不揮発性メインメモリエミュレータの評価

大森　侑[1,a)]　木村　啓二[1]

概要：低消費電力で大容量のメインメモリを構成可能とする不揮発性メモリにより構築されたメインメモリ（NVMM）が注目を集めている．NVMM は，電源遮断時もデータを保持できる, リード・ライト性能の非対称性が存在する，といった特性を持っており，コンパイラ最適化や OS もこれらの特性を考慮して設計する必要がある．しかしながら，現状では NVMM を持つシステムの入手は困難である．そのため現実的なアプリケーションや OS 等のソフトウェアまで含めたシステムの研究を行うにはシミュレータを利用せざるを得ず，その実験に膨大な時間を要する．本稿では，CPU と FPGA を持つ SoC チップ上の FPGA に実装した NVMM エミュレータを提案する．本エミュレータにより，DRAM によるメインメモリと NVMM を併せ持つヘテロジニアスメモリシステムの評価が可能となる．エミュレータとして実装することで実行時間の問題が解決され，シミュレータでは困難な評価が可能となる．本エミュレータ上で SPEC CPU 2017 ベンチマークを用いて評価を行った．評価の結果，アプリケーションの実行時間に NVMM が及ぼす影響の大きさは，キャッシュヒット率だけでなくメモリアクセスの頻度も関係することが分かった．また，連続領域に対してキャッシュフラッシュを実施する場合，そのオーバヘッドは NVMM アクセスレイテンシに影響を受けず，フラッシュする総ライン数に依存することも分かった．

1. はじめに

不揮発性メモリの一種として，フラッシュメモリはストレージデバイスに極めて広範囲に利用され，コンピュータシステムの軽量化及び低消費電力化に大きく貢献してきた．現在ではさらに，PCRAM や MRAM 等の不揮発性メモリ素子によりバイトアクセス可能なメモリモジュールを構成し，不揮発性メインメモリ（NVMM）としてメインメモリを再構成する研究が行われている．

NVMM は電源遮断時でもデータを保持可能でありリフレッシュが不要なため，低消費電力で大容量のメインメモリを構成可能となる．また，障害発生によるシステム停止時においてもデータを保存されているため，障害耐性の高いシステムを構築できる．その一方で，DRAM に比較してレイテンシが高く，さらにライト時のレイテンシや消費エネルギーがリード時に比べて高いという特性を持つ．そのため，NVMM を持つシステムを構築する場合，そのアーキテクチャはもちろんのこと，コンパイラ最適化や OS もこれら NVMM の特性を考慮して設計する必要がある．

しかしながら，現状では NVMM モジュールが入手困難であるため，NVMM を持つシステムの評価にはシミュレータが用いられることが多い．シミュレータは一般に実行時間が長く，評価に多くの時間を要する．そのため，実用上はシミュレーションの範囲をシステムの一部に限定したり，評価アプリケーションを縮小したりする必要があり，NVMM を持つシステム全体の評価や，実用的なアプリケーションを用いたコンパイラ最適化や OS の評価は困難である．

本稿では，CPU と FPGA を持つ SoC チップ上の FPGA にエミュレータとして実装した NVMM 評価環境を提案する．本エミュレータにより従来のシミュレータでは困難であった上記評価を可能とする．システムの構成は DRAM と NVMM を併せ持つヘテロジニアスメモリシステムとし，各メモリの使い分けによる評価も可能とする．また，FPGA 上にメモリコントローラを実装することで，メモリアクセスやメモリコントローラの動作を観察することも可能とする．

本稿の構成は以下の通りとなる．まず，第 2 節で NVMM 評価環境の関連研究を紹介する．次に，第 3 節でエミュレータの実装を述べ，第 4 節で NVMM の評価について報告する．最後に第 5 節でまとめる．

2. 関連研究

NVMM 評価環境に関する先行研究は，大きく分けてシミュレータとエミュレータに分類できる．

[1] 早稲田大学基幹理工学部
School of Fundamental Science and Engeneering, Waseda Univ.
[a)] oy@kasahara.cs.waseda.ac.jp

© 1959 Information Processing Society of Japan

表 1　評価環境詳細

FPGA	Xilinx Zynq-7000 SoC ZC706
デバイス	Zynq-7000 XC7Z045-2FFG900C SoC
CPU コア	Cortex-A9 Dual Core, 667 MHz
L1 キャッシュ	I=32 KB/core, D=32 KB/core
L2 キャッシュ	256 KB/processor
PS DRAM	1 GB, DDR3-1066, 16b×2 components
PL DRAM	1 GB, DDR3-1600, 8b×8, SO-DIMM
PL 動作周波数	200 MHz
カーネル	GNU/Linux 4.14.0-xilinx-00081-g88cc987
ファイルシステム	Ubuntu 16.04 LTS

