情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2011-ARC-195 No.12
Vol.2011-OS-117 No.12
2011/4/13

論文

VMMによるSSDのランダム書き込み性能の改善

Improving Random Write Performance of SSD using VMM

Vol.2011-ARC-195 No.12
Vol.2011-OS-117 No.12
2011/4/13

概要

近年，ストレージデバイスとしてSSD（Solid-State Drive）の需要が高まっている．SSDは一般的に，ランダム書き込み（ランダムライト）性能が低いという特徴をもつ．この原因は，SSDの構造上の問題として，小サイズのデータの書き換えの為に古いデータの退避と削除，新しいデータの書き込みという複雑な処理を要することにある．SSDのランダムライト性能改善に関する既存研究として，SSD内のNANDコントローラを改良する方法が存在するが，この方法では必然的にハードウェアの性能に依存することになる．また，OS側でデバイスドライバレベルにより改善する方法が提案されているが，こちらはOSの修正を要することになる．

そこで本研究では仮想マシンモニタ（VMM）を用いて，ハードウェアの性能に依存せず，かつゲストOSの修正を要さずにSSDのランダムライト性能を向上する手法を提案する．具体的には，VMMによってSSDに書き込まれるデータを連続的な配置に変換し，ランダムライトをシーケンシャルライトに変換する．本研究では提案システムの設計と実装，そして評価実験を行い，提案システムによってSSDのランダムライト性能が最大で8倍程度改善される事を確認した．

Abstract

Solid state drives have become popularly storage during recent years. However, they have a feature poor random write performance cause flash memory restriction. This paper introduces a system improving random write performance of SSD using VMM which isn't depend hardware implemention and operating systems. Our approach is converting random write to sequential write to improve performance. We implemented this system on Hypervisor's device driver. The experimental results shows this system improves in the maximum about 8 times enhanced random write performance.

Vol.2011-ARC-195 No.12
Vol.2011-OS-117 No.12
2011/4/13

1. はじめに

SSD（Solid-State Drive）はNANDフラッシュメモリを用いたストレージであり，ランダム読み込み（ランダムリード）性能がハードディスクのそれより非常に高いという利点が存在する.[1] しかしSSDには，ランダムライト性能がシーケンシャルライト性能と比較して低いという特徴がある．ランダムライトの性能低下の理由は，シーケンシャルライトと比較し書き込み対象では無いデータに対しての内容の退避，消去，新規書き込みといった複雑な操作を要することにある．

そこで本研究では仮想マシンモニタを用いて，ランダムライトをシーケンシャルライトに変換することで，オーバーヘッドの大きいランダムライトの直接的な実行を避ける事により高速化を実現する．具体的には，SSDに対してランダムライトが発生した際に，データの書き込み先アドレスを書き換えることにより，実際に書き込まれるデータの配置を連続的な配置に変換する．仮想マシンモニタを用いて性能改善を行うことによって，SSD内部のNANDコントローラを改善するような，ハードウェア依存の手法とは独立して性能を向上することができる．合わせて，デバイスドライバの追加といったOS側への修正を要さず，かつOSの種類に依存することなくランダムライト性能の向上を図ることが可能となる．

本研究では提案システムを実現する為，当研究室が中心となって開発しているType I型仮想マシンモニタのBitVisor[2]を用いた．BitVisorは，準パススルーアーキテクチャに従い設計された，オーバーヘッドを最小限に留めた仮想マシンモニタである．本システムは，BitVisorのATA準パススルードライバに機能追加をする事によって実装を行った．

提案システムの効果を検証する為，Iozone[3]を用いて，提案システムによってSSDのランダムライト性能を測定した．提案システムによるランダムライト性能改善の効果を測定するため，本システムを用いる場合と用いない場合に分けて，性能測定を行った．実験結果として提案システムを導入した場合，小データサイズのランダムライトが高速化し，SSDのハードウェア性能によってはOSを直接起動した場合の性能と比較し最大8倍程度の高速化を実現した．

Vol.2011-ARC-195 No.12
Vol.2011-OS-117 No.12
2011/4/13

2. 背　　景

SSD（Solid-State Drive）は，近年ハードディスクに代わるストレージデバイスとして注目されている，記憶領域にNAND型フラッシュメモリを用いたストレージデバイスである．

