Vol.2020-OS-148 No.1
2020/2/27

ハイブリッド メモリシステムをリモート PCからアクセスした場合の評価と考察

大江 和一[1,a)]

Abstract: Intel Optane DC Persistent Memory (DCPM) の市販により，PC のメインメモリとして DRAM と DCPM のハイブリッド構成を選択することが可能になった．DCPM は，DRAM と比較してレイテンシは数倍増加するが大容量且つ低価格であり，不揮発な特徴を有している．このハイブリッドメモリシステムを実装した PC は，大容量メモリを必要とするインメモリデータベースや Virtual Machine(VM) などへの適用が期待されている．さらに DCPM の不揮発な特徴を用いた高速ストアとしての運用も考えられる．本論分では，ハイブリッドメモリシステムの一部の DCPM 領域を高速ストア領域として割り当てるケースを想定し，リモート PC から高速ストア領域へのアクセスとローカル PC からの DRAM/DCPM へのアクセスが競合した場合の評価を行った．その結果，ローカル PC から DRAM へアクセスするケースにおいても，リモート PC からのアクセス競合により最大 1/5 まで性能が低下することが分かった．この評価結果より，我々はリモート PC から DCPM 領域へアクセスする時は，ローカル PC からの DRAM/DCPM アクセスとの間で何らかのスケジューリングが必要と考えている．

1. はじめに

近年、様々な種類の Persistent Memory (PM) [1] の研究開発が進められており，その一部は SSD や DIMM に組み込まれ製品としての販売が始まっている．Intel は，昨年の 4 月より Inte DC Persistent Memory (DCPM) [2] の販売を開始した．DCPM は，Dual Inline Memory Modle (DIMM) Slot を介してコンピュータシステムに接続し，メモリとしてもストレージとしても利用することが可能である [3], [4]．DCPM はバイトアクセス可能な特徴を有しており，その性能は DRAM の 1/2 から 1/5 程度であることが知られている [5], [6]．DCPM の容量は，CPU ソケットあたり最大 3TB である．例えば，一般的な 2 ソケット CPU を搭載したサーバだと，DCPM だけで最大 6TB の巨大なメモリ空間を確保することが可能である．

DCPM は，DRAM と比較して大容量であるが低速となる特徴を有しており，DRAM との併用が前提となる．DCPM つき PC は，メモリ容量を大量に必要とする in-memory database system や Virtual Machine (VM) への適用が考えられる．特に in-memory database system においては，DRAM と DCPM のハイブリットメモリシステムを前提にした多くの研究 [7], [8], [9] が行われている．VM に関しては，DRAM と DCPM のハイブリットメモリシステムを用いることで，より多くの VM を 1 つの PC で運用できるようになる．しかしながら，我々は，DCPM の巨大な容量をこれらのアプリケーションのみで使いきれないケースも多く存在すると考えており，余った DCPM 領域は DCPM が付いていない PC のメモリ/ストレージ領域として使うことができる．DCPM が付いていない PC から DCPM へは，Infiniband などがサポートする Remote Direct Memory Acccess (RDMA) を用いることで，高スループット低レイテンシなアクセスを行うことができる．ところで，今村らの報告 [10] によると，DCPM つき PC 上で複数のアプリケーションを同時に動作させた場合，ひとつのアプリケーションが DCPM へのアクセスを行うと残りのアプリケーションが DCPM だけではなく DRAM へアクセスする場合においても大幅な性能低下が起きることが示されている．そこで我々は，DCPM つき PC 上で動作するアプリケーションと DCPM なし PC 上のアプリケーションが同時に DRAM/DCPM をアクセスするケースにおいても同様な性能低下が起きる可能性があると考えた．

本論文は，DCPM つき PC 上で動作するアプリケーションと DCPM なし PC 上で動作するアプリケーションが，DCPM つき PC の DRAM/DCPM を同時アクセスしたときの性能干渉について評価したものである．DCPM つき PC 上で動作するアプリケーションは，intel Memory Latency Checker (intel MLC)[11] を使用した．DCPM なし PC 上で動作するアプリケーションは，Infiniband Verbs Performance Tests [12] の中の ib_write_bw を拡張したテストツールを使用した．このテストツールを ib_write_bw+と呼ぶことにする．その結果，ib_write_bw+から RDMA 経由で DCPM つき PC の DCPM にアクセスを行うと，intel MLC から DRAM へアクセスを行う場合においても性能干渉により 1/5 のスループットになることが分かった．この評価結果より，我々は DCPM つき PC の DCPM 領域を他 PC と共有を行う場合は，DCPM つき PC 上で動作するアプリケーションのメモリアクセスと他 PC 上で動作するアプリケーションの RDMA との間で何らかの排他制御が必要になるという結論に至った．