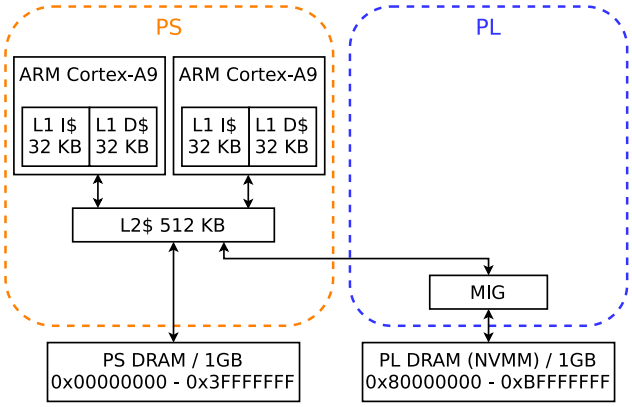

図 1　評価環境ブロック図

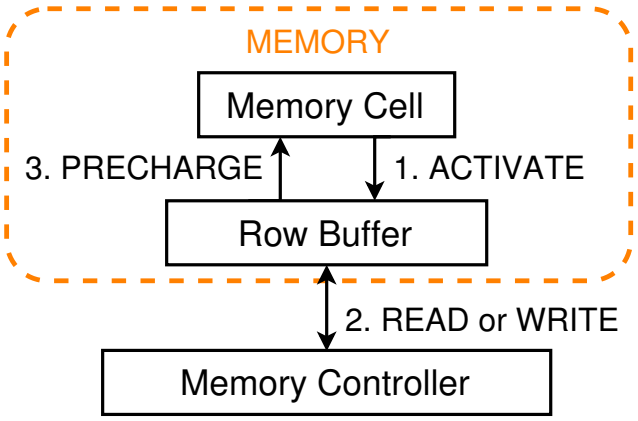

図 2　メモリインタフェース

NVMM シミュレータの先行研究として，gem5[1]，NVMain[2][3]，PCMSim[4]，HMMSim[5]，Quartz[6][7] がある．gem5，NVMain，PCMSim，HMMSim はサイクルアキュレートシミュレータであり，Quartz は Intel プロセッサによる実サーバ機のメインメモリを NVMM として模擬するシミュレータである．サイクルアキュレートシミュレータはハードウェアに依存せず，システムの構成や設定パラメータに柔軟であるが，実行時間が長い．Quartz は，一定時間毎に LLC ミスに応じた時間アプリケーションをスリープさせる．既存システム上で動作させるため実行時間の問題は解決できるが，メモリアクセスやメモリコントローラの細かい動作は評価できない．

エミュレータの先行研究として，TUNA[8][9] がある．TUNA では FPGA 上の実システムを動作させて評価を行うため，評価に要する時間が長大となるという問題を解決している．一方で，TUNA では DRAM をカーネルが乗る最小限に制限して，アプリケーションのデータ構造を全て NVMM に配置するホモジニアスメモリシステムのように評価を行っている．本稿では，TUNA では行っていない，DRAM と NVMM を併せ持つヘテロジニアスメモリシステムとして評価を行う．

3. 評価環境実装

本稿で実装した評価環境の実装について述べる．

実装には，FPGA 搭載 SoC である Xilinx Zynq-7000 SoC ZC706（以下 ZC706）を使用した．コンフィギュレーションを**表 1**，ブロック図を**図 1** に示す．ARM CPU が搭載された PS 側の DRAM を DRAM，FPGA が搭載された PL 側の DRAM を NVMM としてエミュレートするヘテロジニアスメモリシステムを実装するため，以下の実装・改変を行った．

(1) 可変メモリアクセスレイテンシの実装
(2) NVMM キャッシャビリティ改変
(3) キャッシュフラッシュ用のカーネルモジュール実装
(4) アプリケーション移植用のライブラリ実装

3.1 可変メモリアクセスレイテンシの実装

NVMM モジュールが既存のメモリインタフェースで使われるならば，**図 2** のように，メモリセルへのアクセスは ACTIVATE，PRECHARGE のみで行われる．NVMM のアクセスレイテンシはメモリセルのレイテンシに起因すると考えられるため，メモリコントローラ内で ACTIVATE と PRECHARGE のレイテンシである，tRCD と tRP を変化させることで，既存のメモリコントローラと DRAM を用いて NVMM をエミュレートできる．

本稿では，NVMM のメモリコントローラとして，Xilinx が提供する MIG 4.1 を用いた．MIG の実装では tRCD と tRP は固定値であるが，これを外部入力の任意値に設定できるように RTL 記述を改変した．加えて，レイテンシを記憶するレジスタをアドレス空間にマップし，PS から動的に変更できるようにした．レイテンシの設定精度は 5 [ns] である．本改変によりデータ化けが発生しないことを確認した．また，DRAM を使っているためリフレッシュが定期実行されるが，リフレッシュはこの改変の影響を受けない．

3.2 NVMM キャッシャビリティ改変

本稿で実装した評価環境では，カーネルとして Xilinx 提供の Linux Kernel[10] を用いた．このカーネルは，PL へのアクセスを全てノンキャッシャブルに行い，CPU キャッ

© 1959 Information Processing Society of Japan

```
void  NVMM_Initialize(void)
void  NVMM_Finalize(void)
void *NVMM_Malloc(size_t size)
void *NVMM_Calloc(size_t nmemb, size_t size)
void *NVMM_Realloc(void *ptr, size_t size)
void  NVMM_Free(void *ptr)
```

図 3 NVMM 管理ライブラリインタフェース

シュをバイパスする．提案するエミュレータでは NVMM は PL にあるため，NVMM をキャッシャブルにするためのカーネル改変を実施した．

NVMM の確保は mmap() システムコールによって行い，領域のキャッシャビリティはこの中で決定される．本稿では，領域の物理アドレスが NVMM に割り当てられたアドレスならばキャッシャブルと扱うようにした．ただし，O_SYNC が指定されているならばノンキャッシャブルとして扱う．

3.3 キャッシュフラッシュ用のカーネルモジュール実装

CPU がキャッシュを持つ時，ストア命令はデータの主記憶到達を保証しない．キャッシュは揮発性メモリであるため，NVMM で不揮発性を保証する時，データを明示的にキャッシュから NVMM に書き戻す必要がある．

