Vol.2017-HPC-158 No.22
2017/3/9

Knights Landing における電力制約下での性能挙動の分析

小野美由紀[†1]　福本尚人[†1]　中島耕太[†1]

概要：HPC システムやデータセンターの大規模化と高性能化が進んでおり，これに伴う消費電力の増大が課題となっている．Linpack のような消費電力が特別に大きいプログラムを実行すると，HPC システムに供給可能な消費電力を超えてしまうことがある．HPC システムにおいて越えてはならない最大の消費電力（消費電力キャップ）を設定し，この値を超えないように制御することが行われる．HPC システムの消費電力キャップは，気温や供給可能な消費電力に応じて決定される．そのため，同一サーバであっても異なる消費電力キャップが設定されることが考えられる．そこで，本稿では挙動の分析が容易なプログラムとして行列積を選び，異なる消費電力キャップ間における挙動の違いを分析する．特に消費電力による性能差の影響を受けやすい Intel 社の Xeon Phi (Knights Landing) を用いる．分析した結果，異なる消費電力キャップを設定した電力制約下で同じプログラムを実行しても性能が変化しない場合や，異なる実行パターンでは消費電力キャップによって性能の優劣が逆転する場合があることが明らかになった．

1. はじめに

HPC システムやデータセンターの大規模化と高性能化が進んでおり，これに伴う消費電力の増大が課題となっている．この課題を解決するために，電力あたりの性能を高めることが重要である．そのため，限られた消費電力バジェットを効率よく使うための研究が行われている [1]．

また，Linpack [2] のような消費電力が特別に大きいプログラムを実行すると，HPC システムに供給可能な消費電力を超えてしまうことがある．そのため，HPC システムにおいて越えてはならない最大の消費電力（以降，消費電力キャップと呼ぶ）を設定し，この値を超えないように制御することが多い[3]．すべての機器を動作させると供給可能な電力量を超えるように，HPC システムは組まれることがある．例えば，サーバの冷却に必要な消費電力が低い冬に合わせてサーバの台数などを決定する場合，冬と同様に HPC システムを動作させると，供給可能な電力量を超えてしまうことがある．契約した消費電力を超えることを避けるため，消費電力キャップを超えないように制御を行う．これを実現する有名な方法の一つとして，RAPL を用いた動作周波数制御がある[4]．RAPL を用いて CPU や DRAM に対して消費電力キャップを設定し，この値を超えないように動作周波数を制御する．今後は，こういったサーバ単位で消費電力キャップを設定する HPC システムが多く登場すると予想される．

HPC システムの消費電力キャップは，気温や供給可能な消費電力に応じて決定される．そのため，同一サーバであっても異なる消費電力キャップが設定されることが考えられる．ある消費電力キャップで有効だった性能向上手法がより厳しい消費電力キャップでは効果がなくなるということも考えられる．異なる消費電力キャップにおけるアプリケーションの性能の変化はまだよく知られていない．

†1 (株)富士通研究所
Fujitsu Laboratories Ltd.

そこで，本稿では挙動の分析が容易なプログラムとして行列積を選び，異なる消費電力キャップ間における性能，動作周波数，消費電力の分析を行う．消費電力による性能差の影響を受けやすい Intel 社の Xeon Phi (Knights Landing : KNL) を用いる．分析した結果，異なる消費電力キャップを設定した電力制約下で同じプログラムを実行しても性能が変化しない場合や，異なる実行パターンでは消費電力キャップによって性能の優劣が逆転する場合があることが明らかになった．

2 章では KNL における電力制約について述べる．3 章では評価に使用するプログラムについて説明する．次に，4 章では性能評価と分析を行い，最後にまとめる．

2. KNL における電力制約

2.1 電力制約下での性能特性

一般的に，アプリケーションの性能チューニングを行うことにより，IPC (Instructions per Cycles) が向上する．これにより，消費電力が増加し，温度も上昇する．アプリケーションの動作環境において，温度の制限や電力の制限が課せられている場合がある．そのような環境において，課せられた制限を超えると，強制的に動作周波数が下げられる．動作周波数の低下により，性能が低下する場合もある．例えば，性能向上はするが，電力性能が悪くなるような性能チューニングをする場合，アプリケーションの性能チューニングをしても，効果が得られず，逆に性能が低下することになる．