SSDのランダムライト性能の低さの原因は，SSDにおいて小サイズのデータの書き換え処理のオーバーヘッドが大きいことにある．SSDにおいて，既に存在するデータに対して修正を行う際に，上書き処理を行う事が出来ない．その為，既存データに対する書き換えが発生した際には一旦古いデータを全て消去してから新しいデータを書き込む必要がある．合わせて，SSDに対する読み込みと書き込みは，ページと呼ばれる4kB程度の領域単位で行われる．対して，消去処理はSSD上でブロックという単位で処理され，このサイズが512kB程度となっており，書き込みの単位であるページと比較して非常に大きい．この為，小サイズのデータの書き込み処理を複数回発行するようなランダムライト実行時には，無駄なデータの退避，削除，書き込みを行う領域の割合が増加するため，書き込みの為に発生するオーバーヘッドが大きい．

SSDのシーケンシャルライト性能はランダムライト性能に比べて高い．これは，シーケンシャルライトにおいて書き換え処理を行う場合でも，ある程度まとまった領域を書き換える事になる為，オーバーヘッドの割合が小さいことによる．よって，特にオーバーヘッドの割合の大きい，書き込みデータのサイズが小さなランダムライトの実行は，シーケンシャルライトの実行に変換する事によって高速化が可能といえる．

3. VMMによるSSDの書き込みデータ配置変換

3.1 提案システムの設計

本研究ではVMMを用いてSSDのランダムライト性能を向上する．VMMを用いて性能改善を行うことによって，ハードウェアの性能に依存せず，またデバイスドライバの追加のようなOSへの修正を要さず，OSの種類にも依存せずにSSDの書き込み性能の向上を行う事が出来る．

本システムの概要を**図1**に示す．本システムはVMMのデバイスドライバに機能追加を行い，ゲストOSからの書き込み処理を検査する．発生した書き込みがランダムライトであると判断できる場合は，書き込みデータの配置を変換

Vol.2011-ARC-195 No.12
Vol.2011-OS-117 No.12
2011/4/13

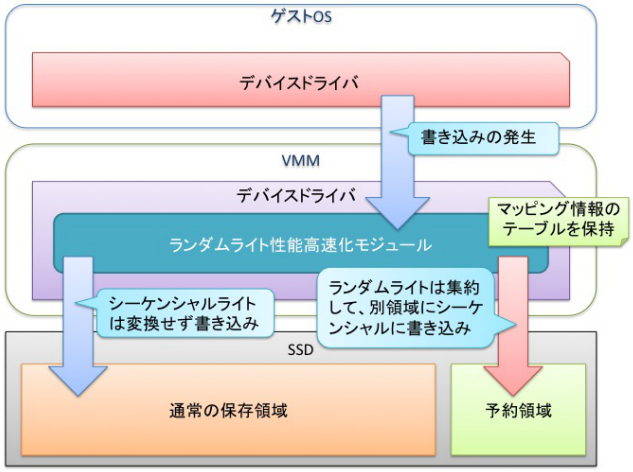

図 1　提案システムの概要

Fig. 1　System model.

することによってシーケンシャルライトへの変換を行う．これによりオーバーヘッドの大きなランダムライト処理の実行を避ける事で，高速化を実現する．また，シーケンシャルライトへ配置変換したデータは，SSD 上に予め予約領域という特別な領域を用意して，そこに対して書き込みを行う．予約領域は，ゲスト OS の制御する領域とは独立させる．これは配置変換を行ったデータがゲスト OS によって誤って破壊される事を防ぐ為である．

なおデータの配置変換時に，配置変換前の本来の書き込み先アドレスと，配置変換後のデータを参照する為の情報をマッピングテーブルに記録する．この情報は，配置変換を行ったデータに対して読み込み処理が要求された際に，正常に配置変換後データを読み込む為に使用する．マッピングテーブルは高速な書き換え処理を行う為にメモリ上に保持しておくが，システムのリブート後も参照可能にする為に，システム終了前に SSD への記録を行う．マッピングテーブルの記録に使用する記憶領域も，データの破壊を防ぐ為に予約領域を使用する．

3.2　書き込みデータ配置変換による SSD のランダムライト性能向上

本研究で提案する SSD のランダムライト性能高速化手法は，実際にランダムライト性能の向上を行う書き込みデータ配置変換処理と，配置変換されたデータの検出と追跡に用いるマッピングテーブルの管理によって実現する．