[1] (株) 富士通研究所
FUJITSU LABORATORIES LTD.
[a)] ooe.kazuichi@fujitsu.com

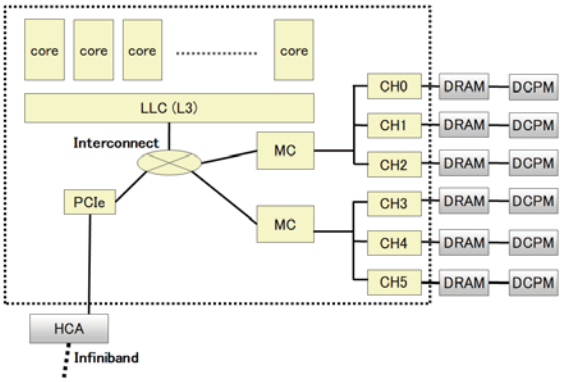

Fig. 1 intel CPU configuration with DCPM

2. 背景

2.1 Persistent Memory (PM) とは

Persistent Memory (PM) は，これまで多くの研究が行われてきた．その特徴は，DRAM と比較して低コストでより大きな容量を実現できるが，性能は DRAM 比 1/2 から 1/5 である [1]．そして，2019 年 4 月より Intel Optane DC Persistent Memory (DCPM) が発売となり，一般ユーザが PM を利用できるようになった．DCPM の特徴は，1) 不揮発且つバイトアクセス可能であること，2) 容量は DRAM 比 4 倍，性能は DRAM 比 1/4 から 1/6，GB コストは DRAM 比 1/2，である [10]．Intel は，DCPM のアップデートを計画しており，近い将来より大容量且つ高速な DCPM 登場するかも知れない．

2.2 DCPM を用いたハイブリッド メモリシステム

図 1 は，DCPM をサポートした intel CPU の内部構成である．CPU cores，Memory Controller (MC)，そして，PCI Express (PCIe) の間は Interconnect で相互結合されている．MC 下に複数の CHANNEL(CH) が存在し，各 CH に DRAM と DCPM が接続される．DCPM は DRAM とペアでの接続が求められる．Infiniband RDMA 経由で DCPM にアクセスする場合は，CPU の Last Level Cache (LLC) などは通らず，PCIe から Interconnect を経由して直接 MC にアクセスを行う．

2.3 アプリケーションから DCPM をアクセスする方法

DRAM と DCPM が混在したハイブリッド メモリシステムは，memory mode もしくは app direct mode のどちらかに設定して利用する [3]．memory mode は，DCPM を揮発メモリとして扱うモードであり，DRAM は DCPM のキャッシュとなる．キャッシュ制御は MC が行う．

app direct mode は，DCPM を不揮発メモリとして扱うモードである．Linux は，1) block device, 2) filesystem dax, 3) device dax の 3 つのアクセス方法を提供している [13]．DCPM を block device として利用する場合は，従来のファイルシステムをそのまま利用することができるが，block 単位でのアクセスとなり DCPM の性能を十分に引き出すことができない．filesystem dax は，dax 対応 filesystem を用いて，DCPM 領域を直接アプリケーション空間に mmap する方法である．device dax は，device dax driver から直接アプリケーション空間に mmap する方法であり，この 3 種類の方法の中で最も DCPM の性能を引き出すことができる．

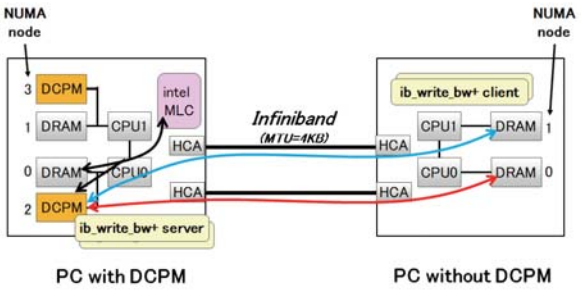

Fig. 2 Evaluation system

Linux は，DCPM を normal RAM として認識する patch も準備している [14]．この patch は device dax を利用して DCPM を NUMA node としてマウントする実装となっており，アプリケーションは Linux numactl command などを利用することで DCPM を指定した領域割り当てが可能になる．