温度や電力に関する制約は，環境依存であり，状況に応じて異なる制約が課せられる場合がある．電力制約によって性能がどう変化するかを把握していないと，効果的なチューニングが行えない．

例えば，Intel Xeon Haswell 上で，コードレベルの性能最適化が電力効率に与える影響を分析した結果が報告されている[5]．報告では，電力制約を想定し，CPU の動作周波数

を変化させてチューニング手法を評価している.その結果,評価プログラムによっては,チューニングの種類により電力効率の向上比率が大きく異なることが示されている.

KNL はメニーコア型プロセッサであり,1 つのコアの動作周波数は低い.動作周波数の変動幅が小さく,電力制約による影響を受けやすいと考えられる.しかし,KNL における電力制約下での性能に関する報告はほとんど行われていない.そこで,NKL における性能と電力制約について基礎調査を行う.

2.2 RAPL による電力制約

今回,電力制約は Sandy Bridge 以降の Intel CPU に搭載されている機能 RAPL (Running Average Power Limit) を使用する.KNL でも本機能は提供されている.

RAPL は,ソフトウェアから消費電力を監視制御機能するための機能であり,本機能により,CPU やメモリの消費電力を監視し,予め設定した電力に達した場合に消費電力を制限することができる.

電力制御として,CPU 全体と DRAM に対して 2 段階の制約を設定することができる.電力制約は,時間区間とその時間内で使用可能な平均消費電力(W)を指定する.消費電力としては最低 98W から最大 258W までの値を指定できる.例えば,1 秒間に 215W という第 1 の制約と,10 ミリ秒間に 258W という第 2 の制約を設定する,この場合,1 秒間に使用できるのは 215W だが,10 ミリ秒という短時間には 258W まで使用可能となる.

RAPL で得られる消費電力は計算値である[6]が,RAPL と外部電力計の消費電力傾向は非常に良く一致するという報告がある[7].これは,電力特性を把握するために,電力制約を設定する,あるいは,消費電力の傾向を捉える用途には十分な精度と考える.また,専用ハードウェアなしに,CPU に組み込まれた機能を利用して,電力情報を採取可能であるという利点もある.

3. 評価プログラム

電力特性の変化を評価するプログラムとして Linpack ベンチマーク[2]の行列積部分を抜き出したものを採用した.

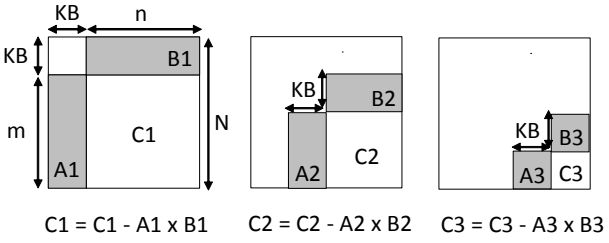
本プログラムを採用した理由は以下の通りである.

- 演算ネックのプログラムであり処理内容が単純なため,電力制約を変化させた場合の性能変化要因の推測が比較的容易であること.
- 本プログラムを高速化することは Linpack ベンチマークの高速化につながるため,有益であること.

3.1 Linpack の行列積

Linpack の行列積の中で最も処理量が大きいのは,行列の更新処理である.そこで,更新処理中の行列積を評価プログラムとして採用した.更新処理の行列積は,図 3.1 のように一辺 N の行列の一部を用いて行われる.まず,行列 A1 と行列 B1 の積と行列 C1 の差を取る.次に,KB 小さい,行列 A2, B2, C2 を用いて同様の計算を行う.このような行列演算を残りの行列がなくなるまで繰り返す.図 3.1 の場合,行列 A3, B3, C3 が最後の行列積となる.