3.2.1　書き込みデータの配置変換

SSD において，ランダムライトの実行は，シーケンシャルライトの実行と比

Vol.2011-ARC-195 No.12
Vol.2011-OS-117 No.12
2011/4/13

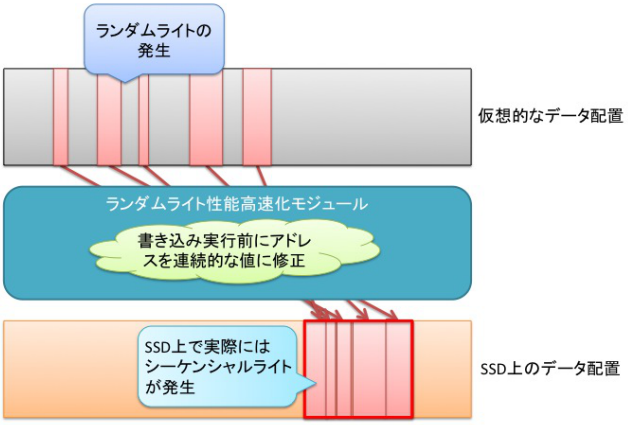

図 2　書き込みデータの配置変換

Fig. 2　Converting Arrangement of write data.

較しオーバーヘッドが大きい．そこで本研究では，ランダムライトの実行を回避することによって書き込み性能の向上をねらう．

ランダムライトの検出は，書き込み対象となるデータのサイズによって判断する．書き込みデータサイズが十分大きければ，SSD 上では連続した領域に対するデータ書き込みが行われ，シーケンシャルライトが発生すると判断できる．対して，書き込みデータサイズが小さければランダムライトが発生している可能性が高いため，配置変換による書き込み性能の向上が期待できる．

実際にランダムライトが発行された際に，SSD に対して実際にはシーケンシャルライトが発生するように修正を行う．これよりランダムライト発生時の，無駄な書き換え処理回数が削減されるので，ランダムライト処理の為のオーバーヘッドを抑制できる．具体的には，**図 2** に示すように，書き込みデータ配置を仮想的なものと見なし，実際に SSD に書き込まれるデータの配置を変換する．データ配置の変換は書き込み先のアドレスを変更することで実現する．ランダムライトが発生した際に，データの書き込み先アドレスの値を連続的な値に修正し，書き込まれるデータの位置を集約することで，SSD 上で実際に発生する処理はシーケンシャルライトとなる．配置変換を行った際に，配置変換されたデータに対する読み込み処理を正常に行う為の情報をマッピングテーブルに記録する．

3.2.2 マッピングテーブルの管理

配置変換の対象となったデータを含むような読み込み処理が要求された際は，最新のデータを正常に読み込む事が出来るよう配置変換後のデータを追跡する．その為に本システムでは，配置変換後のデータを参照する為の情報を格納するマッピングテーブルを持つ．マッピングテーブルに格納される値は，キーとして利用される本来書き込まれるはずだった元のアドレス，そして配置変換されたデータの存在する移動先のアドレスとそのデータのサイズを示す値の三種類である．

本システムにおいて読み込み処理が発生した場合，読み込み対象となる領域のアドレスと一致するマッピングテーブルの値が無いかどうか検査する．もし一致する値が存在する場合，最新のデータは配置変換先アドレスに存在する事になるので，別途読み込む必要がある．従ってマッピングテーブルから移動先のアドレスと移動したデータのサイズを取得し，別途読み込みを行う事で正常な読み込み処理を行う．

また，新しいデータの上書きが発生した場合，配置変換されたデータが不要になるので，そのデータを追跡する為の情報をマッピングテーブルから削除する．この処理は書き込み処理の対象となる領域内に，既に配置変換されたデータが存在するかどうかマッピングテーブルを用いて検査し，存在した場合は該当する情報をマッピングテーブルから削除する事で行う．

4. 実　　装

当研究では，BitVisor 1.0.1 の ATA 準パススルードライバに修正を行う事で，SSD のランダムライト性能改善システムを実装した．

本システムの制約として，BitVisor を利用して実装していることにより，同時に実行できるゲスト OS は一つに限定される点が存在する．また現状では配置変換後に不要になったデータを回収するガベージコレクションを実装していないので，高速化の為に要求される SSD の記憶領域は増加し続けるという問題点も存在する．

4.1 BitVisor の概要

BitVisor[2] は，当研究室が中心となって開発している，セキュリティ強化の為のハイパーバイザ型仮想マシンモニタである．BitVisor は，準パススルー型アーキテクチャに従い設計されており，小サイズの仮想マシンモニタとなって