3. 評価

3.1 概要

評価の目的は，DCPM つき PC 上で動作するアプリケーションのメモリアクセス（DRAM/DCPM）と RDMA による DCPM へのアクセスが同時に発生したときの性能干渉を明確にすることである．DCPM つき PC 上で動作するアプリケーションは intel Memory Latency Checker (intel MLC) を使用し，RDMA は Infiniband Verbs Performance Tests の中の ib_write_bw を拡張したテストツール (ib_write_bw+) を使用した．評価は，intel MLC と ib_write_bw+を同時実行した時の性能値と intel MLC 単体実行性能値及び ib_write_bw+ 単体実行性能値を比較することで行った．DRAM/DCPM に発生したスループットとレイテンシを把握する目的で，intel VTune Amplifier 2019[15] の Platform Profiler 機能を併用した．

3.2 評価環境

3.2.1 システム構成

図 2 は，今回の評価に用いたシステムの構成図である．今回の評価は，DCPM つき PC と DCPM なし PC を Infiniband 2path で接続したシステムで実施した．DCPM つき PC は，16 core Xeon Gold 5218 (Cascade Lake) x2，192 GB DRAM，812 GB DCPM (256 GB x 6) の構成である．DCPM は，6 枚の DIMM を 1 つの領域として設定し，interleaved app direct mode で OS に組み込んだ．さらに，2.3 章で説明した DCPM を NUMA node としてマウントする patch を適用した．NUMA node 0,1 を指定すると DRAM へのアクセスとなり，NUMA node 2,3 を指定すると DCPM へのアクセスとなる．OS は SUSE Linux Enterprise 15 (5.0.0-rc1-25.25) を用いた．

DCPM なし PC は，16 core Xeon E5-2650L (Sandy Bridge) x2, 32 GB DRAM の構成である．OS は Fedora30 (5.1.12-300.fc30) を用いた．

Infiniband Host Channel Adapter (HCA) は，両方の PC に ConnectX-5 を 2 枚づつ実装し，HCA 間同士を直接接続した．HCA の帯域は 1path あたり約 100 Gbpsc であり，2path 合計で 200 Gbps の帯域となる．

3.2.2 intel MLC

MLC は，version 3.7 を使用し，loaded_latency モードで測

定を行った．Read/Write は，R (Read only)，W2 (2:1 read-write ratio)，W5 (1:1 read-write ratio) の 3 種類を設定した．メモリサイズは，CPU cache の影響が無視できる 4194304 bytes とした．メモリオフセットは，ランダム (rand) とシーケンシャル (seq) の 2 種類を設定した．NUMA node は，0 (DRAM) 又は 2(DCPM) に設定して評価した．

3.2.3 ib_write_bw+

Infiniband Verbs Performance Tests 3.0 (March 2015) のソースコードを入手して実装を調査したところ，RDMA Test を行う ib_write_bw などは，同一メモリ領域に繰り返し RDMA 実行することが分かった．さらに，read/write を混在して実行するテストが存在しないことも分かった．RDMA を使用する実際のアプリケーションは，広範囲なメモリ領域に read/write 混在でアクセスを行うことが想定される．そこで我々は，ib_write_bw のソースコードをベースに以下の拡張を行った．

- RDMA を行うバッファサイズを任意の大きさに設定可能に（今回の評価では 10 MB に設定）
- RDMA の種類を Read only/Write only/ReadWrite mixed(1:1) のいずれかに設定可能に
- RDMA offset の変更方法を random/sequential のいずれかに設定可能に

上記拡張を行った ib_write_bw+を図 2 の実験システムで主に 10 多重で並行動作させて評価を行った．多重度 10 は，事前実験で DCPM に十分なアクセスを与える多重度探索を行い，決定した．