行列 A1, B1, C1 に対して,行列積を行うときは図 3.2 のように小さいタイルサイズで区切って行列演算を行う.本来の Linpack の処理では行列積実行と同時に通信を行う.そのため,ある程度の頻度で通信の進捗を進める処理を行いたい.これを実現するために,行列をタイル状に分割し,そのタイルごとに行列積を行い,一つの行列積が終わるごとに進捗処理を進める.タイルのサイズは行列積ライブラリの特性やサーバ間の通信速度を考慮した上で適切に設定しなければならない.本評価プログラムではこのような理由により,タイル状に区切って行列演算を行う.




```
m = N;  ldc = N;
C = malloc ( N*N*sizeof(double));
while ( m > KB ) {
  m = m – KB;   n = m; k = KB;
  A = C + KB; B = C + KB * ldc; C = C + KB (1 + ldc) ;
  DGEMM ( m, n, k, -1, A, ldc, B, ldc,  1, C, ldc ) ;
}
```

図 3-1　Linpack における行列積計算

3.2 行列積関数の呼び出し部分

評価プログラムでは,図 3.2 のように行列をタイル状に区切って行列演算を行う.この行列演算部分のコードを 2 種類準備した.一つ目は従来通りの方法で,図 3.3 のように行列積を行う関数である cblas_dgemm を 2 重ループ内で呼び出す.

二つ目は,図 3.4 のように,cblas_dgemm_pack という関数を使用する. この関数は Intel compiler2017 で活用できる.この関数では,cblas_dgemm 内部で行われている行列を並べ替えてバッファにコピーする処理を行う.cblas_dgemm_compute は,バッファがコピーされていることを前提に行列積を行う.今回の評価プログラムのような行列を再利用する場合(図 3.3 では行列 A1)は,cblas_dgemm_pack を活用することで,バッファにコピーする処理回数を減らすことができる.これにより,性能向上や消費エネルギーの削減を達成できると予想している.

以降では，一つ目のプログラムをタイル型，二つ目のプログラムをタイル型 Pack と呼ぶ.

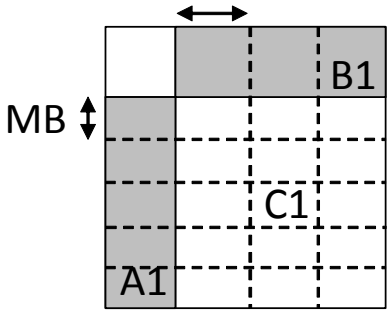


図 3-2　タイル状に区切った行列

```
DGEMM (int m, int n, int, k, int alpha, double *A, int
lda, double *B, int ldb, int beta, double* C, int ldc ) {
 for ( mi = 0; mi < m; mi += MB ) {
  for ( ni = 0; ni < n; ni += NB ) {
      cblas_dgemm ( CblasColMajor, CblasNoTrans,
CblasNotrans, MB, NB, KB, alpha, (A + mi), lda,
(B+ldb*ni), ldb, beta, (C+mi+ldc*ni), ldc );
   }
 }
```

図 3-3　行列積呼び出し部分のコード（通常の DGEMM）

```
DGEMM (int m, int n, int, k, int alpha, double *A, int
lda, double *B, int ldb, int beta, double* C, int ldc ) {
 for ( mi = 0; mi < m; mi += MB ) {
  cblas_dgemm_pack (CblasColMajor, CblasAMatrix,
                    CblasNoTrans, MB, NB, KB,
                    alpha, (A+mi), lda, workA);
  for ( ni = 0; ni < n; ni += NB ) {
      cblas_dgemm_compute ( CblasColMajor,
CblasPacked, CblasNotrans, MB, NB, KB, alpha,
workA, lda, (B+ldb*ni), ldb, beta, (C+mi+ldc*ni), ldc );
   }
 }
```

図 3-4　行列積呼び出し部分のコード（Packed API の利用）

4. 測定と結果の分析

4.1 電力制約下での性能測定

3 章で述べた 2 つの評価プログラムに対して，行列サイズやタイルサイズを変えて実行し，性能を比較する．評価には，富士通製 KNL サーバである Fujitsu Primergy CX1640 M1 を用いた．TDP は 215W である．測定では，評価プログラムの性能だけでなく同時に動作周波数，温度，消費電力を計測する．

