Vol.2017-HPC-162 No.2
2017/12/18

Skylake-SP と Xeon Phi KNL を用いた大容量キャッシュ付メニーコアの主記憶遅延特性評価

田邊 昇[†1, a)]　遠藤敏夫[†1, b)]

概要：DIMM 型 3D Xpoint の市販が 2018 年に予定されており，大容量で大遅延な主記憶が手軽に構築可能になろうとしている．本研究は Xeon Phi KNL のようなオンパッケージの高バンド幅大容量 DRAM キャッシュ(EDC)により，主記憶遅延の隠ぺいがどの程度可能なのかを高スループットに推定する方法を提案する．メニーコアにおいてはマルチスレッド時の性能予測が重要だが，Xeon Phi KNL にはキャッシュミスに起因する Stall を測るカウンタがない．よって従来のエミュレータは移植不可能である．そこで，KNL のメモリシステムを概ね踏襲した構成の Skylake-SP と KNL の EDC と組み合わせた想定システム上での，多様なアプリケーションに対する網羅的な遅延特性の推定について報告する．本研究と同等な評価は CPU の OoO 機構を模擬しない高速簡略型シミュレータでは再現できない．低速詳細型のシミュレータでは想定遅延 1 種あたり千年以上かかるものなので，本報告のような網羅的な評価はできない．

1. はじめに

2015 年 7 月に Intel および Micron の両社は 3D Xpoint[1]と称する新型メモリの開発成功と翌 2016 年の市場投入を発表した．3D Xpoint は DRAM の 10 倍の記憶密度，NAND 型 Flash メモリの 1000 倍のアクセス速度と書込み耐久性を有する．2018 年には DIMM 型の 3D Xpoint と，それを主記憶にできる次世代 Intel Xeon が市販されることがアナウンスされている。

ただし，3D Xpoint のアクセス遅延は Flash よりは桁違いに速いが DRAM より数倍遅いことが予測されている．IDF2015 における Intel によるプレゼン資料[2]の内容が現時点でも正しければ 250ns 程度となる見込みである．

本研究では上記のような記憶階層における劇的な変化を鑑みて，多様なアプリケーション性能へのインパクトについて評価を行う．そのような評価を行うにあたり，現段階では 3D Xpoint を主記憶とする計算インフラの実機は，筆者を含めた大半の HPC 関係者が使えない状況にある．そこで，当面は仮想的な評価環境を構築する必要がある．

現段階では MARSSx86[4][5]などのサイクルアキュレートなアーキテクチャ研究用シミュレータを用いる方法がある．その方法はあまりに実行時間がかかり過ぎで，HPC のような大規模問題に対応困難で，評価スループットが極端に低くなる．大容量メモリが対応するような大規模データを処理するアプリケーションの網羅的評価はほぼ不可能と言える．

一方，高遅延主記憶のソフトウェアエミュレータ Quartz[6]が公開されている．筆者らはその本格的使用を行うつもりで予備実験や精度検証に多大な労力を傾けた．ところが，少なくとも筆者らの環境では十分な精度が確認できない等の重大な問題があった．Quartz は古い世代の Xeon にしか対応していないなど適用性にも問題がある．

多数スレッドで動作させることがほぼ必須な Xeon Phi のようなメニーコア型 CPU を用いる環境においては Quartz のようなメモリレベル並列性の効果を考慮した高スループットな評価方法が望まれる．さらに，3D Xpoint の耐久性不足を補うために必要な DRAM キャッシュを有する構成に対応する必要がある．

本研究は上記のような課題を解決し，DRAM キャッシュを用いる設定がなされたメニーコア上でのアプリケーションの遅延感度測定法を開発する．

本研究の貢献は以下のとおりである．

(1) DRAM キャッシュが無い現在の Skylake における高スループットな主記憶遅延感度評価手法の提案．
(2) Skylake および Xeon Phi KNL 上での DRAM への Outstanding なメモリアクセス多重度の高スループットな測定法の明示．
(3) Skylake 上での Stall や等価メモリアクセス数推定用モデルの構築．
(4) (2)および(3)を用いたメモリレベル並列性効果を反映した DRAM キャッシュを設定したメニーコア上の主記憶遅延感度評価方法の提案．
(5) DRAM キャッシュを設定したメニーコア上の各種アプリケーションの遅延感度評価．

本報告の構成は以下のとおりである．まず，2 章では本報告が扱う研究対象について述べる．3 章では，既存の遅延感度測定法の代表例を概観する．4 章では，除去を検討する誤差要因について述べる．5 章では，Skylake 固有の遅延感度評価法について述べる．6 章では，Stall と Outstanding リードのモデリングについて述べる．7 章では，想定システムの遅延感度評価法について述べる．8 章では各種アプリケーションの主記憶遅延特性評価について述べ，9 章でまとめる．

2. Xeon Phi KNL と遅延感度

本章では本報告が扱う研究対象について述べる．Intel より 3D Xpoint ベースの DIMM の 2018 年出荷がアナウンスされている．3D Xpoint は DRAM の約 10 倍の集積度を有するため，安価に大容量主記憶を構築できる．一方，遅延時間と耐久性は DRAM より大幅に劣る．このため，アプリケーションの遅延感度を測定する意義がある．

3D Xpoint の耐久性は DRAM のように無限とみなせない

ため，実機での使用においては DRAM 階層によって 3D Xpoint へのアクセス頻度を大幅に抑制する必要がある．つまり，DRAM キャッシュの併用がほぼ必須であると考えられる．

現在入手可能で最新の Intel Xeon Phi としては Intel Xeon Phi KNL があり，評価環境として筆者らはこれを用いることを考える．Xeon Phi KNL は 64～72 個の SIMD 拡張された Atom ベースのコアを用いる．それらのコアは Out-of-Order 型であり，バイナリレベルで Xeon と互換性があり，Linux 等の OS がブート可能である. それらに分散配置された共有 L2 キャッシュとオンパッケージに 8 チャネルで合計 16MB の高バンド幅な MCDRAM をメモリシステムとして内蔵する．この MCDRAM は 1/4，1/2 または全体の容量を Embedded DRAM Cache (EDC)に設定して L3 キャッシュとしても利用可能である．よって，3D Xpoint へのアクセス頻度を大幅に抑制する目的で必須となる DRAM キャッシュをハードウェアとして設定可能である点で注目し，本研究では評価環境としての利用を探求する．