いる．

4.1.1 準パススルー型アーキテクチャ

準パススルーアーキテクチャは，基本的にゲスト OS に直接ハードウェアにアクセスすることを許可し，ATA デバイスや NIC などへのアクセスのような一部の限られた I/O に対してはフックを行う，一部の I/O に対してのみ仮想化を行うアーキテクチャである．グラフィックデバイスやサウンドデバイスなどへの，監視する必要の無いデバイスアクセスをパススルーすることで，オーバーヘッドの減少を実現している．

BitVisor は，セキュリティの確保に必要な最低限の I/O を監視する．監視される対象となるのは，ゲスト OS とデバイスのデータ転送を制御する為のコントロール I/O と，実際のデータ転送の際に発生するデータ I/O である．ハイパーバイザはコントロール I/O の内容を確認し，許可すべきでない操作が下ってきた場合にはこれをブロックする．また，データ I/O に関しては BitVisor による制御を可能にし，必要に応じてデータの暗号化などを可能としている．

4.1.2 準パススルードライバ

アクセス制御のために監視すべき I/O は，実際に発行される全体の I/O の一部である．BitVisor のもつドライバは，一部の I/O のみフックしアクセス制御の為のチェックを行い，監視の必要の無い I/O にはパススルーする．結果，実際に仮想化されるのはデバイス全体では無く，デバイスの限られた一部となる．このような機構を持つドライバを，準パススルードライバという．準パススルードライバは，監視するの監視する必要の無い I/O をパススルーすることによりドライバサイズを小さくし，オーバーヘッドを最小限に抑えている．

当研究で提案するシステムは，BitVisor でサポートされている ATA 準パススルードライバに対して，修正を行うことで実現する．ATA 準パススルードライバでは，指定の ATA コマンドがゲスト OS によって発行された際にフックし，ATA レジスタに書き込まれた値の参照と書き換えが可能である．

4.1.3 Shadow DMA Descriptor

DMA 転送は，CPU の介入無くデータ転送を行うため，そのままでは VMM による転送対象のデータの制御が不可能である．このため BitVisor では，DMA 転送によってゲスト OS とデバイスがやり取りするデータを制御する為，Shadow DMA Descriptor という機構が導入されている．ゲスト OS が用意した，データ転送先のバッファ（Guest Buffer）とバッファのアドレスとサイズを表す PRD

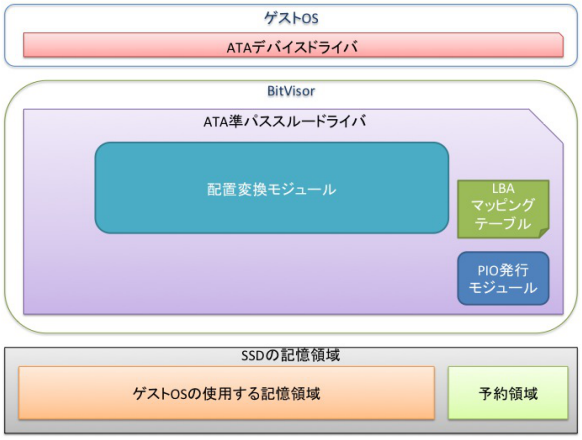

図 3　本システムの概略図

Fig. 3　Detail of System.

（Guest DMA Descriptor）とは別に，BitVisor はそのメモリ空間に Shadow DMA Descriptor と Shadow Buffer を用意する．これにより，実際にデバイスから DMA 転送される転送先を Shadow Buffer とすることによって，BitVisor によるデータ転送の制御を可能としている．Shadow Buffer に転送されたデータは，BitVisor によって Guest Buffer にコピーされ，データ転送を終了する．

4.2　本システムの概要

実装したシステムは，配置変換モジュール，LBA マッピングテーブル，PIO 発行モジュールといったコンポーネントから構成される．概略図を**図 3** に示す．

配置変換モジュールは本システムによる高速化の要となる，ランダムライトをシーケンシャルライトに変換するモジュールである．LBA マッピングテーブルは，配置変換の際にマッピング情報を記録し再アクセスを可能にする．PIO 発行モジュールは，LBA マッピングテーブルの SSD への記録と読み出しに使用される．

4.3　配置変換モジュール

配置変換モジュールが行う処理を図**図 4** に示す．配置変換モジュールは，ATA 準パススルードライバによってフックしたディスク I/O を検査し，小サイズのデータの書き込みかどうか判定する．データサイズの大きな書き込みに対しては，配置変換はせずに通常通りの書き込み処理を行う．対してデータサイズの小さな書き込みが発生した場合，書き込み先 LBA を書き換えることにより，連続的なデータ配置への変換を実行する．また，配置変換時に LBA マッピング