動作周波数と温度は MSR (Model specific registers) から取得した値を元に算出する．

消費電力量は RAPL の値を参照する．RAPL では CPU 全体や CPU 内のコア全体（Core）の消費電力を監視できる．さらに，サーバ機ではメモリ（DRAM）が監視対象となる．これらの監視単位毎に，電力制御や消費電力記録などのためのレジスタが用意されており，消費電力値（Joule）は約 1 ミリ秒毎に更新される．

評価プログラムは CPU に負荷をかけるものであり，メモリはボトルネックではない．そのため，CPU の消費電力に注目し，RAPL を使用して CPU に電力制約を設定する．電力制約を表 4-1 に示す．制約 1 は 1 秒間に 215W という第 1 の制約と，約 10 ミリ秒間に 258W という第 2 の制約を設定する．なお，258W は RAPL で設定可能な最大電力である．制約 1 が最も弱く，制約 3 が最も強い制約となっている．3 段階の電力制約の下で実行し，電力制約による性能への影響を調査する．

表 4-1　電力制約

	第 1 制約		第 2 制約	
	時間 1	電力 1	時間 2	電力 2
制約 1	1 秒	215W	10 ミリ秒	258W
制約 2	1 秒	200W	10 ミリ秒	215W
制約 3	1 秒	185W	10 ミリ秒	215W

なお，同じプログラムを同じサイズ，同じ電力制約下で実行した場合でも性能にばらつきが発生するため，1 つのパターンを 15 回実行し，中央値を各パターンの性能値とした．

評価プログラムでは，行列サイズはある程度大きいほうが性能がよい傾向にある．そこで，タイル型とタイル型 Pack の 2 つの評価プログラムを行列サイズ 30,000, 60,000, 90,000 に対して 3 段階の電力制約で実行した．なお，タイルサイズは m 方向，n 方向ともに行列サイズと同じサイズとした．実行した結果を図 4-1 にまとめる．グラフの縦軸は性能値，横軸は電力制約，折れ線は電力制約下で各評価プログラムを各行列サイズで実行した際に得られた性能値を表す．性能値は行列積のピーク性能に対する相対性能を示している．同一評価プログラムでは，行列サイズが大きいほど性能はよいが，行列サイズ 60,000 と 90,000 ではあまり性能差がないことがわかった．

今回は行列サイズ 30,000 と 60,000 について，タイルサイズを変えて試すこととした．タイルサイズを表 4-2 にまとめる．タイルの m 方向のサイズは行列サイズ N,N/2 の 2 通りであり，n 方向は 4 通りとした．従って，行列サイズ 2 通り，タイルの m 方向サイズ 2 通り，n 方向サイズ 4 通りの 16 パターンを 2 つの評価プログラムで実行した．

また，評価プログラム実行時に，途中経過として，経過時間，コアの平均動作周波数，コアの平均温度，CPU の消

費電力量, DRAM の消費電力量を出力し, 分析に使用する.

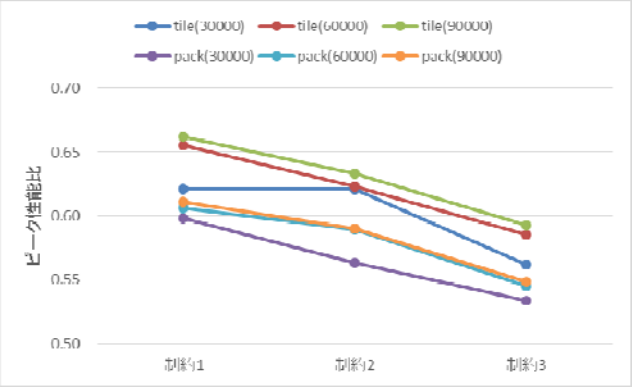


図 4-1　性能と制約

表 4-2　タイルサイズ

行列サイズ	m方向	n 方向
30000	30000	30000,24000,12000,6000
	15000	30000,24000,12000,6000
60000	60000	60000,30000,12000,3000
	30000	60000,30000,12000,3000