マルチコア型 CPU の Xeon 系列と比較して制約的な特徴としては，Out-of-Order 等のハードウェア資源が少なく，コア単体の性能的には大幅に劣る．このため，KNL の利用においては Xeon 系より多くのスレッドを走らせないで利用すると性能面で負けてしまうことが多い．また，KNL の性能カウンタは Xeon 系とは互換性に乏しく，KNL にはあって Xeon 系には無い物や，その逆のケースもある．

3. 既存の遅延感度測定法

本章では既存の代表的な主記憶遅延影響予測手法について述べる．

3.1 ソフトウェアシミュレータ TaskSim

BSC および Samsung のチームは STT-MRAM ベースの主記憶遅延影響の評価[8]を 2016 年に発表した．その際には厳選した HPC アプリケーションのメインループについて，インストゥルメントツール Valgrind を用いてある程度フィルタリングされたメモリアクセストレースを抽出し，ソフトウェアシミュレータ TaskSim[9]の速度優先モードである mem モードに投入している．この mem モードはキャッシュのシミュレートはしているが，Out-of-Order(OoO)パイプラインをシミュレートしておらず正確性に欠ける．その実行時間の大きな割合を占めると思われる TaskSim の mem モードは実機の 1561 倍の時間がかかることが報告[9]されている．疎行列処理のような入力データ依存のカーネルをこの手法で評価しようとすると，例えば実機で 2 日間かかるようなアプリケーション群の評価は TaskSim の実行時間だけで約十年間かかってしまう．実際にはメモリアクセストレース抽出は TaskSim の mem モード実行以上に時間がかかる可能性が高く，この手法を多様かつ大きな疎行列処理等の評価に用いるということは現実的ではない．

3.2 ハードウェアエミュレータ HMEP

上記のような実行時間の問題が無いのがハードウェアエミュレータである. リアルタイムで動作する Intel のハードウェアエミュレータ HMEP を用いた先行研究[10]-[14]がある．ただし HMEP は Intel 関係者のクローズドな評価環境であり, 一般の研究者が容易に利用できるものではない. 遅延可変域は 300ns～500ns と狭い. 設定した遅延により実際に性能が低下してしまうので評価の能率は必ずしも最高ではない．大規模グラフ解析やデータベースや OS の研究に適用した研究はあるが，HPC 系アプリに適用した例は見当たらない. このアプローチは CPU ごとにハードウェアを開発する必要があるが，本研究が対象とする Xeon Phi の後継機に対応するハードウェア式エミュレータが開発されたという情報は現在のところ見当たらない.

3.3 ソフトウェアエミュレータ Quartz

リアルタイムで動作する評価環境としてはソフトウェアエミュレータ Quartz があり git にて公開されているため関係者以外でも試すことができる．ただし，特定型番の Xeon(SandyBridge，IbyBridge，Haswell)の 2 ソケットサーバのみサポートされている．筆者らの Haswell ベースの測定環境では残念ながら精度も低く，動作が不安定であった．実験ごとに初期時間が 5 秒ほどかかり，短時間のアプリの評価を大量にこなすには能率が悪い.

Quartz は原理的には L2 ミスに伴う CPU ストールの性能カウンタからメモリアクセス分を予測する性能モデルを用いて，メモリ遅延増加に伴うストールを予測し，人工的に遅延を挿入する．ところが，本研究が対象とする Xeon Phi KNL には対応するストール関係の性能カウンタが存在しないため，その戦略での Quartz 移植はできない.

4. メニーコア向けに除去したい誤差要因

メニーコアでありがちな利用状況(多数のスレッドで実行してバンド幅の影響も大きい状況)下においては OoO やメモリレベル並列性の効果を無視すると誤差が増大するおそれがあるため，検討が必要である.

Skylake においては L2 ミスおよび L3 ミスに伴う Demand リードの投機的実行数を累積カウントする性能カウンタが存在する．そのカウント値を実行サイクル数(Elapsed time 換算)で除算したものがその期間の平均 Outstanding リード数となる．予備評価としてサーバ向けではない 4 コア(Hyper Thread 数は各コア 2) の Skylake(Core i7-6700K,4.00GHz)を用いた環境で NPB CG および graph500 参照実装の omp-csr を実行した際のスレッド数を変化させた時の平均 Outstanding リード数の変化を 図 1 に示す. 1 スレッド実行の場合は DRAM への Outstanding リード数が非常に少なく MLP の効果を無視しても誤差が小さいことが判る．しかし，スレッド数を増やしていくと誤差が大幅に増える．本評価環境はメモリチャネル数が多いサーバでは

ないためメモリバンド幅が小さい上にコア周波数が高いため，少ないスレッド数で Outstanding リード数が増加しやすい分，現象がより明瞭に観測できている．

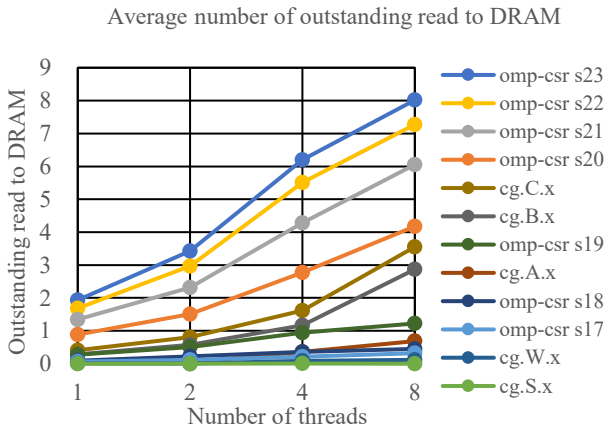

図 1 Skylake(Core i7-6700K) 上の NPB CG および graph500 実行時の Outstanding リード数の時系列変化

5. Skylake 固有の遅延感度評価法

本章では汎用な簡易法に触れたのちに，主に Skylake 世代の CPU 固有のより正確な主記憶遅延影響予測手法について述べる．

5.1 汎用で簡易的な遅延特性評価法

適用可能機種の汎用性を優先する場合，Linux の perf コマンドの pre-defined event である cache-misses を用いることでストールを発生させる可能性のあるメモリアクセスの回数を特定し，近似的に遅延特性を評価することが可能である．その測定用 perf コマンドは以下のとおりである．

perf stat -e cache-misses アプリコマンド

ただし，上記の手法はメモリレベル並列性や OoO の効果を無視した近似法であるため，主に Outstanding リード数が少ないシングルスレッド実行時でしか，正確な挙動をとらえられないと考えられる．メモリ遅延増加のみによるアプリケーションの性能変動は把握できるが，マルチスレッド時に発生しがちなバンド幅ネックやメモリレベル並列性上昇に伴うメモリコントローラ挙動を反映した性能予測はできない．