ZC706 の CPU である Cortex-A9（ARMv7-A）は，仮想アドレスに対応するキャッシュラインをフラッシュする DCCMVAC 命令を持つ．しかし，DCCMVAC 命令は特権命令であるため，ユーザ空間では実行できない．そこで，カーネルモジュールを実装し，ioctl() でカーネル空間の DCCMVAC を呼び出すようにした．また，仮想アドレスと領域サイズを引数とするように実装し，ユーザ/カーネル空間切り替えのオーバヘッドを隠蔽できるようにした．加えて，指定範囲のフラッシュ前後にメモリバリアを実行しメモリオーダリングを保証している．

3.4 アプリケーション移植用のライブラリ実装

提案エミュレータでは，NVMM 上メモリ領域の確保は/dev/mem から直接 mmap() で行う必要がある．しかし，mmap() は確保領域サイズに制限があり，malloc() と互換性がない．munmap()/free() についても同様で，また，calloc()/realloc() に相当する関数は無い．

そこで，本稿では NVMM 管理用ライブラリを実装した．malloc()/calloc()/realloc()/free() と同じ引数を持ち，バイト単位で確保領域のサイズを指定できる．ただし，アプリケーションの先頭で NVMM_Initialize() を呼び，最後に NVMM_Finalize() を呼ぶ必要がある．提案エミュレータ用に実装したインタフェースを**図 3** に示す．

表 2 STRIDE と対応するメモリアクセス

STRIDE	メモリアクセスの様子
4	連続アクセス
32	アクセス毎にキャッシュミスが発生
8192	アクセス毎に ACT/PRE が発生

4. NVMM 性能評価

3 節で実装した評価環境を用いて，二つの評価を行った．4.1 節で NVMM がメモリアクセス性能に及ぼす影響の評価，4.2 節で NVMM がアプリケーションの実行時間に及ぼす影響の評価について，それぞれ述べる．

4.1 マイクロベンチマークによる NVMM 性能評価

NVMM アクセスレイテンシがメモリアクセス性能に及ぼす影響について，マイクロベンチマークを用いて評価した．マイクロベンチマークのカーネルを**図 4** に示す．LLC の 2 倍である 1 MB 範囲に対してアクセスを行い，実行時間を clock() で得る．実行時間をループ回転数で割ることでアクセスあたりの実行時間が得られ，メモリバンド幅が計算できる．カーネル中の STRIDE は，キャッシュラインサイズが 32 Byte，NVMM Row Buffer が 8192 Byte であるため，**表 2** の意味を持つ．

同 STRIDE の DRAM 測定結果で正規化した測定結果を，**図 5**〜**図 16** に示す．リードに関しては，tRP はメモリセルのライトレイテンシに相当するので測定を省略し，Additional tRP = 0 [ns] に固定して測定した．ライトに関しては，メモリリードとメモリライト両方のアクセスが発生するため，tRCD/tRP 両方を変化させて測定した．

本評価環境上では，NVMM リードバンド幅は，Additional tRCD/tRP が 0 [ns] の時 0.5 倍程度に劣化し，1,000 [ns] の時 0.10 倍程度まで劣化する．また，NVMM ライトバンド幅は，Additional tRCD/tRP が共に 0 [ns] の時 0.6 倍程度に劣化し，共に 1,000 [ns] の時 0.03 倍まで劣化する．ライトアクセスはリードとライトを行うため，tRCD/tRP 増加による性能劣化が大きい．また，ACTIVATE と PRECHARGE の回数は等しいため，tRCD/tRP に対して同程度の影響を受ける．

4.2 SPEC CPU 2017 ベンチマークによる NVMM 性能評価

NVMM アクセスレイテンシがアプリケーションの実行時間に及ぼす影響について，SPEC CPU 2017[11] を用いてアプリケーション特性に着目して評価した．着目した特性は LLC ヒット率である．着目した理由は，LLC ヒット率の違いによりメインメモリへのアクセス頻度に差異が生じ，NVMM アクセスレイテンシの影響を観測しやすいと考えられるためである．

© 1959 Information Processing Society of Japan

```
#define STRIDE (4 or 32 or 8192)
#define MiB (1024 * 1024)
base := return value of mmap()

start = clock();
for (addr = base; addr < base + MiB;
     addr += STRIDE) {
#if defined(READ)
    val = *((volatile unsigned long *)addr);
#elif defined(WRITE)
    *((volatile unsigned long *)addr) = 0L;
#endif
}
end = clock();

sec = (double)(end - start) / CLOCKS_PER_SEC;
ave = sec / (1 * MiB / STRIDE);
```

図 4　マイクロベンチマークカーネル

表 3　キャッシュヒット率測定環境

プロセッサ	Intel(R) Xeon(R) E5-2667 v4 @3.20GHz
コア数	8
L1 キャッシュ	I=32 KB/core, D=32 KB/core
L2 キャッシュ	256 KB/core
L3 キャッシュ	25 MB/processor

表 4　評価対象ベンチマーク

ベンチマーク	LLC ヒット [%]	概要
508.namd_r	94.1	分子動力学法
541.leela_r	93.1	モンテカルロ木探索
557.xz_r	19.1	データ圧縮
519.lbm_r	1.6	流体力学

LLC ヒット率の測定は，評価環境上では困難であったため，**表 3** に示す Intel Xeon プロセッサを持つ環境で Intel PCM を用いて行った．表 1 と表 3 より，評価環境の LLC 近い容量を持つ L2 キャッシュヒット率を評価環境の LLC ヒット率と考え，昇順と降順から C/C++で記述されたベンチマーク二つずつを評価対象とした．**表 4** に示す．これらのカーネルデータ構造を NVMM に配置し，アクセスレイテンシを変化させながら実行時間の変化を評価した．アプリケーションの実行は 1 コア・逐次実行であり，実行時間は GNU Time で測定した．

