Vol.2016-HPC-156 No.16
2016/9/16

クラウド環境におけるキャッシュメモリQoS制御の評価

高野 了成[1,a)]　広渕 崇宏[1]

概要：Broadwell 世代以降の Intel Xeon プロセッサでは、キャッシュメモリ QoS 制御機構が実装され、ラストレベルキャッシュをスレッドや仮想マシンに対して明示的に割り当てることが可能になった。本論文では、クラウド環境における性能隔離・確保を目的に、キャッシュメモリ QoS 制御の効果を、SPEC CPU 2006 に含まれる SPECint ベンチマークを用いて評価した結果について報告する。実験の結果、仮想マシンに割り当てるキャッシュメモリを適切に調整することで最大 40%の性能向上が得られる一方で、キャッシュメモリ QoS 制御だけでは、別の仮想マシンからの干渉の影響を完全に抑制できず、10%程度性能が低下するプログラムが存在することがわかった。

1. はじめに

データセンタやエンタプライズ環境では、1 台の物理マシンで複数の仮想マシンを同時に動作させるサーバコンソリデーションが一般化している。なかでも従量課金を行うクラウド環境では、ユーザ間の公平性の観点から仮想マシン間の性能隔離は重要な課題である [1]。従来より、CPU コアやメインメモリ、ネットワーク等の I/O に関しては、仮想マシンに排他的に割り当てることで性能の干渉を最小化する試みが効果をあげてきたが、キャッシュメモリ（ラストレベルキャッシュ）についてはハードウェア支援の不備もあり、十分に検討されてこなかった。キャッシュメモリの割り当てをソフトウェアから制御できる機構は、京に搭載されている SPARC64 VIIIfx のセクタキャッシュ [2] など、HPC 向けに実用化されてきた。さらに近年、クラウド環境で一般に利用される Intel プロセッサにおいても、Broadwell 世代以降の Intel Xeon には CAT (Cache Allocation Technology) と呼ばれるキャッシュメモリ QoS 制御機構が実装され [3]、仮想マシン間の性能隔離を実現するために利用可能となった。

本論文では、仮想マシン環境におけるキャッシュメモリ QoS 制御の効果を、SPEC CPU 2006 [4] に含まれる SPECint ベンチマークを用いて評価する。実験では 1 台の物理マシン上にベンチマークプログラムを実行するターゲット VM とメモリ負荷を加えるバックグランド VM の 2 台の仮想マシンを用意し、バッググランド VM からの干渉による性能劣化、およびキャッシュメモリ QoS 制御による性能改善を計測した。その結果、ターゲット VM にキャッシュメモリを適切に割り当てれば最大で 40%の性能向上が得られた。一方で、キャッシュメモリ QoS 制御だけでは、他 VM からの性能干渉を完全に回避できず、10%程度性能が低下するプログラムが存在することも明らかになった。

本論文の構成は以下のとおりである。本実験の目的と技術的な背景について 2 節で詳しく述べる。続いて 3 節で実験環境、および実験結果を示し、4 節にて考察を行う。関連研究について 5 節で言及し、最後に 6 節でまとめを行う。

2. 目的と背景

2.1 目的

本研究で想定する環境を**図 1** に示す。1 台の物理マシン上に 2 台の仮想マシンが動作している。ターゲットアプリケーションを実行する仮想マシンをターゲット VM、もう一方をバックグランド VM と呼ぶ。バックグランド VM からターゲット VM への性能干渉を抑えることが本研究の目的である。CPU コアとメインメモリは各仮想マシンに対して排他的に割り当てられているが、ラストレベルキャッシュ (LLC) は共有している。

詳細な実験環境は 3 節で説明するが、ターゲット VM において SPECint ベンチマークを実行し、バックグランド VM ではバースト的にメモリアクセスを行うことでキャッシュを汚染するメモリストレスプログラム（cache jammer）を実行した場合の、性能低下を**表 1** に示す。hmmer や h264ref などほとんど性能に影響がないアプリケーションが存在する一方で、mcf や xalancbmk などは 35%以上の性能低下が見られる。

[1] 国立研究開発法人 産業技術総合研究所 情報技術研究部門
Information Technology Research Institute, National Institute of Advanced Industrial Science and Technology (AIST)
[a)] takano-ryousei@aist.go.jp