5.2 L3 ミスに伴うストール数ベースの遅延特性評価法

メモリレベル並列性や OoO の効果を反映した遅延特性評価を行なうためには，メモリアクセスに伴うストール数を観測または算出する必要がある．メモリ遅延と，そのメモリ遅延における Last Level Cache(LLC) ミスに伴うストールサイクル数と数がわかれば，それを元にメモリ遅延が延びた場合のストール数を推定することができる．Quartz の場合は Haswell 世代までのサポートだったため L2 ミスに伴うストールサイクル数と LLC(L3)ミス数，L2 ミス数，メモリ遅延，L3 遅延から L3 ミスに伴うストールサイクル数を換算して，等価的メモリアクセス数を推定していた．

このように Skylake 以前の世代の CPU には搭載されていなかったために周り道を強いられていたが，Skylake 世代の CPU では L3 ミスに伴うストールサイクル数を直接カウントする性能カウンタが存在する．その測定用 perf コマンドは以下のとおりである．

perf stat -e cpu/event=0xA3,umask=0x06,cmask=0x06, name=STALLS_L3_MISS/ アプリコマンド

Quartz と同様に，メモリ遅延が大幅に増加した場合も OoO や MLP の効果が変動しないと仮定すれば，上記で得られた等価アクセス数に設定したメモリ遅延増加分を乗じた時間を実行時間に追加することで，アプリケーション性能のメモリ遅延特性を得ることが可能である．

5.3 Outstanding リード数測定法

Skylake の遅延特性だけ測定したい場合は上記のカウンタを用いた方法で充分である．ただし，3D Xpoint の耐久性は DRAM と比べて圧倒的に低いため，Xeon Phi KNL に搭載されているような大容量の DRAM キャッシュを併用して書き換え頻度を低下させないと長持ちしない．Xeon Phi KNL は上記の機種固有カウンタのみならず，Quartz が用いていたカウンタすら備えていない．前章の予備実験のように内部の挙動を調査する場合や，存在しないストール数を予測する場合，L3 ミスに伴う Outstanding リード数をカウントが有用と思われる．なぜなら，原理的に Outstanding リード数とストール数の間には高い相関関係があると予想されるためである．

前章の予備実験でも使用された方法であるが，Skylake 世代の CPU では L3 ミスに伴う Outstanding リード数をカウントする性能カウンタが存在する．その測定用 perf コマンドは以下のとおりである．

perf stat –e cpu/event=0x60,umask=0x10, name= OUT_L3miss_Dem_RD/ アプリコマンド

6. Stall と Outstanding リード数のモデリング

本章では Skylake-SP におけるストール数と Outstanding リード数の相関に関する予備実験と，それに基づくモデリングについて述べる．

6.1 ストール数と Outstanding リード数の相関

前章に示した方法により Skylake 上ではストール数と Outstanding リード数の双方を取得することが可能である．一方，Skylake 世代のサーバータイプの CPU である Skylake-SP は Xeon Phi KNL と極めて類似度が高いオンチップメモ

リシステムを有する．そこで，Xeon Phi KNL におけるストール数の予測を念頭に入れつつ，測定環境として Skylake-SP を用いて予備実験を行なった．表 1 に予備実験で用いた測定環境の仕様を示す．

表 1　測定環境 Skylake-SP の仕様

CPU name	Intel(R) Xeon (R) gold 6140 (Skylake-SP)
Frequency	Max :2.3GHz (1.4GHz に実質固定して測定)
#CPU processors	18 cores x 2HTs x 2sockets (HT は disable に設定)
Cahce	L1: 32KB/core, L2: 1MB, L3(分散配置):24.75MB
DIMM	Type: DDR4-2666(DIMM)×6ch×2sockets Latency: 平均 115ns,Loacal82.2ns,Remote147.7ns (1.4GHz 時), Capacity: 192GB
COMPILER	gfortran version 4.8.5 20150623 gcc version 4.8.5 20150623

使用した Skylake-SP 自体は本来 2.3GHz で動作可能であるが，ガバナー設定を powersave，最小周波数設定を Xeon Phi KNL と同じ 1.4GHz とした時の測定結果を示す．このような設定により，過熱しない限り安定的に 1.4GHz 付近に固定して Skylake-SP を動作させることができる．

予備測定に用いたアプリケーションと測定条件を表 2 に示す．

表 2　評価用アプリケーションの仕様

Suite	Kernel	Size	#of threads	Note
NPB3.3.1	BT	Class S, W, A, B, C	1, 2, 4, 8, 16, 32	Compiled with gfortran or gcc using AVX512
	CG			
	EP			
	FT			
	IS			
	LU			
	MG			
	SP			
	UA			
GAP	BC	Scale = 17, 18, 19, 20, 21, 22, 23		Compiled with gcc
	BFS			
	CC			
	PR			
	SSSP			
Graph500 2.1.4	omp-csr			

全てのベンチマークアプリケーションにおいて，カーネルごとに全サイズ全スレッド数の測定点について L3 ミスに伴う Outstanding リード数とストールサイクル数のカウンタ値の散布図を作成すると，図 3 の例のように概ね直線上にプロットされる．表 3 に示すように相関係数 R は殆ど 0.99 以上であり，両者の間には極めて強い相関があり，それはサイズやスレッド数には殆ど依存しない．ところが傾きについては表 3 に示すようにカーネルごとに明らかに異なる値を示している． 切片は殆ど 0 に近似できるレベルにある．よってアプリケーション毎の上記傾き(Slope)のモデル式を導き出し，それを用いることで，スレッド数 N_{thread} における時間軸上の重なりの補正を加えると，以下の式(1)によって L3 ミスに伴う Outstanding リード数積算値(Accumulated_Outstanding_Read)から N_{thread} 実行時の L3 ミスに伴うストール数(Stall)を予測できると考えられる．

$$Stall \fallingdotseq Slope * Accumulated\_Outstanding\_Read/N_{thread} \quad (1)$$