4.2.1　508.namd_r

508.namd_r は，分子動力学法アプリケーションである．カーネルアルゴリズムを**図 17** に示す．各イテレーションにおいて calc***() を 6 回，moveatomes() を 1 回それぞれ実行する．calc***() は緑の MayMod で指されるデータを変更し，moveatomes() は青の MayMod で指されるデータを変更する．よって，MayMod で指される，reductionData，f_nbond，f_slow，atoms を NVMM に配置した．

測定結果をオリジナルの実行時間で正規化した結果を**図 18** に示す．

4.2.2　541.leela_r

541.leela_r は，モンテカルロ木探索により囲碁の最善手探索を行う人工知能アプリケーションである．カーネルアルゴリズムを以下に示す．

(1) 現在の盤の状態を根としたノード数 1 の木を作成
(2) 展開済みノードから，最も評価値の高いノードを訪問
(3) 訪問ノードの訪問回数が閾値未満なら (2) に戻る
(4) 訪問ノードの葉を展開（プレイアウト）
　この時，合法手からランダムに展開
(5) (2) に戻る
　ただし，ノード展開数やデッドライン制限を受ける

カーネルデータ構造は動的に展開される木であるから，木のノードを NVMM に配置した．

測定結果をオリジナルの実行時間で正規化した結果を**図 19** に示す．

4.2.3　557.xz_r

557.xz_r は，.xz ファイル圧縮アプリケーションである．カーネルアルゴリズムを**図 20** に示す．FileIO への依存を減らすよう実装されており，最初に.xz ファイルをメモリ上に読み込み，以降の処理は全てインメモリで行う．メモリ上に擬似的な stdin/stdout を確保し，別に確保されたバッファにデータをコピーしながらデータを圧縮する．pseudo stdin/stdout を NVMM に配置した．この配置により，DRAM は NVMM のキャッシュとして機能する．

測定結果をオリジナルの実行時間で正規化した結果を**図 21** に示す．

4.2.4　519.lbm_r

519.lbm_r は，流体力学アプリケーションである．カーネルアルゴリズムを以下に示す．

(1) srcGrid，dstGrid を確保
(2) srcGrid をソースに計算し，dstGrid に格納
(3) srcGrid，dstGrid を交換
(4) (2) に戻る

srcGrid，dstGrid を全て NVMM に配置した．

測定結果をオリジナルの実行時間で正規化した結果を**図 22** に示す．

4.2.5　LLC ヒット率と NVMM アクセスレイテンシの考察

評価アプリケーションの LLC ヒット率と提案エミュレータ上の正規化実行時間の関係を**表 5** に示す．表 5 中の正規化実行時間は，Additional tRCD/tRP をそれぞれ 1,000 [ns] に設定した際の値である．NVMM アクセスレイテンシの影響の大きさは LLC ヒット率に依存すると考えられるが，LLC ヒット率 94.1%の 508.namd_r が 93.1%の 541.leela_r より NVMM アクセスレイテンシの影響が大きいため，LLC ヒット率だけでは判断できない