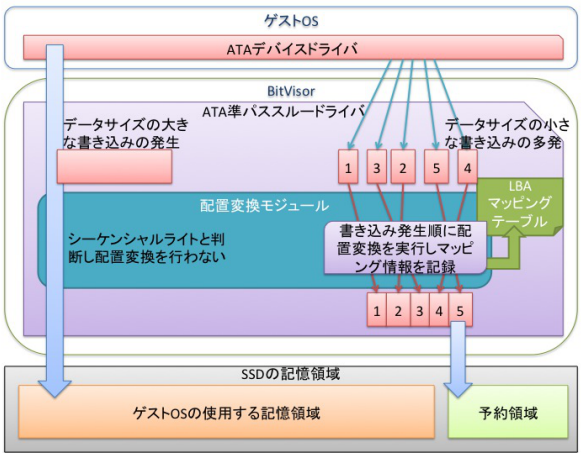

図 4　配置変換モジュールの概要

Fig. 4　Data arrangement conversion module.

テーブルへの新たなマッピング情報の記録を行い，配置変換されたデータへの読み込み処理が発生した際に配置変換先 LBA とデータサイズを取得出来るようにする．

4.3.1　小サイズランダムライトの検出と配置変換

ゲスト OS から ATA コマンドが発行された際に，BitVisor の ATA 準パススルードライバは I/O をフックする．この時，発行された ATA コマンドの内容を参照すれば，書き込みが発生するか否かが判別できる．

配置変換モジュールでは書き込みが発生した際に，ゲスト OS が設定した ATA レジスタの LBA と，書き込み対象データのセクタ長を表すセクタカウントの値を確認する．実際に配置変換するかどうかは，セクタカウントの値をチェックすることで判断する．もしセクタカウントが十分に大きければ，SSD 内部ではシーケンシャルライトが発生するため，配置変換は行わなくてよい．逆にセクタカウントが小さい値の書き込みは，ランダムライトが発生していると判断し配置変換処理を行う．ここでは，SSD の読み込み・書き込み処理単位であるページのサイズに合わせ，書き込みデータサイズが 4kB 以下，つまりセクタカウントの値が 8 以下の書き込み処理に対して配置変換を行う．書き込み処理をフックした際に，ゲスト OS のデバイスドライバによって設定されたセクタカウントの値が 8 以下であるかどうかを判定する事で，ランダムライト検出を実現した．

セクタカウントが 8 より大きい値である場合は，配置変換モジュールによる配

置変換は行わず，ゲストOSが設定したLBAに対して書き込みを発行する．対して，セクタカウントが8以下の小データサイズの書き込みが発生した場合は，ゲストOSによって設定されたLBAの値が連続的な値になるようATAレジスタに新しいLBAをセットし，必要であればATAコマンドの書き換えを行って書き込み処理を実行する．この際，LBAマッピングテーブルに配置変換前のLBAと配置変換後のLBA，書き込みデータのセクタカウントを新たに記録する．

4.3.2 ATAコマンドの書き換え

フックしたランダムライトの書き込み先のLBAが，28ビットで表現可能な範囲である場合，ゲストOSによって発行されるATAコマンドは未拡張のものである可能性がある．配置変換モジュールでは，配置変換する先のLBAが28ビットで表現不可能な領域でも正常に書き込みが行えるよう，拡張されたATAコマンドを発行するようにする．その為当モジュールでは，フックしたI/OのATAコマンドを検査し，もし未拡張のATAコマンドである場合，拡張されたATAコマンドに変換する．

4.4 LBAマッピングテーブル

LBAマッピングテーブルは，配置変換のマッピング情報を格納・管理する．

配置変換を行ったデータは，ゲストOSによって再アクセスが要求された際に参照出来るようにしなければならない．従って本システムではデータ配置変換時に，本来書き込まれるはずだったLBAの値をキーとして，配置変換後のデータのLBAとセクタカウントを参照可能にする為のマッピング情報を記録する．

LBAマッピングテーブルは，下の三種類の値を持つ．48ビットLBAによる参照が可能なよう，LBAに関しては64ビット，セクタカウントに関しては32ビットの範囲の値が格納可能となっている．