4.2 性能評価

結果の全体的な傾向としては，行列サイズ 30000，60000 ともに, タイル型プログラムのほうがタイル型 Pack プログラムよりも性能がよい傾向にあった．タイルサイズに関しては，2 つの行列サイズともに，タイル型プログラムでは m 方向のサイズによる違いはあまりなかった．

行列サイズ 60000 で採取した 16 パターンを実行した結果を電力制約 1 の性能値で降順ソートしたグラフを図 4-2 と図 4-3 に示す．図 4-2 から図 4-5 までのグラフの縦軸は性能値，横軸は電力制約，棒グラフは電力制約下で各評価プログラムを各タイルサイズで実行した際に得られた性能値を表す．凡例の”m=”はタイルの m 方向サイズ,”n=”はタイルの n 方向サイズを表している．図 4-2 はタイル型プログラムの結果であり，n 方向サイズが大きいほど性能がよい．実行パターンによる性能差はあまり大きくない．

図 4-3 はタイル型 Pack プログラムの結果であり, m 方向および n 方向のサイズによる性能の明確な傾向は見られない．2 つの行列サイズとも，電力制約の強さに応じて，性能が劣化していることがわかる．

行列サイズ 30000 で採取した 16 パターンの結果を電力制約 1 の性能値で降順ソートしたグラフが図 4-4 と図 4-5 である．タイル型プログラムの結果をグラフ化した図 4-4 から, 電力制約 1 ではほとんど性能差がないことがわかる．電力制約 2 ではほぼ 2 つに性能が分かれている．制約 1 でソートした順とは性能が逆転しているパターンが存在するが，制約 1 の性能差がわずかであることから，問題でないと考えられる．制約 3 においても制約 1 でソートした順と性能が逆転しているパターンは存在するが，制約 2 で性能が悪かったパターン群からさらに制約 3 で性能が悪いものが出ているとみられる．

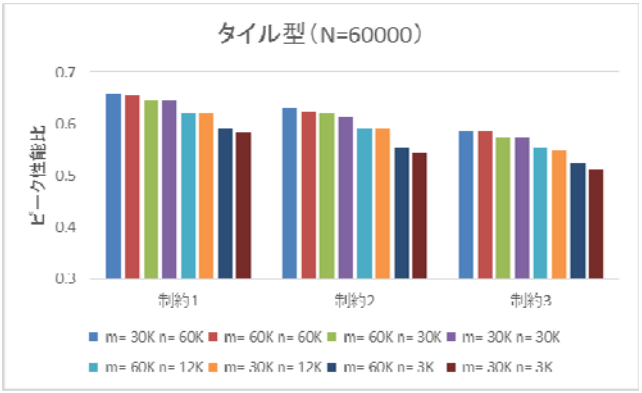


図 4-2　タイル型（行列サイズ 60000）の性能値

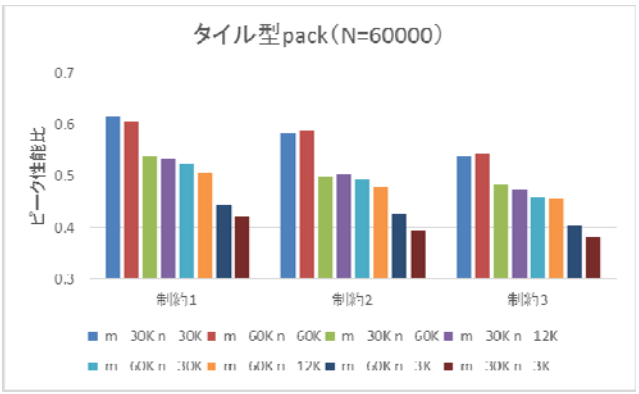


図 4-3　タイル型 Pack（行列サイズ 60000）の性能値

一方，タイル型 Pack プログラムの結果は図 4-5 に示すように，制約 2 と制約 3 において，制約 1 の性能値でソートした順と性能が逆転しているパターン”m 15000 n=30000”が存在する．つまり，電力制約の強さに応じて性能が劣化していないものが存在することがわかる．