3.3 評価方法

評価は，まず ib_write_bw+単体評価と MCL 単体評価を行い，性能干渉がない場合の性能値を把握した．次に ib_write_bw+と MCL を其々の実行条件を変えながら同時実行し，性能干渉が大きくなる条件の探索を行った．

3.4 評価結果

3.4.1 ib_write_bw+のみ

DCPM つき PC 上で ib_write_bw+ server を起動し，DCPM なし PC 上で ib_write_bw+ client を起動して実験を行った．server 側の RDMA 領域は，DRAM と DCPM の両方のケースで実験した．図 3 は RDMA を行う領域を random に変更したケースの実験結果であり，図 4 は RDMA を行う領域を sequential に変更したケースの実験結果である．RDMA size は 2KB から 64KB の間で変更して測定し，Read/Write/ReadWrite mixed の 3 種類の測定を行った．

最初に random に変更したケースに関して議論する．RDMA read を実行した場合は，server 側の RDMA 領域が DRAM であっても DCPM であっても殆ど同じ性能値となった．特に RDMA size が 8KB 以上では，両者の性能値がほぼ一致していることが分かる．一方，RDMA Write と RDMA ReadWrite mixed は，server 側の RDMA 領域を DCPM にすると DRAM にした場合の 1/3 前後になることが分かる．DCPM への書き込み性能が影響していると考えられる．

次に sequential に変更したケースに関して議論する．RDMA read の実行結果は random に変更したケースとほぼ一致するが，RDMA Write と ReadWrite mixed は異なる結果となった．RDMA Write は，1 回あたりの RDMA size を大きくしていくと DRAM と DCPM の差が小さくなり，64KB で一致することが分かる．あくまでも推定のレベルであるが，Memory Controller (MC) 内の write buffer などの効果が結果に影響しているのかも知れない．ReadWrite mixed は，RDMA size を大きくすると 30 GB/sec 前後のスループット値となった．これは Infiniband は送信と受信で別々に 100 Gbps の帯域があることがその理由である．

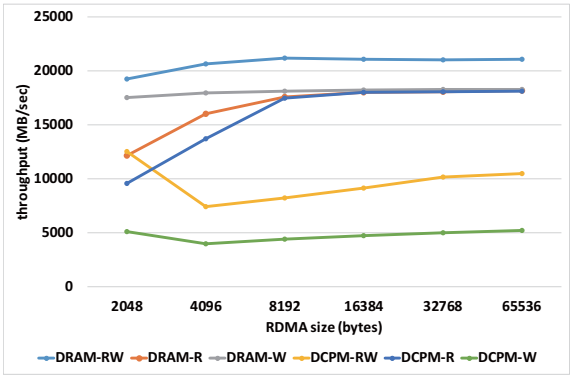

Fig. 3 RDMA random throughput (MB/sec)

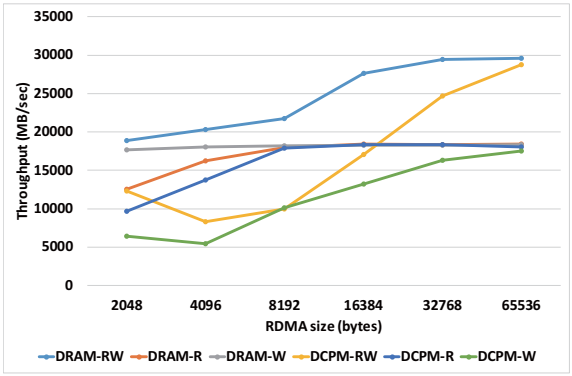

Fig. 4 RDMA sequential throughput (MB/sec)

Table 1 Throughput for MCL only (MB/sec)

	R	W2	W5
rand+DCPM	2632	2592	2326
seq+DCPM	4042	5247	5715
rand+DRAM	6733	8576	7758
seq+DRAM	12414	17213	19943

3.4.2 MLC のみ

表 1 は，3.2.2 章で説明した内容にしたがって実験した結果である．MLC は 1 スレッドで実行した．この結果より，DCPM の性能値は，DRAM を同条件で実行したときと比較して 1/3 から 1/4 になることが分かる．