本来のLBA

配置変換が行われる前のLBAを表す．64ビットの範囲の値を格納可能．マッピングテーブルのキーとして使用する．

配置変換先LBA

配置変換されたデータが格納されているアドレスを表す．64ビットの範囲の値を格納可能．本システムで配置変換されたデータの追加読み込みを行う際に指定するLBAの値となる．

配置変換済データのセクタ数

配置変換されたデータの長さを表す．32ビットの範囲の値を格納可能．本

システムで配置変換されたデータの追加読み込みを行う際に指定するセクタカウントの値となる．

4.4.1 マッピングテーブルの読み込みと記録

LBA マッピングテーブルは，システム動作中はメモリ上に保持する．しかし，システムの再起動後にも配置変換済データへのアクセスが発生した際には，追加の読み込み処理を行わなければならない為，LBA マッピングテーブルを参照可能にする必要がある．その為，当システムではゲスト OS の終了後，VMM を終了する前に，マッピングテーブルのサイズや配置変換用インデックス値といったメタ情報と，マッピングテーブルの内容を SSD 上の特定領域に書き込むことでマッピング情報の消失を防ぐ．システム起動時はまずマッピングテーブルのメタ情報を捜索し，メタ情報が存在する場合 LBA マッピングテーブルが存在すると判別し，保存されたマッピングテーブルを読み込みメモリ上に保持する．

4.4.2 配置変換の追跡

ゲスト OS から読み込みが発生した際に，読み込むべき領域に対して，マッピングテーブルを用いて配置変換されたデータが無いかどうかの検査を行う．この検査は，読み込み先 LBA からセクタカウント分の領域内を示す LBA と一致する値が，マッピングテーブルのキー値として格納されていないかチェックする事で行う．もし配置変換されているデータが含まれている場合は，マッピングテーブルから配置変換済データの存在する LBA と，そのセクタカウントを取得し，VMM から別途読み込み処理を行う．読み込んだ配置変換済データを用いて，ゲスト OS に渡すデータを修正することで配置変換済データの追跡を完了する．

本システムでは，配置変換済みデータの内容を反映する為に，BitVisor の Shadow DMA Descriptor 機構における，Shadow Buffer に対する修正処理を実装した．読み込みの対象領域に配置変換済みデータが存在する場合は，まずゲスト OS が想定する領域に対する読み込みを実行し，Shadow Buffer にその内容を転送する．その後，配置変換済データをマッピングテーブルを用いて追加で読み込み，Shadow Buffer に対して別途読み込んだデータを上書きした後で，ゲスト OS 側に用意された Guest Buffer に Shadow Buffer の中身をコピーする事によって配置変換の追跡を実現した．

4.4.3 マッピングテーブルの管理

ゲスト OS から書き込みが発生した際に，既に配置変換を行ったデータが上

書きされ，不要となる場合がある．その場合には，該当する配置変換済データの追跡を行う必要が無くなる為，マッピング情報を破棄する．

ゲスト OS からの書き込み処理が発生した場合，本システムは書き込み先の領域に対して配置変換されたデータが存在しないかマッピングテーブルを用いて検査する．この検査は配置変換済データの追跡と同様，書き込み先 LBA からセクタカウント分の領域内を示す LBA と一致する値が，マッピングテーブルのキー値として格納されていないかチェックする事で行う．もし，書き込まれる新しいデータが配置変換済の領域に対して行われる場合は，配置変換された古いデータは不要となる．そのため，そのデータを追跡する為のマッピングテーブルのキー値を 0 に設定することで，不要となったマッピング情報のクリアを行う．

4.5 PIO 発行モジュール

BitVisor は受動的な VMM であり，ゲスト OS から I/O が発行された際に処理を行うが，BitVisor から自発的に I/O を発行することはない．しかし本研究で提案するシステムでは，設定データをシステムの再起動後でも使用出来るようストレージ上に記録しておく必要がある．

この為，本システムでは PIO 発行モジュールを用意し，READ SECTOR（s），WRITE SECTOR（s）コマンドによる，BitVisor からの自発的な I/O の発行を可能にしている．このモジュールは LBA マッピングテーブルの読み込みと記録，配置変換済データの追加読み込みに使用する．

4.6 予 約 領 域

本システムでは，VMM によって制御されるデータを保存する領域を用意する．この領域は，データが破壊されないようゲスト OS が使用する領域と分離し，LBA マッピングテーブルと，配置変換が行われたデータを格納する．本システムでは，導入時に OS をインストールするものとは別にパーティションを作成し，そのパーティションを保存領域として使用する．

5. 実　　験

ここでは，**表 1** に示すような性能のマシンを用いて，Iozone を使用してディスク I/O 性能を測定した．

SSD のランダムライト性能の低さが顕著に現れるのは，SSD 上でデータの書き換えが起こるパターンである．Iozone ではベンチマークの開始時にファイル

Vol.2011-ARC-195 No.12
Vol.2011-OS-117 No.12
2011/4/13

表 1　実験に用いたマシンの性能

Table 1　machine specification.