Quartz において用いられている MLP の効果を反映した等価 NVM アクセス数(NVM_access)を式(2)で表される．これに想定するメモリ遅延の増加分を乗ずることでアプリケーションの実行時間増加分を推定することが可能である．

$$NVM\_access \fallingdotseq Stall / DRAM\_latency \quad (2)$$

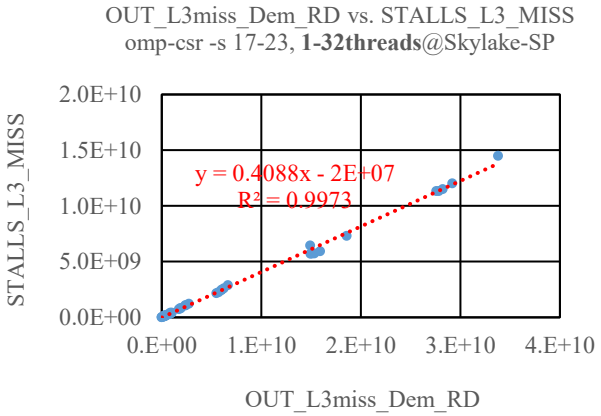

図 2　Skylake-SP 上の GAP BFS 実行時の L3 ミスに伴う Outstanding リード数積算値と Stall 積算値の関係(1.4GHz, Scale=17,18,19,20,21,22,23，スレッド数 1,2,4,8,16,32)

6.2 ストール数への変換用傾きのモデリング

アプリごとに決まる傾き(Slope)に影響があると考えた異なる測定項目の 3 種類説明変数(EV1,2,3)の候補の 32 スレッドにおける値を表 3 に示す．

表 3　モデリングのための予備評価結果

Benchmark	R	Slope	EV1	EV2	EV3
npb-bt	0.9993	0.77	2.7	298577164	64.0
npb-cg	0.9988	0.49	11.1	992851802	12.6
npb-ep	0.9989	0.66	0.0	217455	16.0
npb-ft	0.9993	0.15	7.5	382609959	12.8
npb-is	0.9976	0.84	0.3	231422234	1.8
npb-lu	0.9962	0.39	5.1	346388642	42.7
npb-mg	0.9975	0.49	2.9	1388638566	4.8
npb-sp	0.9903	0.73	3.5	1108632086	60.8
npb-ua	0.9892	0.80	1.2	536968054	53.0
gap-bfs	0.9987	0.41	3.0	183302564	8.1
gap-bc	0.9970	0.37	18.4	322837488	35.8
gap-cc	0.9980	0.36	17.8	362820073	26.7
gap-pr	0.9949	0.36	27.7	480166392	33.8
gap-sssp	0.9987	0.41	17.9	326147275	22.6
omp-csr	0.9984	0.46	15.6	256366683	91.7

傾きはスレッド数に依存しないので，今回の計測の中で最も測定時間が短い 32 スレッドにおける測定値を用いて

いる．候補として選択した説明変数 EV1 は最大サイズにおける Outstanding リード積算値/Elapsed time，EV2 は LLC ミス回数/ Elapsed time，EV3 は Elapsed time である．

表 3 のデータに対して Excel のデータ解析アドイン上で重回帰分析を実行すると以下のような傾きのモデル式の係数が得られる．

Slope＝-1.51E-02*EV1 + 1.97E-11*EV2 ＋ 2.42E-03*EV3 + 5.58E-01

≒-1.51E-02*EV1+2.42E-03*EV3+ 5.58E-01　　(3)

EV2 については遅延特性には重要な意味を持つものであるが，上記の傾きには殆ど相関が無いことがわかった．ゆえに相対的にけた違いに強い相関がある EV1,EV3 のみを用いる近似が可能である．図 4 は傾きの観測値とモデル式(3)から得られる傾きの予測値である．傾きが最大である NPB IS や最小の NPB FT などズレが大きいサンプルもあるが，概ねモデル式(3)での予測と観測値は合っていることがわかる．

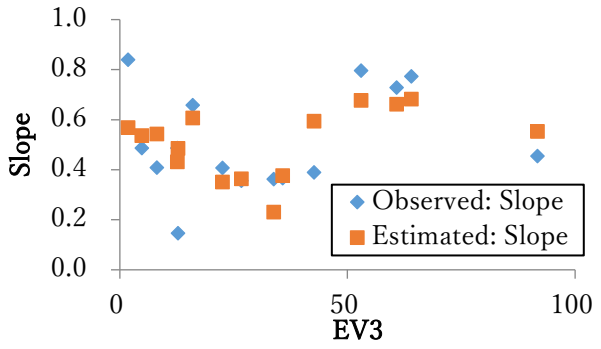

図 3　傾きの観測値とモデル式(3)による予測値

7. 想定システムの遅延感度評価法

本章では大容量キャッシュを有するメニーコア上の主記憶遅延影響予測手法について述べる．

7.1 想定システム

大容量キャッシュを有するメニーコアの実機としては Intel Xeon Phi KNL が市販されている．ところが，KNL にはキャッシュミスに伴うメモリアクセスによる Stall を測る性能カウンタが存在しない．一方，Skylake-SP には L3 ミスに伴うメモリアクセスによる Stall を測る性能カウンタが存在する．さらに，Skylake-SP のメモリシステムは KNL のメモリシステムの設計を流用していると考えられる部分が多い．両者の共通点はネットワークオンチップのトポロジー，タイル内の L2 容量(1MB)，外部 DRAM チャネル数とその仕様である．

最大の相違点は L3 キャッシュであり，KNL の場合はオフチップの MCDRAM(容量 4,8,16GB に設定可能)を用いた大容量なダイレクトマップキャッシュであるのに対して，Skylake-SP の L3 はタイル内に 1.375MB ずつ分散させた低遅延小容量キャッシュ(測定環境では 24.75MB)である点である．

そこで，現状での実機評価可能性を考慮して，本研究で性能予測を行う想定システムを**「KNL の周波数(1.4GHz)で動作する KNL のコアと EDC を有し，1 ソケットの Skylake-SP の外部メモリ関連回路に，3D Xpoint のように高遅延な DIMM を装着したもの」**とする．

なお，CPU コアが Skylake-SP と KNL では OoO の資源などで異なるが，数百 ns 単位の大きなメモリ遅延を追加する想定システムではコア内の OoO 資源の吸収可能な遅延を大幅に超えており，その差は少ないと考えられる．例えば Skylake-SP の OoO Window サイズは 224 で，それは KNL のそれよりも 3 倍程度大きいものの，サイクル毎に 6 並列で μOP は消費されるため，最悪時の遅延隠ぺい効果は Skylake-SP でも 37 サイクル(1.4GHz 動作時に 27ns)にすぎない．よって大きなメモリ遅延に対しては Skylake-SP と KNL のコアは近い挙動を示すと考えられる．