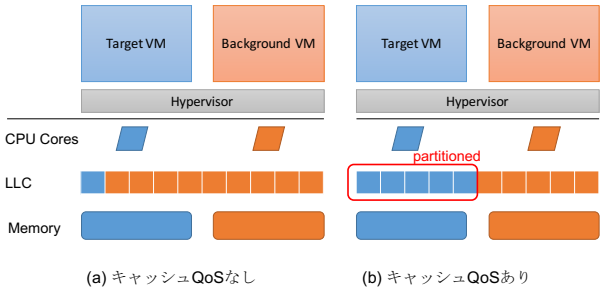


図 1 想定環境

表 1 SPECint ベンチマークに対する cache jammer の影響

Benchmarks	w/o jammer	w/ jammer	Ratio
400.perlbench	31.89	27.90	0.87
401.bzip2	18.67	15.18	0.81
403.gcc	25.91	21.49	0.83
429.mcf	31.36	19.86	0.63
445.gobmk	20.97	19.70	0.94
456.hmmer	23.26	22.91	0.99
458.sjeng	22.38	20.43	0.91
462.libquantum	52.59	46.75	0.89
464.h264ref	37.36	35.50	0.95
471.omnetpp	18.03	12.91	0.72
473.astar	16.73	14.37	0.86
483.xalancbmk	29.63	19.40	0.65

上記の問題を解決するために、図 1 (b) に示すように、プロセッサが持つハードウェアキャッシュメモリ QoS 制御を用いることで、ターゲット VM に明示的に LLC を割り当て、性能の確保を図る。

2.2 Intel RDT の概要

本研究で使用するキャッシュメモリ QoS 制御機構である Intel Resource Director Technology (RDT) [5] の概要について述べる。RDT は LLC やメモリ帯域といった共有資源を監視・管理するハードウェアフレームワークであり、Cache Monitoring Technology (CMT)、Memory Bandwidth Monitoring (MBM)、Cache Allocation Technology (CAT)、Code and Data Prioritization (CDP) 等の総称である。なかでも本研究では CAT および CMT を用いた。CAT は Broadwell 世代以降の Xeon において搭載されており[*1]、アプリケーション優先度や Class of Service (COS) に基づいて LLC を隔離することができる。

仮想マシン毎に LLC を割り当てたり、LLC 占有率、パフォーマンスモニタリングカウンタ (PMC) を取得するために pqos ユーティリティプログラム [5] を用いた。pqos を用いて CAT の設定を行う方法について以下で述べる。-e オプションで COS を定義し、-a オプションで CPU コアに COS を割り当てる。下記の例では、COS0 に 8 MB、COS7 に 4 MB の LLC を排他的に割当て、CPU コア 7 が COS7、

[*1] Haswell 世代においても、特定の SKU は CAT に対応している。

表 2 物理マシンの諸元

Hardware	
CPU	8-core Intel Xeon D-1540/2.0 GHz LLC 12 MB
M/B	Supermicro X10SDV-TLN4F
Memory	64 GB DDR4-2133
Storage	Crucial m4 CT512M4SSD2 512 GB
Software	
OS	Debian GNU/Linux 8.3
Kernel	Linux kernel 4.1.17

その他のコアが COS0 を利用するよう設定している。COS の定義はビットマップにより指定するが、本研究で使用した Xeon D 環境では 1 ビットが 1 MB に相当する。以上の設定により、CPU コア 7 とその他の CPU コアが利用する LLC は互いに隔離されることになる。

```
# pqos -e "llc:0=0xff0;llc:7=0x00f;" \
       -a "llc:0=0-6;llc:7=7;"
```

pqos プログラムでは、CMT と MBM のモニタリング結果を出力することもできる。3 節の実験では、「MISSES」列に示される LLC ミス回数と、「LLC」列に示される LLC 占有サイズを参照する。

```
# pqos -r -i 10
CORE    IPC    MISSES   LLC[KB]  MBL[MB/s]  MBR[MB/s]
   :
   0   0.41      9907      32.0        0.3        0.0
   1   0.36 190312791    8160.0    14524.3        0.0
   2   0.23      2537       0.0        0.0        0.0
   3   0.25      2563       0.0        0.0        0.0
   4   0.88     66305      32.0        2.6        0.0
   5   0.24     13501       0.0        0.1        0.0
   6   0.10     18645       0.0        0.3        0.0
   7   1.20  19505540    4096.0       89.3        0.0
```