3.4.3 ib_write_bw+と MLC の競合

3.4.3.1 MCL の性能

図 5 は，ib_write_bw+と MLC を同時に実行したときの MLC の性能である．X 軸は実行条件を示している．R, W2, W5 は，3.2.2 章で説明した MLC の実行条件である．ib_write_bw+による RDMA は，DCPM 領域を用いた場合と DRAM 領域を用いた場合の其々に対して行い，RDMA size は 4KB，random offset, ReadWrite mixed を選択し，10 多重で実行した．Y 軸は MLC が出力するスループット値である．凡例は，3.2.2 章で説明した MLC の残りの実行条件である．

実行結果より，DCPM 領域への RDMA と競合すると，MLC の性能が単体で実行した場合と比較して半分以下になることが分かる．MLC が DRAM にアクセスする場合でも大幅な性能低下が発生しており，MLC の実行条件が

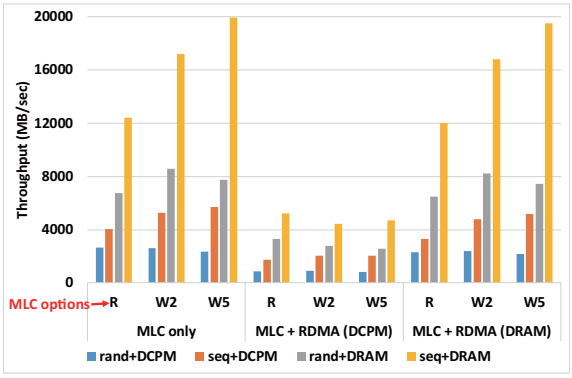

Fig. 5 MLC results on executing ib_write_bw+ 4KB,random offset,RW mixed (MB/sec)

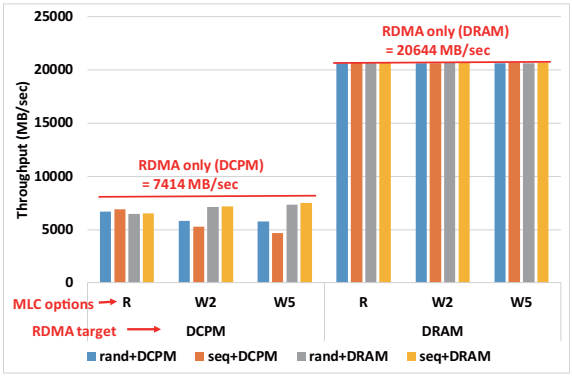

Fig. 6 ib_write_bw+ results on executing MLC (MB/sec)

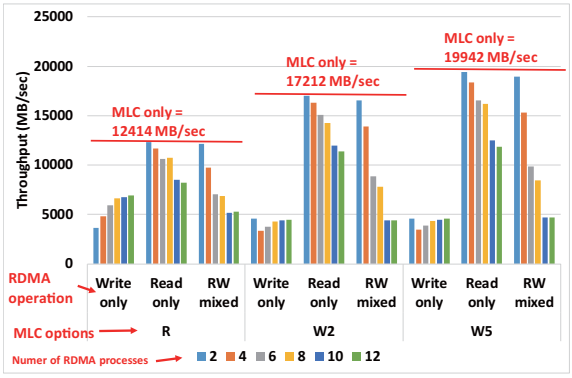

Fig. 7 MLC results with various RDMA setting (MB/sec)

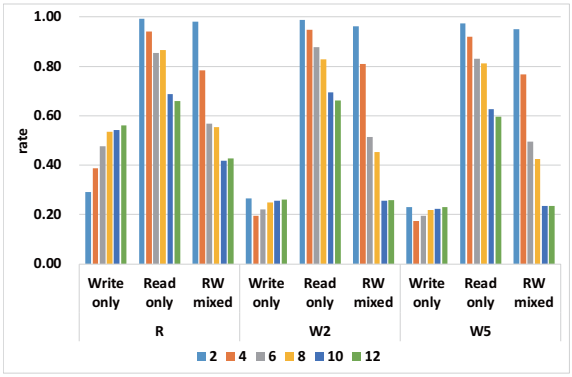

Fig. 8 MLC results with various RDMA setting (rate)

W5+seq+DCPM のときに単体実行時の 24%のスループットとなった．一方，RDMA を DRAM 領域に実行した場合の影響は小さく，最大でも 20%までの低下に留まっている．

