルはバッキングファイルとスナップショットファイルから構成される．バッキングファイルはシステムの元となるデータを保存している．スナップショットファイルはバッキングファイルが持つデータに対しての変更点や新たに追加したデータを保存している．ゲストOSが仮想ディスクへアクセスするとき，バッキングファイルとスナップショットファイルへのアクセスの場合と，どちらか片方のファイルへのアクセスの場合がある．また，ゲストOSが連続した領域として仮想ディスクにアクセス

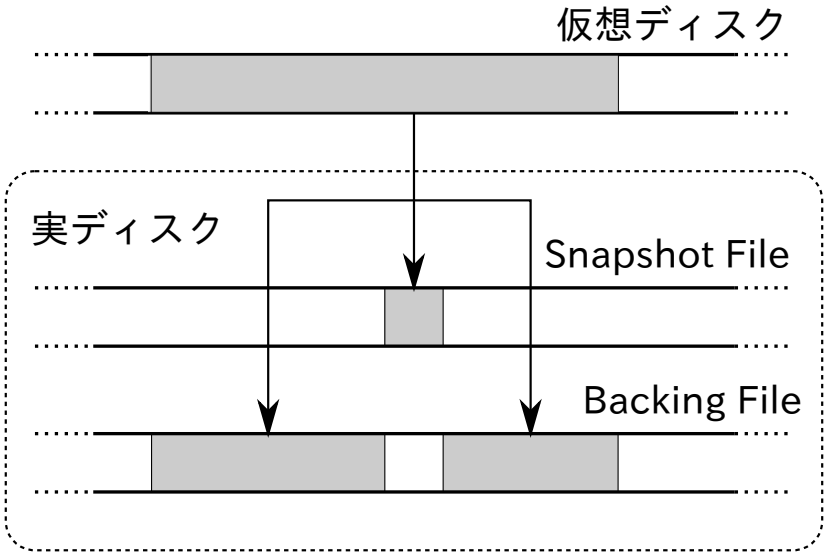

図4　仮想ディスクの物理ディスク内での状態

しても，物理ディスクでは不連続の複数の領域へのアクセスとなる．この場合，物理ディスクでのシーク時間やシーク距離を低減することができておらず，I/O スケジューリングを行わない場合とほとんど変わらない．したがって，仮想ディスクに対する I/O スケジューリングは，ディスクアクセス効率化として十分に機能していないという問題を抱えている．

4　提案機構の構成

提案機構は，バッキングファイルとスナップショットファイルが保存されている物理ディスクのセクタ位置を考慮したI/Oスケジューリングを行う．図5に示すように，同じファイル内のデータと異なるファイルのデータでは，同じファイル内のデータの方が近い位置のセクタに保存される可能性が高い．バッキングファイルとスナップショットファイルそれぞれに対して，I/O要求をまとめて発行することで効率的に読み書きすることを可能にする．提案機構は以下のようなポリシーを持つ．

- バッキングファイルへのI/O要求を優先的にデバイスドライバに送る．
- I/O要求をセクタ順にデバイスドライバに送る．
- 処理されていないI/O要求を無くすために I/O要求に期限を設定し，期限を過ぎたらデバイスドライバに送る．