© 1959 Information Processing Society of Japan

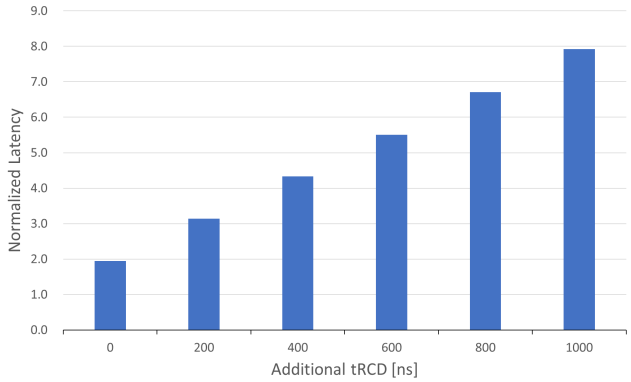

図 5　NVMM 正規化リードレイテンシ（STRIDE=4）

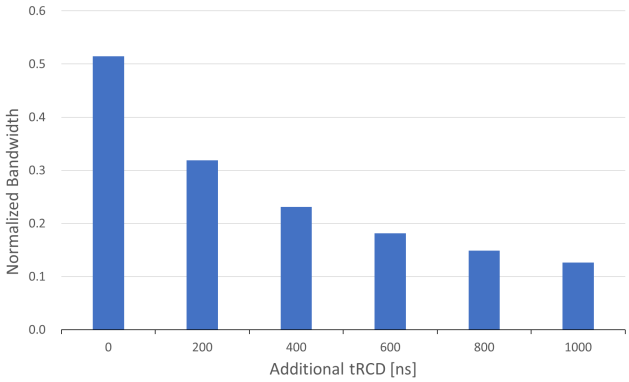

図 6　NVMM 正規化リードバンド幅（STRIDE=4）

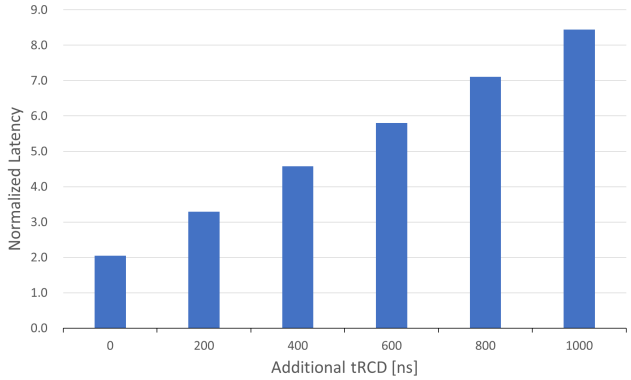

図 7　NVMM 正規化リードレイテンシ（STRIDE=32）

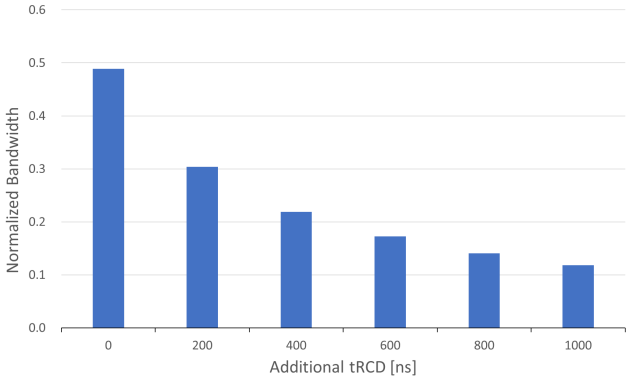

図 8　NVMM 正規化リードバンド幅（STRIDE=32）

表 5　各ベンチマークの LLC ヒット率及び正規化実行時間

ベンチマーク	LLC ヒット率 [%]	正規化実行時間
508.namd_r	94.1	1.32
541.leela_r	93.1	1.13
557.xz_r	19.1	3.35
519.lbm_r	1.6	32.52

表 6　各ベンチマークのメモリアクセス頻度

ベンチマーク	リード [/s]	ライト [/s]	トータル [/s]
508.namd_r	120,130	61,782	181,912
541.leela_r	31,047	15,813	46,860
557.xz_r	1,151,854	638,996	1,790,850
519.lbm_r	7,524,022	4,286,303	11,810,325

上記の現象の調査のため，さらにメモリアクセスの頻度を測定した．メモリアクセスは CPU と MIG の間に論理回路を実装して行った．測定結果を**表 6** に示す．この測定結果を用いて，508.namd_r と 541.leela_r を比較する．namd は leela に比べて LLC ヒット率は 1%高いが，正規化実行時間は 1.17 倍である．これについてメモリアクセス頻度を見ると，namd は leela の 3.88 倍である．すなわち，LLC ヒット率が同程度の場合，NVMM アクセスレイテンシの影響はメモリアクセス頻度に依存することがわかる．

4.2.6　キャッシュフラッシュオーバヘッド

3.3 節で述べたように，NVMM を使う時はキャッシュフラッシュを考える必要がある．評価アプリケーションに対して，キャッシュフラッシュを挿入してオーバヘッドを測定した．508.namd_r は，calc***()/moveatomes() の直後に対応する MayMod をフラッシュした．541.leela_r は，ノードの作成直後に新規ノードをフラッシュした．557.xz_r は，stdout への書き戻し直後に対応する stdout の領域をフラッシュした．519.lbm_r は，srcGrid/dstGrid の交換直前に dstGrid を全てフラッシュした．測定結果を**表 7**〜**表 10** に示す．オーバヘッドが実行時間に占める割合を**表 11** に示す．

まず，表 7〜表 10 より，バラつきは見られるが，Additional tRCD/tRP による一定の増加は見られない．これは，フラッシュはラインの置換時に発生するライトバックを明示的に行っているだけで，ライトバックの総計は増加しないためである．ただし，フラッシュされたラインが置換されず再びダーティーになる場合，影響は大きくなる．さらに，本評価で用いたキャッシュフラッシュ API では，指定されたメモリ領域に対するキャッシュフラッシュ処理の前後でメモリバリア命令を実行するため，一度のキャッシュフラッシュ API 呼び出し中はメモリアクセス並列性が阻害されないことも理由として挙げられる．

表 11 より，評価ベンチマークでは，キャッシュフラッシュオーバヘッドは，最大でも実行時間の 8%未満であることがわかる．519.lbm_r は後述する原因によりオーバヘッドに無駄が多いため，これを除くと 3%未満となり，オー