3.4.3.2 ib_write_bw+の性能

図 6 は，ib_write_bw+と MLC を同時に実行したときの ib_write_bw+の性能である．X 軸と凡例は図 5 と同一である．Y 軸は 10 多重で動かした ib_write_bw+が出力するスループット値の合計である．

実行結果より，DCPM 領域に RDMA した場合でも 20%未満の性能低下に留まり，DRAM 領域に RDMA した場合は殆ど性能低下しないことが分かる．

3.4.3.3 ib_write_bw+の負荷が変化したときの MLC の性能

ここまでの実験結果より，DCPM 領域へ RDMA 実行と MLC 実行が競合すると MLC の性能値が大きく低下することが分かった．そこで，DCPM 領域への RDMA のアクセス量やアクセスパターンを変化させた時の MLC 実行結果への影響を調査することにした．RDMA のアクセス量やアクセスパターンの調整は，ib_write_bw+のパラメータを変更することで実施した．具体的には，これまでの RDMA ReadWrite mixed に加えて RDMA Read only と RDMA Write only を追加し，ib_write_bw+の多重度も 2 から 12 まで変更して実行した．MLC は DRAM 領域への sequential を選択し，R,W2,W5 の 3 パターンで実行した．

図 7 が実行結果である．凡例は ib_write_bw+の多重度を表している．実行結果より，R,W2,W5 の全てのケースにおいて，RDMA Write only は ib_write_bw+の多重度が小さくても MLC へ大きな性能干渉をもたらすが、RDMA Read only と ReadWrite mixed では ib_write_bw+の多重度の増加と共に性能干渉が大きくなることが分かる．図 8 は，MCL only の実行結果を 1.00 としたときの各実験結果の割合を示しており，RDMA Write による性能干渉が大きいことが分かる．最も性能干渉が大きいケースは，4 多重で RDMA Write only を実行し，且つ MLC W5 の時で，MLC only の 18%の性能となった．

図 7,8 の結果となる原因を探る目的で，intel VTune Amplifier 2019 の Platform Profiler 機能を用いて DCPM の read/write レイテンシを測定した．図 9 が read レイテンシ，図 9 が write レイテンシの測定結果である．

最初に RDMA Write only に関して議論する．レイテンシの測定結果より，ib_write_bw+の多重度が小さいとレイテンシが大きく，多重度の増加と共にレイテンシが小さくなることが確認できる．このレイテンシの増減が MCL の性能に影響を与えていると判断している．DCPM には Write だけではなく Read も発生していた．Read 側のスループット値は Write 側の 1/4 ほどであった．ib_write_bw+の実装を確認したところ RDMA Write を行う DCPM 領域は 4KB アライメントで alloc 要求を出していた．よって，コード上からは Read が発生する理由は分からなかった．

次に RDMA Read only に関して議論する．MLC を R option で動かしたときは DCPM のレイテンシはほぼ一定であり，MLC を W2 又は W5 で動かしたときは ib_write_bw+の多重度増加と共にレイテンシが若干小さくなることが読み取れる．この結果より，ib_write_bw+の多重度増加による RDMA read アクセス数の増加の影響で MLC の性能干渉が大きくなったと考えられる．

最後に RDMA ReadWrite mixed に関して議論する．Read レイテンシ Write レイテンシ共に ib_write_bw+の多重度が増えるとレイテンシが増加することが読み取れる．この結果より，ib_write_bw+の多重度増加による RDMA read/write ア

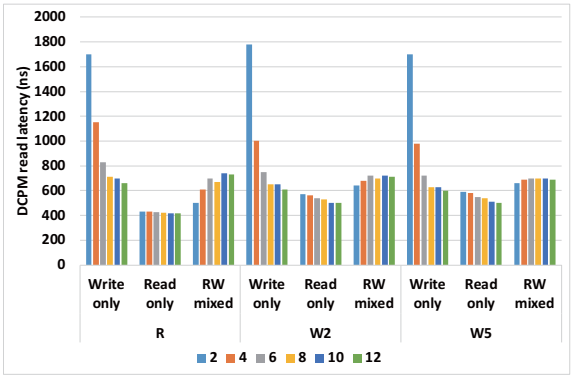

Fig. 9 DCPM read latency by using intel vtune (ns)