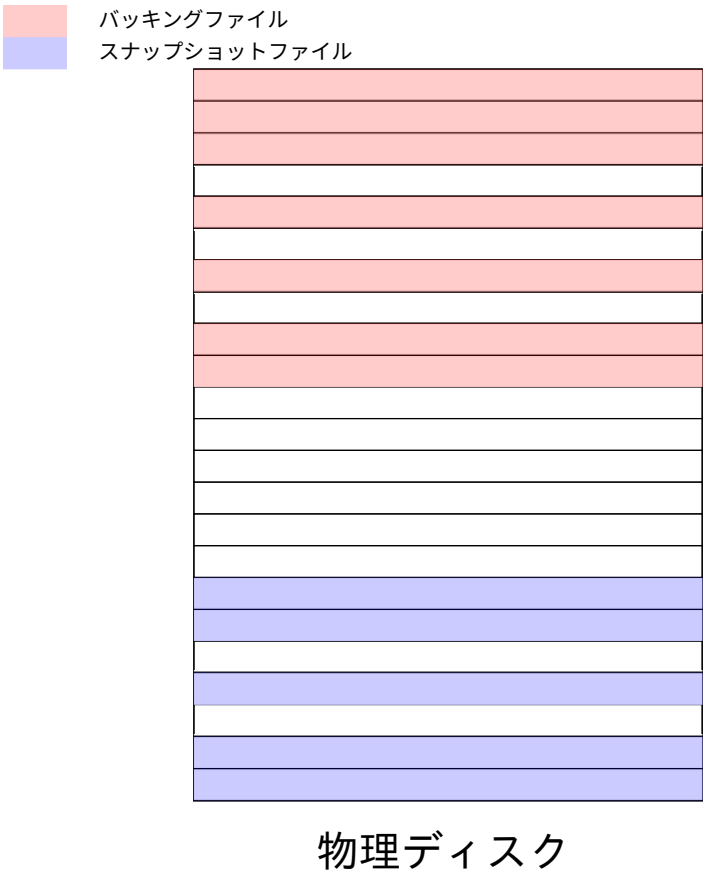

図 5　バッキングファイルとスナップショットファイルの保存されているセクタ位置

提案機構は，I/O スケジューリングを行うためのキューに加えて，書き込みがあったセクタの番号を保持するためにセクタ管理リストを持つ．書き込みがないセクタはバッキングファイルで，書き込みがあったセクタはスナップショットファイルで管理されていると区別する．本機構を Linux カーネルが持つ MQ-deadline を拡張し，実装した．

5　提案機構の動作

図 6 に示すように，本機構は以下の手順でバッキングファイルとスナップショットファイルのどちらへのアクセスかを考慮した I/O スケジューリングを行う．

(1) 仮装計算機上で I/O プロセスを発行する．
(2) I/O スケジューラは書き込み要求の場合，セクタ管理リスト内のセクタに書き込みがあったことを記録する．
(3) I/O スケジューラは I/O 要求が持つセクタに書き込みがあったかセクタ管理リストに確認し，I/O 要求に記録する．
(4) I/O スケジューラは書き込みの有無とセクタ番号から I/O 要求をキューに入れる位置を決める．その後，I/O 要求に期限を設定し，キューに入れる．
(5) I/O スケジューラは I/O 要求をデバイスドライバに送る．

(4) において，キューに入れる位置はバッキングファイル側かスナップショットファイル側かを決めた後，セクタ番号順に I/O 要求が並ぶように入れる．これによって，I/O 要求の発行は，バッキングファイルに対しての要求の中でセクタ番号が小さいものに対して優先的に行なわれる．バッキングファイルに対しての I/O 要求をすべてデバイスドライバに送った後，スナップショットファイルに対しての I/O 要求をデバイスドラ