3. 実験

3.1 実験環境

表 2 に実験に用いた PC の諸元を示す。プロセッサとして、Intel Xeon (Broadwell 世代) ベースの SoC (System-on-Chip) である Xeon D-1540 を用いた。CPU コア数は 8 であり、各コアは L1 キャッシュをデータ/命令ともに 32 KB、L2 キャッシュを 256 KB、L3 キャッシュ (LLC) を 1.5 MB 持ち、各コアはリングバスで接続される。したがって LLC の総量は 12 MB となる。Hyper threading は無効にした。なお、Debian GNU/Linux 8.3 の標準カーネルでは、Xeon D 固有のパフォーマンスカウンタを取得できないため、Linux kernel 4.1.7 を用いた。

続いて**表 3** に実験に用いた仮想マシンの諸元を示す。本実験では、ターゲット VM とバックグランド VM として

表 3　仮想マシンの諸元

Hardware	
CPU	QEMU Virtual CPU version 2.1.2
Memory	4 GB
Storage	QEMU QCOW Image (v3) 40 GB
Software	
OS	Debian GNU/Linux 8.3
Kernel	Linux kernel 3.16.0-4-amd64

表 4　SPECint ベンチマーク

Benchmarks	Description
400.perlbench	PERL Programming Language Compression
401.bzip2	Compression
403.gcc	C Compiler
429.mcf	Combinatorial Optimization
445.gobmk	Artificial Intelligence: go
456.hmmer	Search Gene Sequence
458.sjeng	Artificial Intelligence: chess
462.libquantum	Physics: Quantum Computing
464.h264ref	Video Compression
471.omnetpp	Discrete Event Simulation
473.astar	Path-finding Algorithms
483.xalancbmk	XML Processing

同一構成の仮想マシンを各 1 台準備し、taskset コマンドを用いて、使用する物理 CPU コアをそれぞれの仮想マシンに対して排他的に割り当てて起動した。メインメモリも重複しない領域から 4 GB づつ割り当てた。

3.2　ベンチマークプログラム

ターゲットアプリケーションとして、SPEC CPU 2006 に含まれる SPECint ベンチマークプログラムを使用した。**表 4** に SPECint に含まれるプログラムを示す。コンパイル手法（チューニング）は base、測定メトリックスは speed を用いた。スコアは 3 回実行したときの中央値を用いた。また、cache jammer として、下記のオプションを指定して stress コマンドを実行した。スレッド数は 1、64 MB のメモリを malloc し、キャッシュラインサイズの 64 バイトごとにストライドアクセスする。

```
% stress -m 1 --vm-keep --vm-bytes 64M --vm-stride
    64
```

3.3　予備実験：SPECint 性能と LLC サイズの関係

予備実験として、SPECint 性能と LLC サイズの関係について調べた。本実験に限り、バックグランド VM は起動せず、ターゲット VM のみ起動した状況で測定を行った。CAT を用いてターゲット VM に割り当てる LLC サイズを 10 MB から 2 MB に 2 MB 刻みで変更した。ターゲットアプリケーション以外のプロセス実行は無視できるので、CAT 無効時には、ほぼすべての LLC をターゲットアプリケーションが占有していると推測する。