© 1959 Information Processing Society of Japan

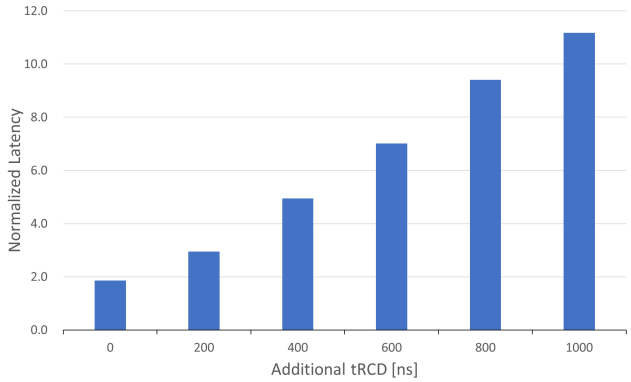

図 9　NVMM 正規化リードレイテンシ（STRIDE=8192）

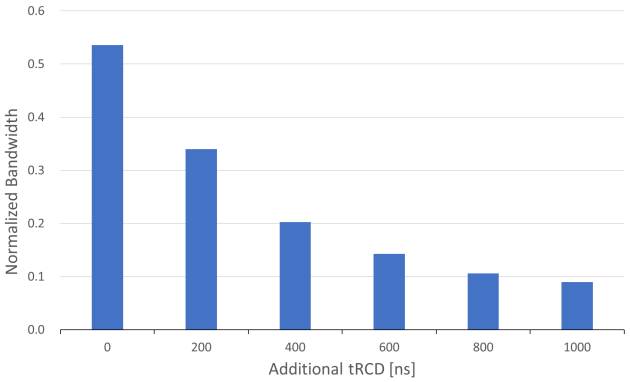

図 10　NVMM 正規化リードバンド幅（STRIDE=8192）

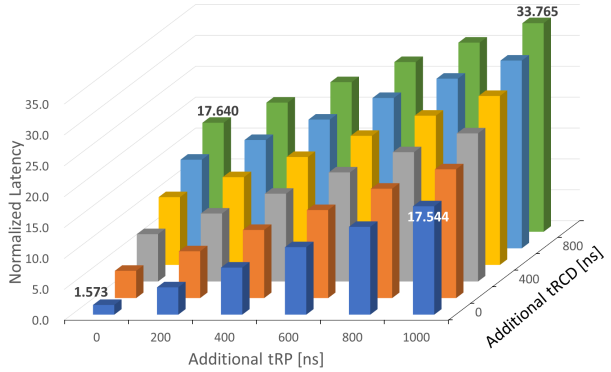

図 11　NVMM 正規化ライトレイテンシ（STRIDE=4）

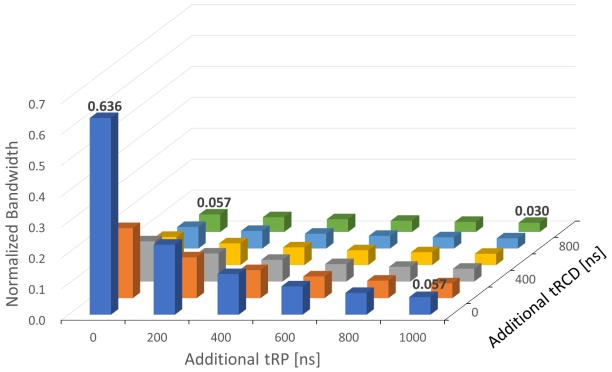

図 12　NVMM 正規化ライトバンド幅（STRIDE=4）

表 7　508.namd_r キャッシュフラッシュオーバヘッド [s]

NAMD		Additional tRP [ns]					
		0	200	400	600	800	1000
Additional tRCD [ns]	0	2.52	1.64	1.74	2.55	1.11	0.97
	200	1.38	1.15	1.53	1.76	1.84	1.91
	400	1.18	1.44	0.96	1.84	2.76	1.23
	600	1.16	0.51	1.83	2.59	1.05	1.16
	800	0.32	0.96	0.98	0.65	1.09	1.29
	1000	0.98	1.24	1.13	0.91	1.92	0.94

表 8　541.leela_r キャッシュフラッシュオーバヘッド [s]

LEELA		Additional tRP [ns]					
		0	200	400	600	800	1000
Additional tRCD [ns]	0	0.26	0.32	0.26	0.41	0.37	0.16
	200	0.35	0.31	0.28	0.18	0.21	0.20
	400	0.30	0.25	0.25	0.32	0.27	0.22
	600	0.24	0.34	0.42	0.29	0.28	0.26
	800	0.18	0.08	0.34	0.32	0.06	0.25
	1000	0.26	0.34	0.23	0.28	0.44	0.21

バヘッドは十分に小さい．

キャッシュフラッシュオーバヘッドが依存する要因を調査するため，フラッシュを挿入した際の，フラッシュしたライン数と頻度を測定した．すなわち，実際にライトバックされたラインではなく，フラッシュを発行したライン数である．結果を**表 12** に示す．オーバヘッドは，表 11 の最大列より，519.lbm_r，508.namd_r，541.leela_r，557.xz_r の順で大きく，これは頻度の大小ではなくライン数の大小と一致する．よって，キャッシュフラッシュオーバヘッドはフラッシュする総ライン数に依存する．特に，519.lbm_r は 204 MB の dstGrid 全領域にフラッシュをかけているが，LLC が 512 KB であるため無駄が多くオーバヘッドが大きくなっている．

表 9　557.xz_r キャッシュフラッシュオーバヘッド [s]

XZ		Additional tRP [ns]					
		0	200	400	600	800	1000
Additional tRCD [ns]	0	0.03	0.01	0.06	0.02	0.05	0.05
	200	0.01	0.03	0.01	0.28	0.08	0.06
	400	0.11	0.06	0.09	0.01	0.18	0.15
	600	0.05	0.05	0.02	0.08	0.32	0.23
	800	0.02	0.06	0.15	0.03	0.01	0.01
	1000	0.04	0.07	0.12	0.01	0.05	0.02