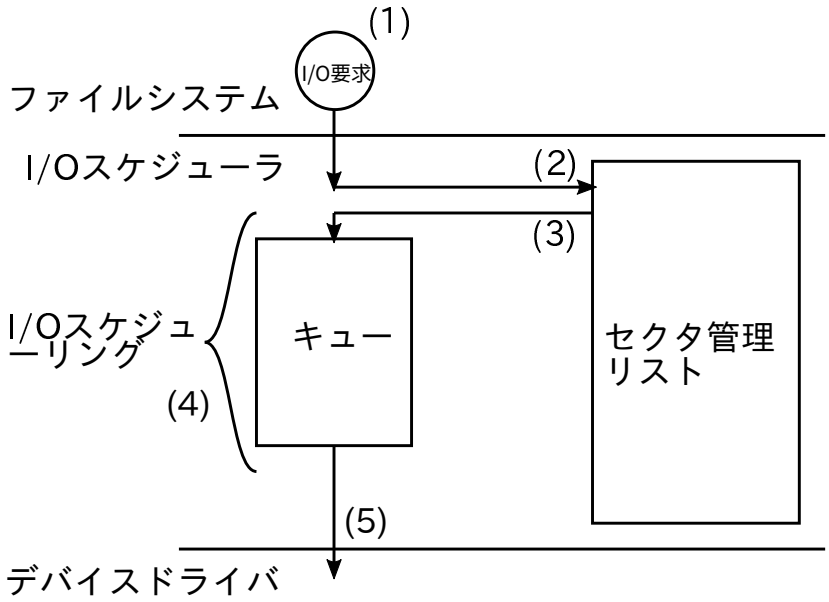

図 6　提案機構の動作

イバに送る．デバイスドライバにI/O要求を送るときに，期限が切れたI/O要求がある場合は優先的にそのI/O要求をデバイスドライバに送る．期限切れのI/O要求が複数ある場合，期限が切れた順にI/O要求をデバイスドライバに送る．

このようなI/Oスケジューリングを行うことによって，物理ディスクでのバッキングファイルからスナップショットファイルへのシーク回数を減らすことができ，結果としてスループットの向上が期待できる．

5.1 動作の具体例

提案機構のI/Oスケジューリングの具体例を図7に示す．キューの中にセクタ番号が221，231，267のバッキングファイルへのI/O要求とセクタ番号が222，300，315のスナップショットファイルへのI/O要求がある．新たにセクタ番号が310のスナップショットファイルへのI/O要求をキューに入れるとき，その位置は，スナップショットファイル側のI/O要求のセクタ番号が300と315の間になる．

提案機構がデバイスドライバにI/O要求を送る具体例を図8に示す．キューの中にセクタ番号221，231，267のバッキングファイルへのI/O要求とセクタ番号が222，300，310，315のスナップショットファイルへのI/O要求がある．キューに期限切れのI/O要求がない場合，I/Oスケジューラはバッキングファイルへの I/O要求をセクタ番号が211，231，267の順でデバイスドライバに送った後，スナップショットファイルへのI/O要求をセクタ番号が300，310，315の順でデバイスドライバに送る．バッキングファイルへのI/O要求でセクタ番号が267のI/O要求とスナップショットファイルへのI/O要求でセクタ番号が310のI/O要求が期限切れている場合，先に2つのI/O要求を処理する．スナップショットファイルへのI/O要求が先に期限が切れていた場合，