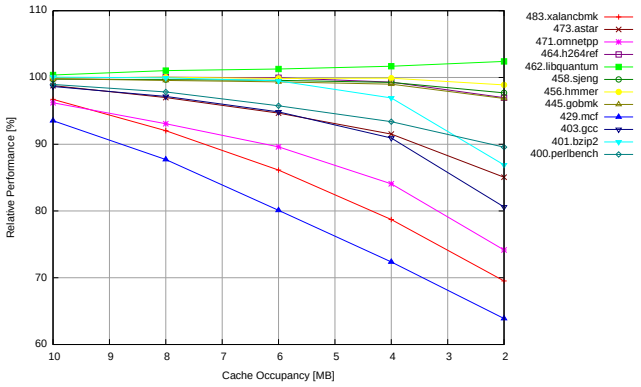


図 2　ターゲット VM が利用可能な LLC サイズを縮小したときの相対性能

表 5　物理マシンと仮想マシンの性能比較

Benchmarks	PM	VM	Ratio
400.perlbench	32.20	31.89	0.99
401.bzip2	18.74	18.67	1.00
403.gcc	25.37	25.91	0.98
429.mcf	32.37	31.36	0.97
445.gobmk	21.05	20.97	1.00
456.hmmer	23.33	23.26	1.00
458.sjeng	22.72	22.38	0.98
462.libquantum	54.05	52.59	0.97
464.h264ref	37.30	37.36	1.00
471.omnetpp	18.89	18.03	0.95
473.astar	17.24	16.73	0.97
483.xalancbmk	31.56	29.63	0.94

図 2 に CAT 無効時と比較した相対性能を示す。キャッシュサイズに対する影響度合いはまちまちであるが、概して利用可能なキャッシュサイズが減少すると性能が低下する。例外として、462.libquantum では僅かながら性能が向上している。この原因については 4 節にて考察する。また、物理マシンでも同様の実験を行ったが、傾向に差は見られなかった。**表 5** に、LLC サイズが 12 MB のときの物理マシンと仮想マシンの性能差を示す。SPEC CPU に含まれるベンチマークは CPU インテンシブなプログラムなので、概して仮想化によるオーバヘッドは無視できる。

3.4　キャッシュメモリ QoS 制御の効果

キャッシュメモリ QoS 制御の効果を確認するため、バックグランド VM で cache jammer を起動しつつ、ターゲット VM に割り当てる LLC サイズを 10 MB から 2 MB まで 2 MB 刻みで変更し、SPECint の性能を測定した。

結果を**図 3** に示す。キャッシュメモリ QoS 制御の効果が明白に確認できる。特に 486.xalancbmk は 40%以上の性能向上が得られている。LLC 割当てを縮小すると性能が低下し、アプリケーションによっては LLC が 4 MB 以下辺りからキャッシュ割当てが少なすぎるために、QoS 制御

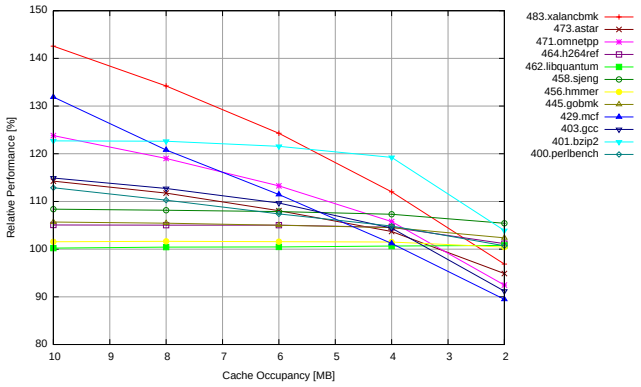


図 3 キャッシュメモリ QoS 制御の効果（QoS 制御無効時と比較した相対性能）

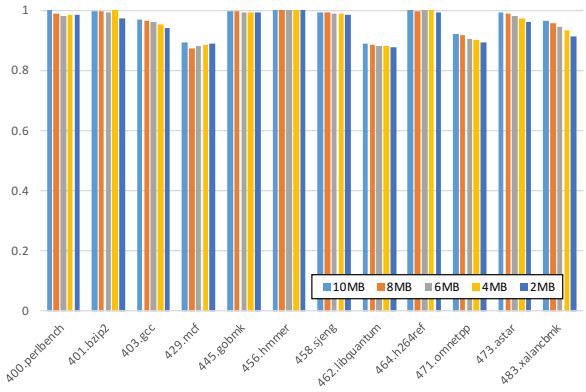


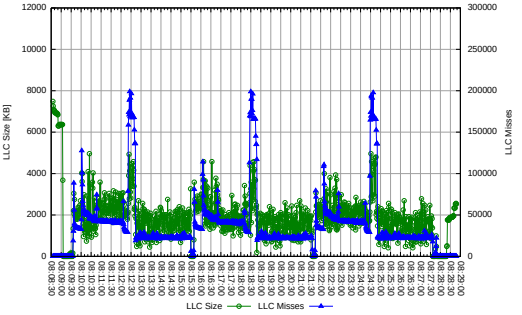
図 4 キャッシュメモリ QoS 制御による性能隔離（バックグランド VM 未動作時と比較した相対性能）