5. まとめ

本稿では，NVMM エミュレータを CPU と FPGA を持

© 1959 Information Processing Society of Japan

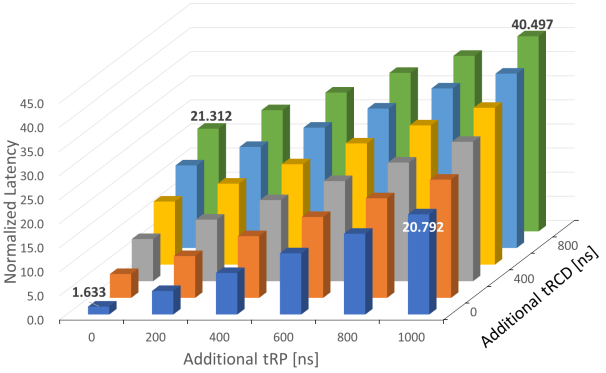

図 13　NVMM 正規化ライトレイテンシ（STRIDE=32）

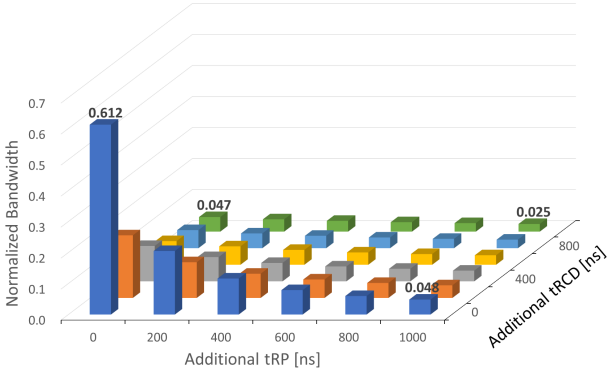

図 14　NVMM 正規化ライトバンド幅（STRIDE=32）

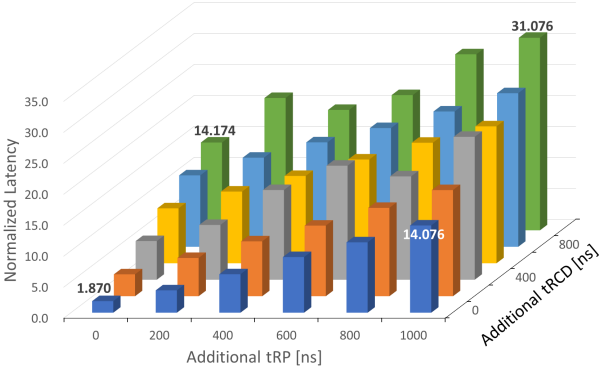

図 15　NVMM 正規化ライトレイテンシ（STRIDE=8192）

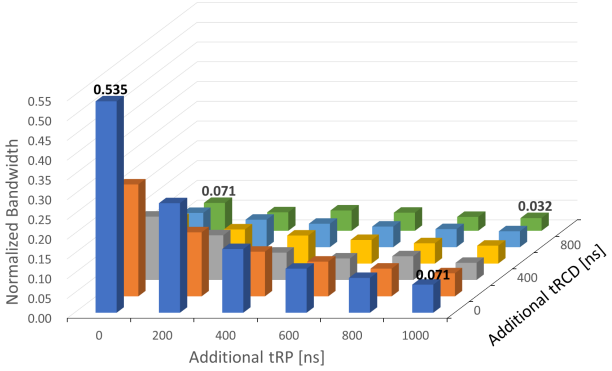

図 16　NVMM 正規化ライトバンド幅（STRIDE=8192）

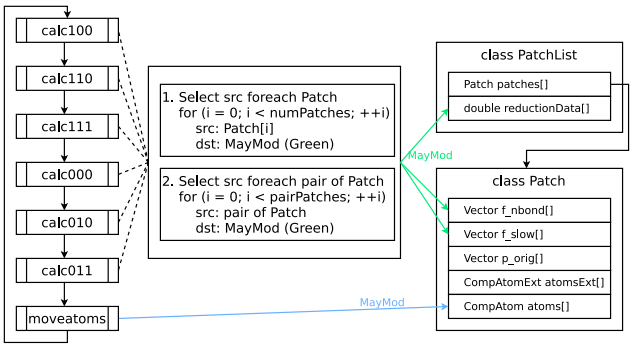

図 17　508.namd_r カーネルアルゴリズム

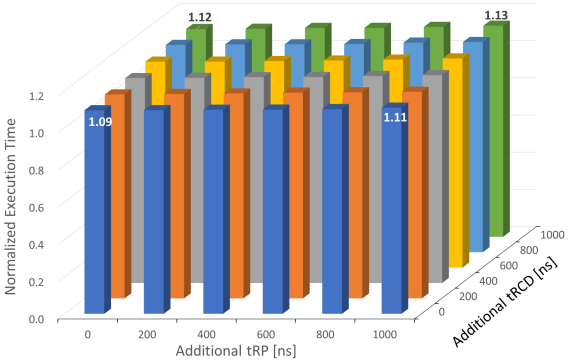

図 19　541.leela_r 正規化実行時間

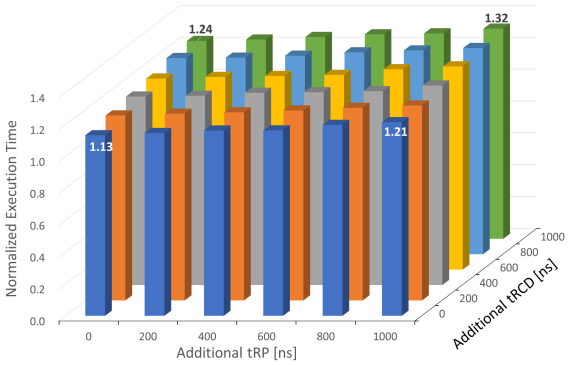

図 18　508.namd_r 正規化実行時間

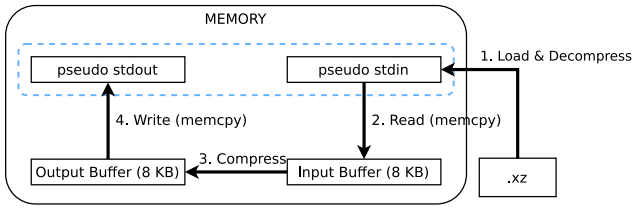

図 20　557.xz_r カーネルアルゴリズム

つ SoC 上に実装し，NVMM アクセスレイテンシがアプリケーションの実行時間に及ぼす影響を SPEC CPU 2017 ベンチマークを用いて評価した．

評価により，アプリケーションの実行時間に NVMM が及ぼす影響は，LLC ヒット率だけでなくメモリアクセス頻度も関係することが分かった．また，連続領域に対してキャッシュフラッシュを行いキャッシュ上のデータを