一方，メモリレベル並列性(MLP)の活用能力に関しては大きなメモリ遅延においても有効に機能すると考えられるが，その挙動はメモリコントローラから外側になるので，ソケット数を揃えれば Skylake-SP と KNL の差は少ないと考えられる．

7.2 Outstanding リード数測定法

5.1 に記載の簡易手法において無視したメモリレベル並列性を反映した実行時間評価の手段として，KNL 上の投機的なリード数の測定法を述べる．イベントマニュアル[15]によると KNL には前章で紹介した Skylake と異なり，Outstanding リードに関するカウンタは 1 種類しか取れない．

このカウンタは root 権限を必要としないため高スループットで OS ノイズを拾いにくい評価が可能である．Xeon 系とは大きく異なり，Offcore_Response Event (名称 OFFCORE_RESP，event select=0xB7)であり，Umask=0x01 とした上で，さらに MSR_OFFCORE_RESP0 レジスタに細かい測定条件を設定することで測れる．単に event select と Umask の組で指定できた他のケースと perf コマンドの形が異なるので注意が必要である．DIMM への Outstanding リードの場合以下の bit を 1 とする． Request type として bit0 (DEMAND_DATA_RD), Response type として bit23 (DRAM_NEAR), bit24 (DRAM_FAR), bit31 (SNOOP_NONE), bit32 (NO_SNOOP_NEEDED), Outstanding requests として bit38 (OUTSTANDING)

以上の測定方針に基づいて提案する DIMM への Outstanding リード数を測定する perf コマンドは以下のとおりである．

perf stat -e cpu/event=0xB7,umask=0x01, offcore_rsp=0x4181800001, name=OUTSTANDING_RD_DRAM/ アプリコマンド

上記のコマンドで得られる *OUTSTANDING_RD_DRAM* の値を同時に出力される Elapsed time で除算したものが DRAM への平均 Outstanding リード数となる．ここで時間

を CPU タイムである(pre-defined event である cycle)を元に算出するとマルチスレッド時に平均 Outstanding リード数を過小に評価する方向のズレが出る．メモリコントローラ上での混雑度を反映するには CPU タイムではなく，実際にかかった時間である Elapsed time を基準とすることが適切である．

なお，perf コマンドは-I オプションを用いることで刻み幅は 100ms 以上という制約があるものの Quartz のように小期間に区切ってカウント値を記録することが可能である．

7.3 想定システム上の MLP 効果付遅延感度測定法

Skylake-SP に KNL の EDC を追加した想定システムにおける，メモリレベル並列性を反映した実行時間評価方法を提案する．具体的には，前章の Skylake-SP 挙動のモデリングと前節の KNL 上での Outstanding リード数測定法を組み合せる．

想定システムの挙動の予測において，想定システムの Last Level Cache である EDC の挙動(EDC ミスの発生挙動やそれに伴う DIMM への Outstanding リード数に対応)は EDC を有する KNL 上の測定データを元にすべきである．なぜならば，25MB 足らずの Skylake-SP の L3 と 4〜16GB の KNL の L3 は容量差が大きすぎるため，前者から後者の L3 ミス数を推定するのは困難だからである．

一方，EDC にミスした後の挙動は同様な構成のメモリコントローラ群の性能なので，そこでさばかれる MLP の効果は Skylake-SP 上のデータを元にした前章記載の性能モデルで近似可能と考える．ここで言う MLP の効果とは，アプリケーション固有のアクセスパターンで外部メモリに対する Outstanding リード要求が与えられた時に，Stall を基準とした等価的なメモリアクセス数がいくつになるのかという関係性を意味する．その一例が前章の式(1)(2)(3)で表現されており，提案ではこれらで想定システムの MLP の効果を近似する．

より正確には，15 系統のサンプルの重回帰分析で得られたモデル式(3)ではなく，予測を行いたいアプリケーションによる Skylake-SP 上での Stall と Outstanding リード数の実測値を用いることが望ましい．モデル作成に用いたアプリケーションとメモリアクセスの特徴が大きく異なるアプリケーションではモデルでの予測誤差が大きい可能性がある．

さらに正確性を期すには前述の-I オプションで Outstanding リード数の時系列的な変動を観測し，そのムラがどれ位あるかを確認すると良い．メモリアクセスが少ない期間が全体を多く占め，Outstanding リード数が跳ね上がる期間が短い場合，全実行時間の平均で評価することは誤差を取り込むので，時系列での観測値を用いるべきである．

7.4 提案手法の手順のまとめ

提案手法を用いる場合の手順の概要を以下にまとめる．

(1) Skylake-SP 上で OutstandingRD と Stall を測る(詳細は 5.2 および 5.3 参照．–e 部分を併記して同時測定可)
(2) KNL 上で同じアプリで OutstandingRD と Elapsed time のみ測る (詳細は 7.2 参照)
(3) モデル式(3)に KNL 上の OutstandingRD と Elapsed time を代入して傾きを得る（モデル誤差を避けたい場合は Skylake-SP の Stall 実測値から傾きを得る）
(4) Skylake-SP での Elapsed time を KNL での Elapsed time で割った値を乗じて想定システム用の傾きとする
(5) 想定システムの傾きを KNL 上の OutstandingRD に乗じ，スレッド数で割って想定システムの Stall を得る
(6) 想定システムの Stall と DRAM 遅延を式(2)に代入し，等価外部メモリアクセス数を得る
(7) 想定する NVM 遅延と DRAM 遅延の差を等価外部メモリアクセス数に乗じて追加時間を得る

7.5 提案手法の正当性検証

EDC を有する KNL は 2 ソケットシステムを構築可能に設計されていないため，上記の提案測定法の正当性は Quartz 論文と同様の 2 ソケットを用いた実験による検証が原理的にできない．よって手法を構成する各部分の検証を行うのが現実的である．

式(1)は図 3 に代表される実験により直線の式となることが検証済みである．式(3)は図 4 において検証済みである．ただし，検証範囲はモデル作成に用いたアプリケーションに限定される上に誤差が大きい物も少数ある．よって他のアプリケーションにはモデルの誤差を許容しつつ用いるか，Skylake-SP 上の実測で得られる傾きの値を用いることでこのモデルの潜在的誤差は避けられる．

式(2)自体は Quartz の論文において遅延挿入や L2 向けカウンタからモデルによる L3 への変換等に伴う誤差も込みで検証済みのものである．さらに，想定システムの LLC ミス以降の動作は Skylake-SP は一致させている．