無効時よりも性能が低下する。

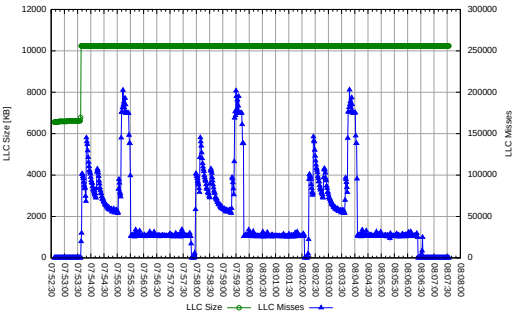
仮想マシン間において LLC を含む資源を完全に分割できたとすると、理想的には、ターゲットアプリケーションは cache jammer による性能劣化を受けないはずである。これを確認するため、バックグランド VM の有無の違いによる相対性能を **図 4** に比較する。429.mcf、462.libquantum、471.onnetpp は 1 割程度性能が低下しており、cache jammer の影響を受けていることがわかる。この原因については、4 節で考察する。

3.5 キャッシュ占有率とキャッシュミスの関係

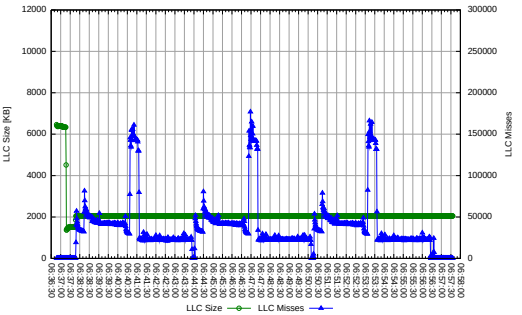
CMT によるキャッシュ占有サイズと PMC による LCC ミス回数を qpos コマンドを用いて計測した。まず、もっともキャッシュメモリ QoS 制御の効果が大きかった xalancbmk の場合の結果を**図 5** に示す。横軸は経過時間、縦軸は秒間の LLC 占有サイズおよび LLC ミス回数である。同じベンチマークを 3 回連続して実行した結果を示しているため、同じパターンが 3 回現れている。なお、横軸は経過時間を示しているがキャッシュメモリ QoS 制御が無効の場合 (a) と有効の場合 (b) および (c) で若干スケールが異なることに注意されたい。なお、(b) はターゲット VM に対して 10 MB のキャッシュを占有し、(c) は 2 MB を占有した



(a) QoS 制御なし



(b) QoS 制御あり（キャッシュ 10 MB 占有）

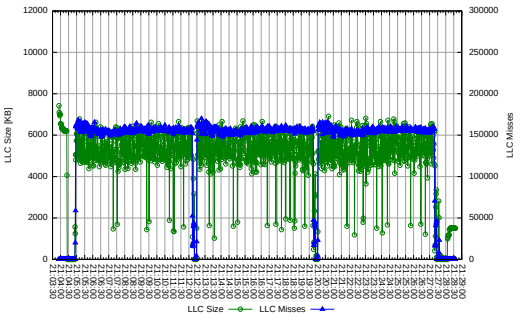


(c) QoS 制御あり（キャッシュ 2 MB 占有）

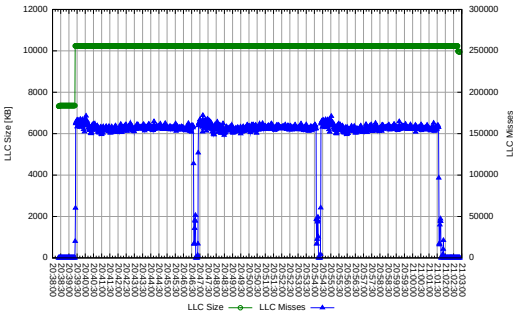
図 5 キャッシュメモリ QoS 制御による性能隔離: xalancbmk

場合の結果である。3 者でキャッシュミスのパターンは類似しているが、キャッシュの占有サイズは大きく異なる。(a) は、cache jammer が LLC を利用するためターゲットアプリケーションが利用できる LLC が大きく変動しているのに対して、(b) や (c) では、常に 10 MB または 2 MB の LLC がターゲットアプリケーションに占有されていることがわかる。さらに全体の LLC ミス回数は、(a) が 45.4 G 回、(b) が 44.8 G 回と、キャッシュメモリ QoS 制御を行うことで約 560 M 回削減されている。この結果として実行時間が 5 秒短縮した。一方、(c) の LLC ミス回数は 46.8 G 回とこれらを上回っており、結果として実行時間の増加を導いている。これは必要以上に LLC 割り当てを削減したことによりキャッシュの競合が増加したことが原因と考えられる。