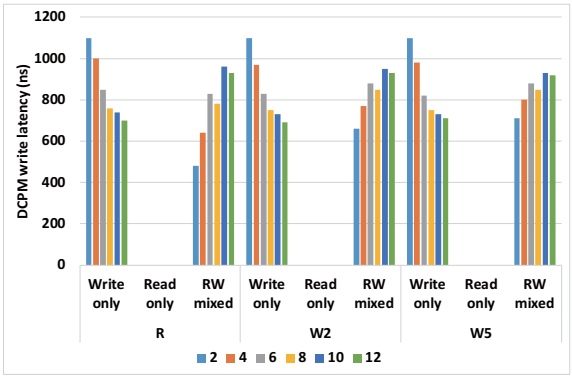

Fig. 10 DCPM write latency by using intel vtune (ns)

クセス数の増加と read/write レイテンシ増加の相乗効果の影響で MLC の性能干渉が大きくなったと考えられる．

4. 考察

3.4 章より，DCPM 領域への RDMA 実行と MLC による DRAM/DCPM アクセスが競合すると，性能干渉により MLC の性能が大幅に遅延することが分かった．その一方，RDMA 遅延は最大でも 20%程度に留まることが分かった．この結果より，DCPM つき PC の一部を RDMA を用いて他の PC と共有する運用を行う場合，ローカル PC からの DRAM/DCPM アクセスと外部 PC からの RDMA との間で何らかの排他制御が必要になる，という結論に至った．例えば，DCPM つき PC が MC のアクセス状況を監視し，ローカルからのアクセス量が小さいタイミングを検知して RDMA を起動する方法などが考えられる．この方法の探求は今後の課題としたい．

今村らの報告 [10] では，一方の MLC が DCPM へ Write アクセスを行うともう一方の MLC が DRAM にアクセスする場合でも大幅な性能低下が起きることが報告されている．大幅な性能低下が起こる理由の考察も行われており，DCPM への大量の Write アクセスが Memory Controller (MC) 内の write queue に滞留し，その影響で DRAM アクセスが遅延した可能性があることを示している．さらに，DCPM へのライトアクセスレイテンシが 1.5 micro sec 前後に達すると DRAM アクセス遅延が始まることも示されている．本論文の結果も，RDMA を用いて DCPM 側に Write を行うと MLC 側の性能が DRAM/DCPM を問わず大幅に低下することを示しており，この点では今村らの報告と一致する．しかしながら，我々の評価では DCPM への Write レイテンシが 0.7 から 0.9 micro sec に達すると性能干渉が大きくなることが異なっている．MLC からの DRAM/DCPM アクセスと RDMA Write による DCPM アクセスは，MC から DRAM/DCPM までを共有アクセスするので，MC の何らかの資源競合が原因で MLC の性能低下が起きていると考えている．資源競合が発生した箇所は，今村らの報告とは性能干渉が始まるポイントが異なるので，今村らの報告とは別の要因が含まれているのかも知れない．

5. 関連研究

DCPM の性能評価を行った論文は既に多く発表されている．Izraelevitz ら [5] は，DCPM の各機能ごとの性能を網羅的に測定している．Renon ら [6] は，database logging に DCPM を適用するための基礎評価を行っている．Hirofuchi ら [16] は，virtual machines (VMs) に DCPM を適用する目的で評価を行っている．これらの報告は，DCPM へのアクセスパターンに応じてそのアクセスレイテンシが 100 ns から 800 ns の間で変動することを示している．

6. まとめ

本論文は，DCPM つき PC 上で動作するアプリケーションと DCPM なし PC 上で動作するアプリケーションが，DCPM つき PC の DRAM/DCPM を同時アクセスしたときの性能干渉を評価した．DCPM つき PC 上で動作するアプリケーションは，Intel Memory Latency Checker (MLC) を用いた．DCPM なし PC 上で動作するアプリケーションは，Infiniband の標準ベンチマークである Infiniband Verbs Performance Tests の中の ib_write_bw を拡張して (ib_write_bw+) 用いた．