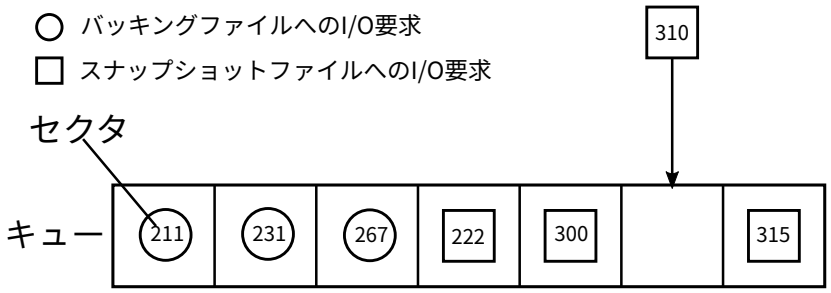

図7　I/Oスケジューリングの具体例

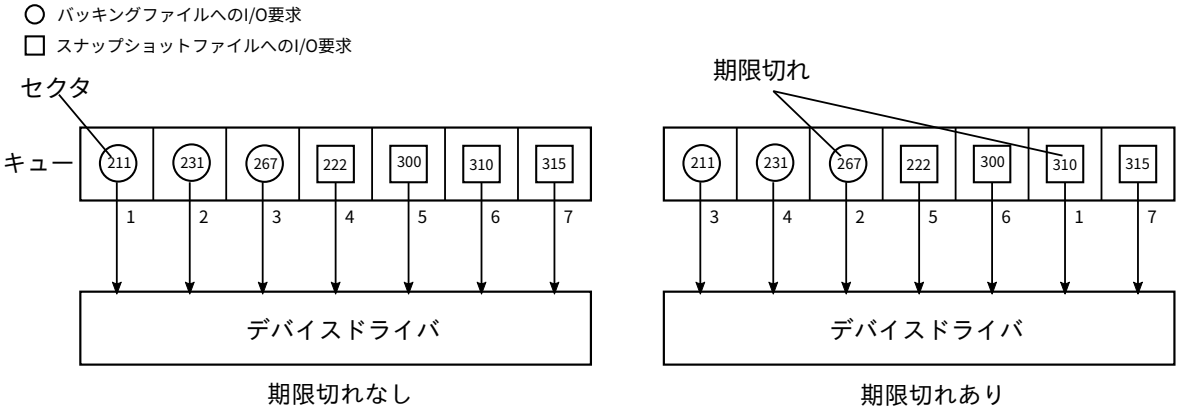

図 8　デバイスドライバに I/O 要求を送る処理の具体例

I/O スケジューラはセクタ番号が 310 のスナップショットファイルへの I/O 要求，セクタ番号が 267 のバッキングファイルへの I/O 要求の順で I/O 要求をデバイスドライバに送る．残りの I/O 要求は期限切れがない場合と同様の順でデバイスドライバに送る．

6　評価

実験環境の物理計算機側の仕様を表 1，仮想計算機側の仕様を表 2 に示す．

提案する I/O スケジューラの性能を測定するためにベンチマークソフトの fio を用いて，I/O 処理速度を測定する．測定は本 I/O スケジューラが最も苦手とする，すべての I/O 要求がバッキングファイルに対して発行される場合の読み込み速度で行う．また，比較対象として仮想計算機で一番オーソドックスな None スケジューラの読み込み速度も測定する．測定は 8 つのスレッドがそれぞれ 4kB ごとに合計 100MB のファイルを読み込む速度を測定する．この処理を，提案する I/O スケジューラと None スケジューラで 5 回行った結果の平均を比べる．測定結果を表 3 に示す．読み込み速度は None スケジューラが平均で 271MB/sec，提案スケジューラが平均で 209MB/sec という結果になった．提案スケジューラが苦手な処理の場合は従来の I/O スケジューラと比較して読み込み速度が約 25%遅くなっている．

表 1　物理計算機側実験環境

CPU	Core i5-12400
CPU Core	12
OS	Ubuntu 22.04.1 LTS
メモリ	32GB
カーネル	5.15.0-58-generic
HDD	1TB

表 2　仮想計算機側実験環境

CPU Core	2
OS	Debian-11.1
メモリ	4GB
カーネル	5.15.44
Strage Size	20GB

表 3　fio による読み込み速度の結果

	None スケジューラ	提案スケジューラ
1 回目	293MB/sec	180MB/sec
2 回目	205MB/sec	227MB/sec
3 回目	382MB/sec	215MB/sec
4 回目	216MB/sec	166MB/sec
5 回目	260MB/sec	259MB/sec
平均	271MB/sec	209MB/sec

読み込み速度が 25%も遅くなってしまった原因として，I/O スケジューリングの処理に時間を取られすぎていることが考えられる．セクタ番号を検索するという処理が多大な時間を必要としている可能性があるので，検索時間の削減ができないか検討していく．

7　おわりに

本稿では，仮想化環境における物理ディスクのセクタ位置を考慮したI/Oスケジューリングについて述べた．本機構は，ゲストカーネルが仮想ディスクのセクタに書き込みがあったかどうか確認することによって，バッキングファイルとスナップショットファイルを区別したI/Oスケジューリングを行う．これにより，仮想ディスクでのデータの読み書きを効率的に行うことが可能になる．

謝辞

本研究を行うにあたり，温かいご指導，ご鞭撻を頂いた芝公仁助教授，三好力教授に感謝します．また，日々の研究で貴重な意見をくださった芝研究室の皆様に感謝します．

参考文献

[1] : QEMU. https://www.qemu.org/.

[2] Bjørling, M., Axboe, J., Nellans, D. and Bonnet, P.: Linux Block IO: Introducing Multi-Queue SSD Access on Multi-Core Systems, *Proceedings of the 6th International Systems and Storage Conference*, SYSTOR '13, New York, NY, USA, Association for Computing Machinery (2013).

[3] 新居健一，山口実靖：仮想化環境における I/O スケジューラの動作解析，第 73 回全国大会講演論文集，Vol. 2011, No. 1, pp. 191–192 (2011).