続いて、キャッシュメモリ QoS 制御の効果が小さかった libquantum と hmmer の結果を**図 6** と**図 7** にそれぞれ示す。libquantum の LLC ミス回数は、(a) が 209.8 G 回、

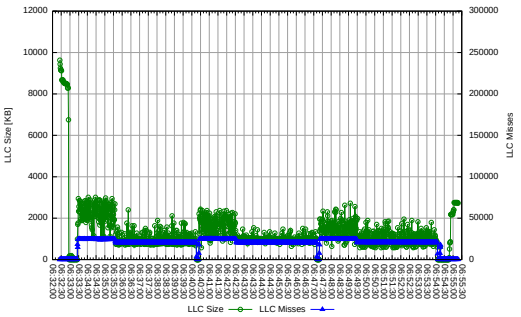


(a) QoS 制御なし

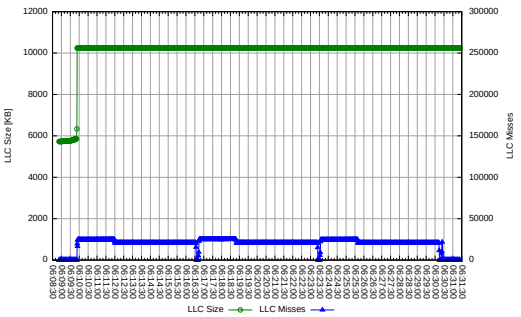


(b) QoS 制御あり (キャッシュ 10 MB 占有)

図 6 キャッシュメモリ QoS 制御による性能隔離: libquantum



(a) QoS 制御なし



(b) QoS 制御あり (キャッシュ 10 MB 占有)

図 7 キャッシュメモリ QoS 制御による性能隔離: hmmer

(b) が 209.7 G 回で、その差は 132 M 回である。(b) の実行時間は (a) より約 1 秒短縮している。一方、hmmer の LLC ミス回数は、(a) が 27.4 G 回、(b) が 27.3 G 回で、その差は 175 M 回あるが、実行時間に差はない。

4. 議論

図 3 より明らかなように、仮想マシン間における LLC アクセスの干渉を抑制することで最大 40%の性能向上が得られた。一方で、LLC 占有量を制限しすぎた場合は、ターゲットアプリケーションが十分な LLC を利用できず性能低下を招いてしまう。全体の性能バランスを考慮したキャッシュメモリ QoS 制御の最適化は今後の課題である。

また、図 4 で示した性能隔離の不完全性については、下記のように考えられる。CAT 機能により LLC は仮想マシンごとに隔離できたが、LLC からメインメモリへのアクセスについてはメモリバスを共有することになるので、性能の干渉を受ける可能性がある。言い換えると、CAT 機能はキャッシュミス率を制御することはできるが、キャッシュレイテンシの問題を解決することはできない。この仮定を検証するには、メモリチャネルのインタリービングを無効化し、仮想マシンのメモリを別々のメモリチャネルに接続されたメモリから確保して、実験する方法が考えられる。しかし、実験に用いた Xeon D-1540 は 2 つのメモリチャネルを有しているが、インタリービングを無効化できないため、追加実験を行うことはできなかった。本問題のさらなる追求は今後の課題とする。

図 2 において、直感に反して、libquantum は LLC サイズを少なくした方がわずかながら性能がよい。図 6 などからわかるように、キャッシュメモリ QoS 制御の有無に関わらず、LLC ミス回数は定常的に 150 K 回/秒と他のベンチマークプログラムと比較してキャッシュミスの頻度が高い。これは libquantum がキャッシュセンシティブなプログラムである証拠であるが、2 MB から 12 MB の範囲に限っては、キャッシュサイズには影響を受けていない。また、キャッシュ占有率を小さくすると、キャッシュミス回数が減少している。この結果より、キャッシュ割り当てが大きい方が、プリフェッチ等により、無駄なキャッシュの追い出しが発生し、性能が低下した可能性が考えられる。