よって，Quartz より誤差が入り込む余地は少ないものの Skylake-SP 固有の Stall カウンタを用いた追加遅延算出値の検証が残されている．この予測値と，2 ソケットの Skylake-SP においてメモリ領域がアプリケーションを実行する CPU コアと同じソケットにある場合と，異なるソケットにある場合の速度低下の差について検証を行った．表 2 の全アプリおよび Stream における誤差を図 4 に示す．DRAM 遅延のローカル(同ソケット)とリモート(別ソケット)は Intel Memory Latency Checker による実測値を用いた．メモリとコアのソケットへの固定は hwloc-bind によって行なった．Stall カウンタによる予測と実測の誤差は全体の平方二乗誤差は 10.7%だった．ただし，Stream や NPB CG などバンド幅ネックになりやすいアプリでは UPI(ソケット間リンク)バンド幅が Roofline になったことによる．よって大きく負の値になっている結果は提案手法の誤差を意味しないため排除すべきである．誤差が-10%以上のアプリを除いた場合の誤差は 6.0%であり，実用範囲にあると言える．

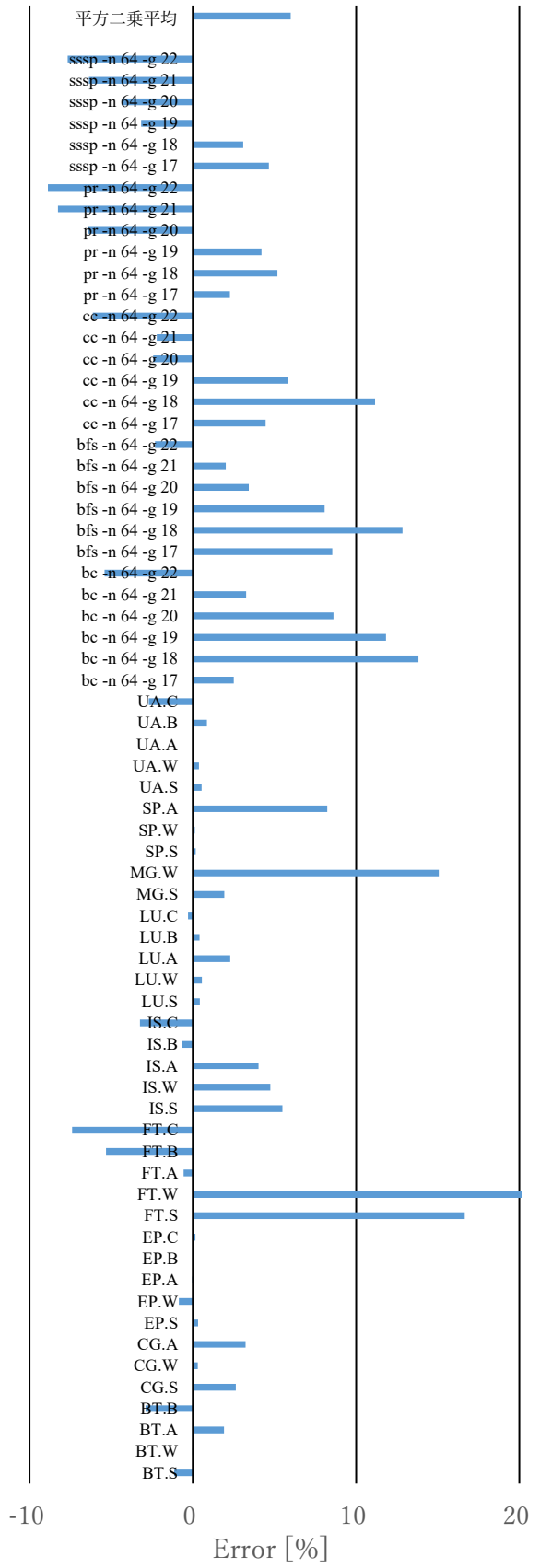

図 4　Skylake-SP 上での遠隔メモリ利用による性能低下の予測誤差 (注：UPI バンド幅ネックにより大きく負の値になったアプリは，誤差確認法の限界のため除外)

8. 主記憶遅延特性評価

提案手法を応用した主記憶遅延増加時のアプリケーション性能の評価を実施した．本章ではその実験内容と結果について述べる．

8.1 測定環境

表 4　に KNL を用いる実験で用いた測定環境の仕様を示す．Skylake-SP を用いる実験で用いた測定環境の仕様は表 1 に示したものと同様である．DRAM 遅延，キャッシュミス時に発生する 1Read+1Write のアクセス比率における DRAM 最大バンド幅は Intel Memory Latency Checker による実測値を示した．

表 4　測定環境の仕様

CPU name	Intel(R) Xeon Phi(R) 7210 (KNL: Knights Landing)
Frequency	Base:1.3GHz, Turbo boost:1.5GHz
#CPU processors	64 cores x 4HTs (1 socket , HT は enable に設定)
Cahce	L1: 32KB/core, L2: 32MB(分散配置), L3(EDC):8GB
MCDRAM	MAX 16GB (この半分の 8GB を EDC に設定)
DIMM	Type: DDR4-2133(DIMM)×6ch, Latency: 175ns, Capacity: 192GB Bandwidth: 1Read+1Write : 70.7GB/s / MAX 102GB/s
COMPILER	Intel icc, icpc, ifort ver. 17.0.1 gcc (GNU C Compiler) version 4.8.5

8.2 実験方法

本章の実験では表 2 のワークロードについて，想定システムにおける主記憶遅延特性評価を行なった． なお，前章の精度評価におけるローカルにメモリを割り当てた実験と遅延感度特性評価における Skylake-SP を利用して測定する部分は同時に実施できる．

KNL が 1 ソケットであるため，Skylake-SP 上の Slope の測定は Hyper Threading(HT)を用いない設定で 1 ソケットの物理コアに収まる 2 の累乗の最大値である 16 スレッドで行なったものに関してのみ結果を示す．本実験では誤差が入り込む可能性のある式(3)のモデルは用いていない．

8.3 実験結果

表 5　に提案手法による 7.1 に記載の想定システムのメモリ遅延による性能低下率（Slowdown）と，その算出に用いられた Slope 値を示す．濃い赤系のセルほど性能低下率が高いように彩色してあり，一種のヒートマップとなっている．これにより NPB CG や SP，GAP PR（PageRank）の大き目のサイズの遅延感度が高いことが一目瞭然である．概ね問題サイズが大きいほど，メモリ遅延が大きい程性能低下率が上昇する．逆に NPB EP や FT，GAP BFS は今回のベンチマーク中では比較的遅延感度が鈍い．