CPU	Intel Core 2 Duo E6850 @ 2.00GHz
メモリ	4GB
OS	Ubuntu 10.04
ストレージ	Intel X25-M Mainstream 80G, MTRON MSD-SATA3035-032

を作成し，そのファイルに対してランダムリード・ライトを行うので，書き換えが起こっている場合のランダムライトの性能を測定することが可能である．

ここではIozoneによって，6GBのサイズのファイルに対するランダムライト性能の測定を行った．Iozoneによるベンチマークは，メモリサイズより巨大なファイルサイズを指定することで，バッファキャッシュによるディスクI/Oの発生回数の減退を抑制することで正確な性能を測定出来る．尚ランダムライト性能については，I/Oの対象となるデータのサイズ（レコードサイズ）を，本システムの高速化の対象となる4kBに指定した．また，本システムの有用性を確認する為，BitVisorを用いない場合（Normal Boot）の性能，BitVisorを用いた場合の性能，本システムを導入したBitVisorを用いた場合（conversion enabled）の性能の測定を行った．

Intel X25-M Mainstream 80Gにおける測定結果は**図 5**，MTRON MSD-SATA3035-032における測定結果は**図 6**に示す通りとなった．

実験に用いた二種類のSSDの両者とも，本システム導入時は未導入時と比較して性能向上が見られており，本システムによってSSDのランダムライトが高速化されているのが分かる．特に図6が示す通り，低いランダムライト性能を示したMTRON MSD-SATA3035-032については，本システムによる高速化の効果が顕著であり，BitVisorを用いずOSを直接起動した場合のランダムライト性能と比較して8倍程度の性能向上が見られた．Intel X25-M Mainstream 80Gにおいても，図5の通り，本システムを用いない場合のBitVisorの上でのベンチマーク結果と比較すると，本システムの導入により若干の高速化の効果が現れている．以上の結果より，本システムによってランダムライトを高速化する事が可能であり，かつ元々のランダムライトの性能が低ければその性能向上が顕著なものとなるといえる．

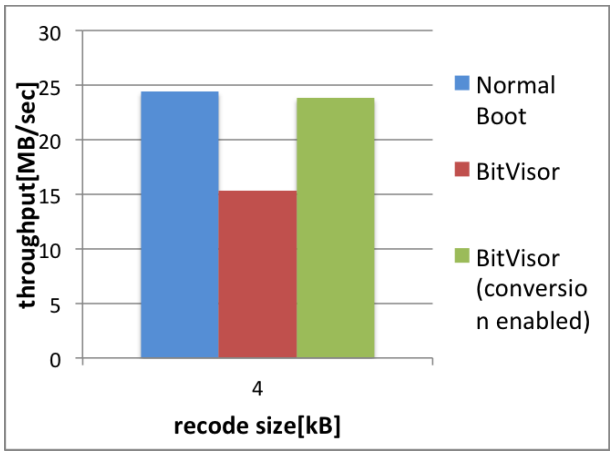

図 5 random write 性能比較 (Intel X25-M)

Fig. 5 random write performance.(Intel X25-M)

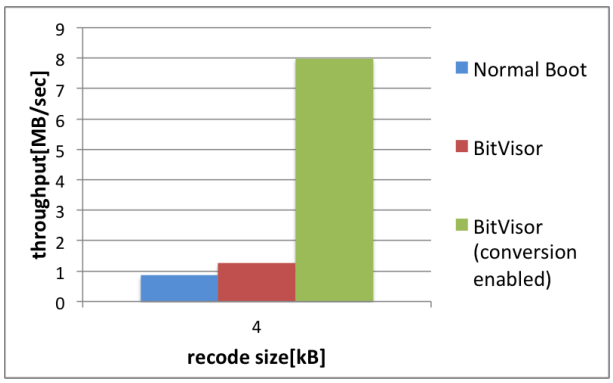

図 6 random write 性能比較 (MTRON MSD)

Fig. 6 random write performance.(MTRON MSD)