5. 関連研究

クラウド環境における仮想マシン間の性能干渉に関しては数多くの先行研究がある [1]。なかでもキャッシュメモリに着目した研究としては、アプリケーションが提示する SLA に応じて、プロセッサの割り当てを調整する資源管理フレームワークである Q-clouds [6] や、干渉による性能低下を予測し、最適なワークロードの組み合わせを求める Caunta [7] などの研究がある。本研究と異なり、これらはキャッシュメモリのハードウェア QoS 制御機構を前提としていないため、アプリケーションに対するキャッシュメモリ割り当てを直接的に制御しているわけではない。

ソフトウェアによるキャッシュメモリ隔離手法として

は、page coloring または cache coloring がよく知られているが [8]、本研究ではハードウェア実装に着目して性能評価を行った。

6. おわりに

本論文では、仮想マシン環境におけるキャッシュメモリ QoS 制御の効果を、SPEC CPU 2006 [4] に含まれる SPECint ベンチマークを用いて評価した。キャッシュメモリを仮想マシンに適切に割り当てることで、他の仮想マシンからの干渉を抑制できることを確認した。例えば、486.xalancbmk では 40%以上の性能向上が得られた。しかし一方で、キャッシュメモリ QoS 制御だけでは、他の仮想マシンによる性能干渉を完全に避けることはできず、10%程度の性能低下の影響を受けるプログラム（429.mcf、462.libquantum、471.omnetpp）も存在した。その原因として、メモリバス等の共有資源の存在による影響が考えられる。今後は、本問題の原因を追求すると共に、CPU インテンシブなターゲットプログラムに限らず、メモリや I/O インテンシブなプログラムを対象とした実験を行う予定である。具体的には、ネットワーク仮想化（Network Function Virtualization）アプライアンスの性能確保への応用を検討している。NFV を仮想マシン上に実装する際に、キャッシュメモリ QoS 制御を適用することにより、パケット処理スループットの向上や、レイテンシの削減等の効果が期待できる。

謝辞 本研究は、文部科学省イノベーションシステム整備事業の補助による「光ネットワーク超低エネルギー化技術拠点（VICTORIES 拠点）」での研究成果の一部を用いている。

参考文献

[1] Koh, Y., Knauerhase, R., Brett, P., Bowman, M., Wen, Z. and Pu, C.: An analysis of performance interference effects in virtual environments, *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS 2007)*, pp. 200–2009 (2007).

[2] Perarnau, S. and Sato, M.: Discovering Sector Cache Optimizations on the K Computer, *Asia-Pacific Programming Languages and Compilers Workshop (APPLC)* (2013).

[3] Herdrich, A., Verplanke, E., Autee, P., Illikkal, R., Gianos, C., Singhal, R. and Iyer, R.: Cache QoS: From concept to reality in the Intel® Xeon® processor E5-2600 v3 product family, *Intl. Symp. on High Performance Computer Architecture (HPCA)*, IEEE, pp. 657–668 (2016).

[4] Henning, J. L.: SPEC CPU2006 benchmark descriptions, *ACM SIGARCH Computer Architecture News*, Vol. 34, No. 4, pp. 1–17 (2006).

[5] Intel: Intel(R) RDT Software Package, https://github.com/01org/intel-cmt-cat (2016).

[6] Ripal Nathuji, Aman Kansal, A. G.: Q-Clouds: Managing Performance Interference Effects for QoS-Aware Clouds, *Proceedings of the 5th European conference on Computer systems (EuroSys 2010)*, pp. 237–250 (2010).

[7] Govindan, S., Liu, J., Kansal, A. and Sivasubramaniam, A.: Cuanta: Quantifying Effects of Shared On-chip Resource Interference for Consolidated Virtual Machines, *Proceedings of the 2nd ACM Symposium on Cloud Computing*, SOCC '11, ACM, pp. 22:1–22:14 (2011).

[8] Zhang, X., Dwarkadas, S. and Shen, K.: Towards practical page coloring-based multicore cache management, *Proceedings of the 4th European conference on Computer systems (EuroSys 2009)*, pp. 89–102 (2009).