© 1959 Information Processing Society of Japan

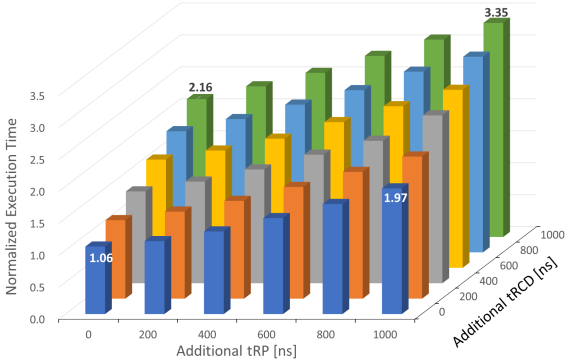

図 21　557.xz_r 正規化実行時間

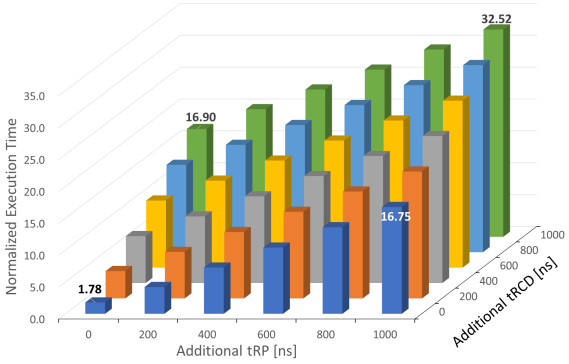

図 22　519.lbm_r 正規化実行時間

表 10　519.lbm_r キャッシュフラッシュオーバヘッド [s]

		Additional tRP [ns]					
LBM		0	200	400	600	800	1000
Additional tRCD [ns]	0	5.70	5.47	5.48	5.43	5.49	5.49
	200	5.50	5.52	5.52	5.44	5.38	5.49
	400	5.49	5.44	5.57	5.42	5.48	5.44
	600	5.44	5.53	5.57	5.43	5.42	5.48
	800	5.49	5.52	5.65	5.55	5.58	5.50
	1000	5.60	5.41	5.43	5.46	5.49	4.94

表 11　キャッシュフラッシュオーバヘッド [%]

ベンチマーク	最小 [%]	最大 [%]
508.namd_r	0.32	2.72
541.leela_r	0.10	0.75
557.xz_r	0.04	1.50
519.lbm_r	0.41	7.91

表 12　キャッシュフラッシュ測定結果

ベンチマーク	ライン数 [line]	頻度 [line/s]
508.namd_r	922,288	9,775
541.leela_r	248,525	4,353
557.xz_r	166,898	9,382
519.lbm_r	134,000,000	1,859,045

NVMM に書き戻す際は，キャッシュフラッシュオーバヘッドは NVMM アクセスレイテンシに影響を受けず，フラッシュする総ライン数に依存することも分かった．

今後，本エミュレータによりソフトウェア・ハードウェア両方からの NVMM 利用最適化技術の開発が可能となる．

謝辞　本研究の一部は東芝メモリ株式会社と早稲田大学との組織連携活動の一環として実施した．

参考文献

[1] Binkert, N., Beckmann, B., Black, G., Reinhardt, S. K., Saidi, A., Basu, A., Hestness, J., Hower, D. R., Krishna, T., Sardashti, S., Sen, R., Sewell, K., Shoaib, M., Vaish, N., Hill, M. D. and Wood, D. A.: The Gem5 Simulator, *SIGARCH Comput. Archit. News*, Vol. 39, No. 2, pp. 1–7 (2011).

[2] Poremba, M. and Xie, Y.: NVMain: An Architectural-Level Main Memory Simulator for Emerging Non-volatile Memories, *2012 IEEE Computer Society Annual Symposium on VLSI*, pp. 392–397 (2012).

[3] Poremba, M., Zhang, T. and Xie, Y.: NVMain 2.0: A User-Friendly Memory Simulator to Model (Non-)Volatile Memory Systems, *IEEE Computer Architecture Letters*, Vol. 14, No. 2, pp. 140–143 (2015).

[4] Wang, J. and Wang, B.: PCMSim: A Hybrid Memory System Simulator for the Cloud Storage, *2017 Fifth International Conference on Advanced Cloud and Big Data (CBD)*, pp. 81–86 (2017).

[5] Bock, S., Childers, B. R., Melhem, R. and Mosse, D.: HMMSim: a simulator for hardware-software co-design of hybrid main memory, *2015 IEEE Non-Volatile Memory System and Applications Symposium (NVMSA)*, pp. 1–6 (2015).

[6] Volos, H., Magalhaes, G., Cherkasova, L. and Li, J.: Quartz: A Lightweight Performance Emulator for Persistent Memory Software, *Proceedings of the 16th Annual Middleware Conference*, Middleware '15, ACM, pp. 37–49 (2015).

[7] Koshiba, A., Hirofuchi, T., Akiyama, S., Takano, R. and Namiki, M.: Towards write-back aware software emulator for non-volatile memory, *2017 IEEE 6th Non-Volatile Memory Systems and Applications Symposium (NVMSA)*, pp. 1–6 (2017).

[8] Lee, T., Kim, D., Park, H., Yoo, S. and Lee, S.: FPGA-based prototyping systems for emerging memory technologies, *2014 25nd IEEE International Symposium on Rapid System Prototyping*, pp. 115–120 (2014).

[9] Lee, T. and Yoo, S.: An FPGA-based platform for non volatile memory emulation, *2017 IEEE 6th Non-Volatile Memory Systems and Applications Symposium (NVMSA)*, pp. 1–4 (2017).

[10] Xilinx: The official Linux kernel from Xilinx, Xilinx (online), available from ⟨https://github.com/Xilinx/linux-xlnx⟩ (accessed 2019-01-27).

[11] spec.org: SPEC CPU(R) 2017, Standard Performance Evaluation Corporation (online), available from ⟨https://www.spec.org/cpu2017/⟩ (accessed 2019-01-27).

© 1959 Information Processing Society of Japan