今回はスレッド数を 16 に固定してサイズやベンチマーク種類を変動させて網羅的に調査しているが，スレッド数や入力行列を多数にしたり多様な軸に関して網羅的に調査するのに十分なスループットがあることを確認できた．

表 5　提案手法による想定システムのメモリ遅延による性能低下率と Slope 値

Benchmark	Slope on Skylake-SP	Estimated Slowdown by latency			
		300ns	500ns	750ns	1000ns
NPB　BT.S	0.673	1.013	1.025	1.040	1.055
NPB　BT.W	0.640	1.096	1.185	1.295	1.406
NPB　BT.A	0.945	2.459	3.798	5.473	7.147
NPB　BT.B	0.882	2.407	3.700	5.315	6.931
NPB　BT.C	0.833	2.477	3.833	5.528	7.223
NPB　CG.S	0.654	1.083	1.159	1.254	1.348
NPB　CG.W	0.601	1.214	1.410	1.656	1.901
NPB　CG.A	0.568	4.169	7.078	10.716	14.353
NPB　CG.B	0.582	5.669	9.957	15.316	20.676
NPB　CG.C	0.524	7.272	13.032	20.232	27.432
NPB　EP.S	0.647	1.133	1.256	1.409	1.562
NPB　EP.W	0.647	1.192	1.369	1.590	1.811
NPB　EP.A	0.658	1.286	1.548	1.876	2.204
NPB　EP.B	0.668	1.300	1.575	1.920	2.264
NPB　EP.C	0.673	1.300	1.576	1.920	2.265
NPB　FT.S	0.657	1.064	1.123	1.197	1.271
NPB　FT.W	0.589	1.241	1.462	1.738	2.014
NPB　FT.A	0.162	1.342	1.656	2.048	2.440
NPB　FT.B	0.141	1.565	2.084	2.732	3.381
NPB　FT.C	0.148	2.059	3.031	4.246	5.461
NPB　IS.S	0.643	1.028	1.054	1.086	1.118
NPB　IS.W	0.632	1.194	1.372	1.594	1.817
NPB　IS.A	0.817	1.744	2.427	3.280	4.134
NPB　IS.B	0.908	2.037	2.989	4.178	5.368
NPB　IS.C	0.837	1.900	2.727	3.760	4.793
NPB　LU.S	0.663	1.021	1.040	1.064	1.088
NPB　LU.W	0.644	1.325	1.624	1.997	2.371
NPB　LU.A	0.760	1.874	2.676	3.679	4.683
NPB　LU.B	0.464	2.226	3.351	4.758	6.165
NPB　LU.C	0.433	2.731	4.320	6.307	8.293
NPB　MG.S	0.654	1.007	1.014	1.022	1.031
NPB　MG.W	0.539	1.593	2.138	2.819	3.501
NPB　MG.A	0.388	2.635	4.135	6.012	7.888
NPB　MG.B	0.397	3.334	5.477	8.157	10.836
NPB　MG.C	0.472	4.327	7.383	11.202	15.021
NPB　SP.S	0.647	1.035	1.068	1.108	1.148
NPB　SP.W	0.650	1.762	2.463	3.338	4.213
NPB　SP.A	1.010	7.619	13.697	21.294	28.891
NPB　SP.B	0.893	7.919	14.273	22.215	30.157
NPB　SP.C	0.805	8.031	14.488	22.559	30.629
NPB　UA.S	0.662	1.200	1.384	1.613	1.843
NPB　UA.W	0.665	1.748	2.436	3.295	4.154
NPB　UA.A	0.815	2.683	4.229	6.161	8.093
NPB　UA.B	0.956	3.457	5.713	8.533	11.353
NPB　UA.C	1.000	3.898	6.560	9.887	13.214
GAP bc –g 17	0.668	1.539	2.033	2.651	3.269
GAP bc -g 18	0.658	1.899	2.724	3.756	4.787
GAP bc -g 19	0.626	2.303	3.499	4.994	6.490
GAP bc -g 20	0.591	2.366	3.620	5.188	6.757
GAP bc -g 21	0.534	2.399	3.684	5.291	6.897
GAP bc -g 22	0.448	2.214	3.328	4.721	6.115
GAP bfs -g 17	0.695	1.611	2.173	2.875	3.577
GAP bfs -g 18	0.747	2.150	3.206	4.526	5.846
GAP bfs -g 19	0.618	1.450	1.863	2.380	2.897
GAP bfs -g 20	0.533	1.441	1.846	2.351	2.857
GAP bfs -g 21	0.478	1.418	1.803	2.283	2.763
GAP bfs -g 22	0.431	1.367	1.704	2.125	2.547
GAP cc -g 17	0.668	1.581	2.114	2.780	3.447
GAP cc -g 18	0.652	1.968	2.856	3.967	5.078
GAP cc -g 19	0.644	2.545	3.963	5.737	7.510
GAP cc -g 20	0.592	2.901	4.647	6.829	9.011
GAP cc -g 21	0.567	3.103	5.034	7.447	9.861
GAP cc -g 22	0.549	3.229	5.277	7.836	10.395
GAP pr -g 17	0.667	1.526	2.009	2.613	3.217
GAP pr -g 18	0.560	1.711	2.363	3.179	3.995
GAP pr -g 19	0.560	3.728	6.234	9.366	12.498
GAP pr -g 20	0.493	4.320	7.368	11.178	14.988
GAP pr -g 21	0.487	4.021	6.796	10.264	13.732
GAP pr -g 22	0.483	4.648	7.998	12.185	16.373
GAP sssp -g 17	0.651	1.517	1.992	2.585	3.178
GAP sssp -g 18	0.627	1.826	2.585	3.534	4.482
GAP sssp -g 19	0.583	2.143	3.193	4.505	5.817
GAP sssp -g 20	0.553	2.025	2.965	4.142	5.318
GAP sssp -g 21	0.528	2.698	4.257	6.206	8.154
GAP sssp -g 22	0.509	2.308	3.509	5.010	6.511

Slope はスレッド数に対しては一定値であることは前述の通りだが, アプリケーションやそのサイズごとに異なる. 遅延に対する感度(性能低下率の大小)を必ずしも表していないことが全体的に見て取れる. Slope はメモリシステムとアプリケーションのアクセスパターンの相性に相当するものと考えられ, 大きい値ほど CPU のストールを引き起こしやすいことを意味する. ただし Slope が大きくてもメモリアクセス数が少ないと性能低下率は高くならない.