6. 関 連 研 究

SSD のランダムライト性能の低さを改善する既存研究としては，主にデバイスドライバレベルの研究とハードウェアレベルの研究が存在する．

ReSSD[4] はデバイスドライバのレイヤで動作し，SSD の記憶領域を通常のエリアと速度上昇の為に使用する予約エリアに分割する．そして，OS からランダムライトの要求が来た際に，その書き込みサイズが小さければアドレスを変換して予約エリアに書き込まれるようにする．これを繰り返すことによって，小サイズのランダム書き込みを予約エリアに集約してシーケンシャル書き込みに変換することで，書き込みのスループットを向上している．

CFLRU[5] は，Linux カーネルを修正しフラッシュメモリに適したバッファキャッシュの管理を行うことで書き込みのコストを抑える．フラッシュメモリにおいて書き換えを行うコストが高い為，書き換えが起こりうるページ（dirty page）へのアクセスをキャッシュし，新規書き込みが起こるページへのアクセスに対してはキャッシュを行わないようにすることで，実際にフラッシュメモリに対して起こりうる書き換え処理の回数を減らしている．

BPLRU[6] は，SSD 内部に小規模なプロセッサとライトバッファを導入し，ライトバッファをブロック単位で LRU（Least Recently Used）で管理することで頻繁に発生するランダムライトのパターンを学習し，SSD に実際に書き込まれる回数を削減する事で性能向上を狙っている．

7. おわりに

本研究では，VMM による書き込みデータのアドレス変換を行い，ランダムライトをシーケンシャルライトに変換することで，SSD のランダムライト性能を向上する手法の提案と実装，性能測定を行った．実装は BitVisor の ATA 準パススルードライバに対して機能追加することで実現した．実装したモジュールは，ランダムライト発生時にデータの書き込み先アドレスの変更を行いランダムライトの実行をシーケンシャルライトの実行に変換する．また，アドレス変換を追跡可能にするマッピングテーブルの記録を行う．性能測定として Iozone を用いたディスク I/O ベンチマークを行い，本システムによって SSD のランダムライト性能の向上が見込める事を確認した．

今後の課題として，配置変換したデータが不要になった際に空き領域の回収を行うガベージコレクション機構を実装する事が挙げられる．配置変換により保存領域の空き領域が圧迫されていくと，効率良く配置変換が行えなくなるため，本システムによる高速化の効率化を図る為にもガベージコレクション機構の実装は必要となる．また，マッピングテーブルの読み出しや保存に用いて PIO 発行モジュールは，現状ではシンプルな実装となっているので，DMA 転送の実現による高速化や動作の安定化など改良を行う必要がある．

References

1) Dumitru, D.: Understanding Flash SSD Performance, *DRAFT*, (online), available from

⟨http://managedflash.com/news/papers/easyco-flashperformance-art.pdf⟩
.

2) Shinagawa, T., Eiraku, H., Tanimoto, K., Omote, K., Hasegawa, S., Horie, T., Hirano, M., Kourai, K., Oyama, Y., Kawai, E., Kono, K., Chiba, S., Shinjo, Y. and Kato, K.: BitVisor: a thin hypervisor for enforcing i/o device security, *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, VEE '09, New York, NY, USA, ACM, pp.121–130 (online), DOI:http://doi.acm.org/10.1145/1508293.1508311 (2009).

3) : IOzone Filesystem Benchmark, (online), available from ⟨http://www.iozone.org/⟩.

4) Lee, Y., Kim, J.-S. and Maeng, S.: ReSSD: a software layer for resuscitating SSDs from poor small random write performance, *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, New York, NY, USA, ACM, pp. 242–243 (online), DOI:http://doi.acm.org/10.1145/1774088.1774138 (2010).

5) Park, S.-y., Jung, D., Kang, J.-u., Kim, J.-s. and Lee, J.: CFLRU: a replacement algorithm for flash memory, *Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems*, CASES '06, New York, NY, USA, ACM, pp. 234–241 (online), DOI:http://doi.acm.org/10.1145/1176760.1176789 (2006).

6) Kim, H. and Ahn, S.: BPLRU: a buffer management scheme for improving random writes in flash storage, *Proceedings of the 6th USENIX Conference on File and Storage Technologies*, FAST'08, Berkeley, CA, USA, USENIX Association, pp.16:1–16:14 (online), available from ⟨http://portal.acm.org/citation.cfm?id=1364813.1364829⟩ (2008).