評価により，ib_write_bw+を用いて DCPM つき PC の DCPM 領域に RDMA Write を行い且つ DCPM つき PC 上で MLC を実行すると，RDMA による性能遅延が原因で MLC の性能が大幅に遅延することが分かった．一方で，RDMA 側の性能遅延は軽微であった．この評価結果より，我々は RDMA による DCPM 領域への Write アクセスとローカルで実行されるアプリケーションによるメモリアクセスの間で何らの排他制御が必要と考えている．排他制御の方法は，今後の課題としたい．

7. 謝辞

株式会社富士通研究所の今村 智史博士により，intel MLC や intel VTune Amplifier 2019 の測定方法、Linux kernel の NUMA node patch 等に関して多くの助言を頂き，さらに本論文の実験結果に関しても有用な助言を頂きました．ここに感謝致します．

富士通株式会社の佐藤 充博士より，CPU 内部構成一般，DRAM/DCPM，Memory Controller の構成等に関して多くの助言を頂き，研究を進めることができました．ここに感謝致します．

References

[1] Dulloor, S. R., Roy, A., Zhao, Z., Sundaram, N., Satish, N., Sankaran, R., Jackson, J. and Schwan, K.: Data Tiering in Heterogeneous Memory Systems, *Proc. the 11th ACM European conference on Computer systems (EuroSys 2016)* (2016).
[2] Intel: Intel Optane DC Persistent Memory. https://www.intel.com/content/www/us/en/products/memory-storage/optane-dc-persistent-memory.html.
[3] Intel: Intel Optane DC Persistent Memory Quick Start Guide. May 2019 Revision 1.0.
[4] Romik, S.: INTRODUCTION TO PROGRAMMING FOR

PERSISTENT MEMORY. May 2019.
[5] Izraelevitz, J., Yang, J., Zhang, L., Kim, J., Memaripour, A., Soh, Y. J., Wang, Z., Xu, Y., Dulloor, S. R., Zhao, J. and Swanson, S.: Basic Performance Measurements of the Intel Optane DC Persistent Memory Module. https://arxiv.org/abs/1903.05714.
[6] van Renon, A., Vogel, L., Leis, V., Neumann, T. and Kemper, A.: Persistent Memory I/O Primitives, *Proc. of the 15th International Workshop on Data Management on New Hardware (DaMoN'19), Amsterdam, Netherlands* (2019).
[7] van Renon, A., Leis, V., Kemper, A., Neumann, T., Hashida, T., Oe, K., Doi, Y., Harada, L. and Sato, M.: Managing Non-Volatile Memory in Database Systems, *Proc. of the 2018 ACM SIGMOD/PODS Conference (SIGMOD'18), Houston, TX, USA* (2018).
[8] Kimura, H.: FOEDUS: OLTP Engine for a Thousand Cores and NVRAM, *Proc. of the 2015 ACM SIGMOD/PODS Conference (SIGMOD'15), Melbourne, VIC, Australia* (2015).
[9] Andrei, M., Lemke, C., Radestock, G., Schulze, R., Thiel, C., Blanco, R., Meghlan, A., Sharique, M., Seifert, S., Vishnoi, S., Booss, D., Peh, T., Schreter, I., Thesing, W. and Wagle, M.: SAP HANA Adoption of Non-Volatile Memory, *Proc. of the 43rd International Conference on Very Large Data Bases (VLDB 2017), Munich, Germany* (2017).
[10] Imamura, S. and Yoshida, E.: The Analysis of Inter-Process Interference on a Hybrid Memory System, *IXPUG Workshop on HPC Asia 2020, Fukuoka, Japan* (2020).
[11] Intel: Vish Vishwanathan 2019, Intel Memory Latency Checker. https://software.intel.com/en-us/articles/intel-memory-latency-checker.
[12] Linux-rdma: Infiniband Verbs Performance Tests. https://github.com/linux-rdma/perftest.
[13] Goto, Y.: How to use NVDIMM in Linux (Japanese document). https://www.slideshare.net/ygotokernel/nvdimmlinuxfor-comsys-2019-ver.
[14] Hansen, D.: Allow persistent memory to be used like normal RAM. https://lwn.net/Articles/780711/.
[15] Intel: Intel VTune Amplifier. https://software.intel.com/en-us/vtune.
[16] Hirofuchi, T. and Takano, R.: The Preliminary Evaluation of a Hypervisor-based Virtualization Mechanism for Intel Optane DC Persistent Memory Module. https://arxiv.org/abs/1907.1201v1.