9. まとめ

本報告では Intel Xeon Phi KNL 上で実行可能で，かつメモリレベル並列性や Out-of-Order 機構の効果を反映可能な主記憶遅延感度評価法を提案した. 提案手法により Stall カウンタを持たない上にシミュレーション時間が膨大になりがちなメニーコア上での大容量 DRAM キャッシュ設定時におけるメモリ遅延による性能低下率を高スループット（基本的に実時間）で予測することが可能である. 1 回の測定で任意の種類の遅延に対する性能低下率を得られるため，実質的には実時間以上の評価スループットを有すると言える.

本手法で 1 スレッド実行時の 208 行列，8 格納方式に対する疎行列ベクトル積の性能評価を実行するのに KNL だと 21 日かかった. BSC と Samsung のチームが 2016 年発表した方法でこの例と類似した結果を得る場合は TaskSim の OoO パイプラインを模擬しない MEM モードの実行時間だけで 1 遅延につき 30 年，4 遅延だと 120 年かかると予想される. OoO パイプラインの効果を反映している本測定と同等の結果を得るためには TaskSim なら Instr モード(実機の 31,173 倍遅い)等で実行する必要があり，さらに約 20 倍の時間(1 遅延あたり 1,800 年)がかかってしまう.

また，本手法は高スループットなだけでなく，メニーコア時に必須となるマルチスレッド実行に対する対応力を有しており，16 スレッド時の誤差についても評価結果を示した.

今後は提案手法をより多くの種類の大きなフットプリントを有するアプリケーションの評価を行なう予定である.

謝辞　本研究は，科学技術振興機構戦略的創造研究推進事業(JST CREST)の研究課題「ポストペタスケール時代のメモリ階層の深化に対応するソフトウェア技術」の支援を受けている. 最新の Skylake-SP の環境は Visual Technology 社のテストドライブセンターのリモートアクセスを使わせていただいた. この環境については特に同社の鬼澤氏の多大なご協力の下，本実験は遂行された. 九州大学の南里准教授にはその代替機として同大学の ITO システムの利用に関し，ご尽力いただいた. ご両名ほかその関係者に深く感謝申し上げます.

参考文献

[1] Intel. 3D Xpoint Technology. http://www.intelsalestraining.com/infographics/memory/3DXPoint c.pdf, (参照 2016-11-20).
[2] Intel. IDF15 におけるスライドの抜粋. http://www.3dnews.ru/919364, (参照 2016-11-20).
[3] 田邊昇，遠藤敏夫. 中遅延大容量メモリ階層出現のインパクトと新たな対応に関する初期検討. 第 157 回ハイパフォーマンスコンピューティング研究会，2016, Vol.2016-HPC-157,No.11.
[4] Avadh Patel, Furat Afram, Shunfei Chen, and Kanad Ghose. MARSS: a full system simulator for multicore x86 CPUs. In Proceedings of the 48th Design Automation Conference (DAC '11). ACM Press, 2011, p.1050-1055.
[5] MARSSx86 - Micro-ARchitectural and System Simulator for x86-based Systems. http://marss86.org/~marss86/index.php/Home, (参照 2016-02-20).
[6] Haris Volos, Guilherme Magalhaes, Cherkasova, and Jun Li. Quartz: A Lightweight Performance Emulator for Persistent Memory Software. In Proceedings of the 16th Annual Middleware Conference (Middleware '15), ACM Press, 2015, p.37-49.
[7] 田邊昇，遠藤敏夫. 疎行列系アプリケーション性能の主記憶遅延増加の影響評価. 第 158 回ハイパフォーマンスコンピューティング研究会，2017, Vol.2017-HPC-158,No.15.
[8] Kazi Asifuzzaman, Milan Pavlovic, Milan Radulovic, David Zaragoza, Ohseong Kwon, Kyung-Chang Ryoo, and Petar Radojkovi?. Performance Impact of a Slower Main Memory: A case study of STT-MRAM in HPC. In Proceedings of the Second International Symposium on Memory Systems (MEMSYS '16), 2016, p.40-49.
[9] Alejandro Rico, Felipe Cabarcas, Carlos Villavieja, Milan Pavlovic, Augusto Vega, Yoav Etsion, Alex Ramirez, and Mateo Valero. On the simulation of large-scale architectures using multiple application abstraction levels. 2015. ACM Trans. Archit. Code Optim. 8, 4, Article 36.
[10] Jasmina Malicevic, Subramanya Dulloor, Narayanan Sundaram, Nadathur Satish, Jeff Jackson, and Willy Zwaenepoel. Exploiting NVM in large-scale graph analytics. In Proceedings of the 3rd Workshop on Interactions of NVM/FLASH with Operating Systems and Workloads (INFLOW '15). ACM Press, 2015, Article 2.
[11] Joy Arulraj, Andrew Pavlo, and Subramanya R. Dulloor. Let's Talk About Storage & Recovery Methods for Non-Volatile Memory Database Systems. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15). ACM Press, 2015, p.707-722.
[12] Subramanya R. Dulloor, Sanjay Kumar, Anil Keshavamurthy, Philip Lantz, Dheeraj Reddy, Rajesh Sankaran, and Jeff Jackson. 2014. System software for persistent memory. In Proceedings of the Ninth European Conference on Computer Systems (EuroSys '14). ACM Press, 2015, Article 15.
[13] Ismail Oukid, Daniel Booss, Wolfgang Lehner, Peter Bumbulis, and Thomas Willhalm. SOFORT: a hybrid SCM-DRAM storage engine for fast data recovery. In Proceedings of the Tenth International Workshop on Data Management on New Hardware (DaMoN '14), ACM Press, 2015, Article 8.
[14] Yiying Zhang, Jian Yang, Amirsaman Memaripour, and Steven Swanson. Mojim: A Reliable and Highly-Available Non-Volatile Memory System. SIGARCH Comput. Archit. News 43, 1 (March 2015), ACM Press, p.3-18.
[15] Intel(R) Xeon Phi(TM) Processor Performance Monitoring Reference Manual Volume 2: Events
[16] 反復解法ライブラリ LIS. http://www.ssisc.org/lis/
[17] Tim Davis. The SuiteSparse Matrix Collection. http://www.cise.ufl.edu/research/sparse/matrices/