図 4-7 と図 4-8 は, 図 4-4 と図 4-5 と同じ結果を実行パターン毎にまとめたものである．縦軸は性能値，横軸は実行パターン毎の各電力制約での性能値である．図 4-7 のタイル型プログラムでは，実行パターン”m=15000 n=30000”，”m=24000 n=30000”,” m=30000 n=30000”の制約 1 と制約 2,” m=30000 n=6000”，” m=15000 n=24000”，” m=30000 n=12000”の制約 2 と制約 3 の性能値がほぼ等しいことがわかる．一方，図 4-8 のタイル型 Pack プログラムでは，電力制約の強さに応じて性能が低下している．

このように，電力制約により，性能の逆転が発生する場合がある．このような状態が発生すると，電力制約を考慮した性能チューニングが必要となると言える．

電力制約に応じた性能低下が発生しないパターンが存

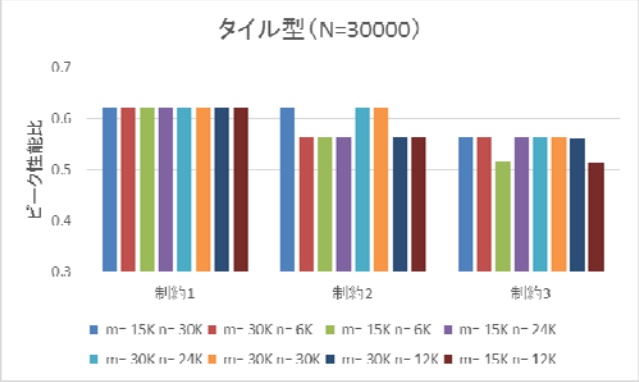


図 4-4　タイル型（行列サイズ 30000）の性能値

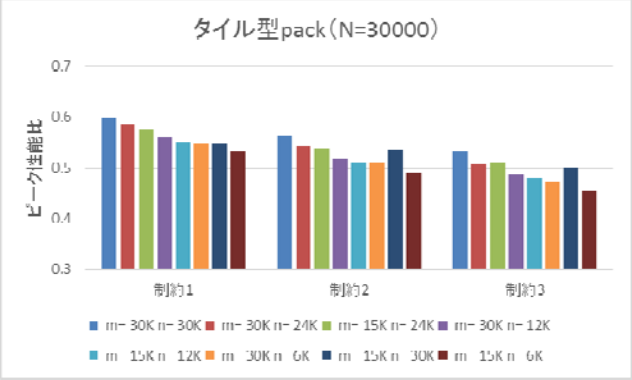


図 4-5　タイル型 Pack（行列サイズ 30000）の性能値

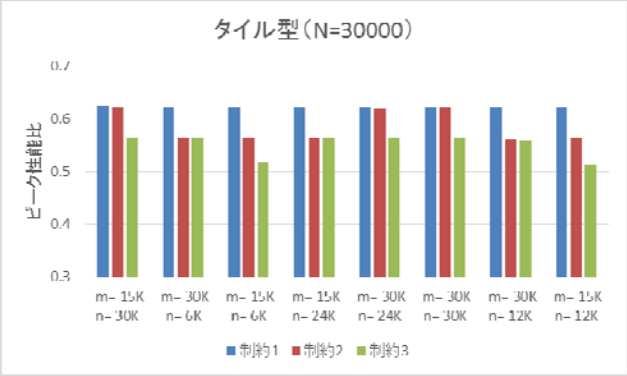


図 4-7　タイル型の実行パターン毎の性能値

図 4-8　タイル型 pack の実行パターン毎の性能値

在する，行列サイズ 30000 について，さらに分析を進める．

4.3　電力制約による性能逆転が発生する事例

異なるパターンで電力制約によって性能が逆転する事例について，その原因の分析を試みる．表 4-3 は性能を逆転された実行パターンと性能を逆転した実行パターンについて，各電力制約下での性能データをまとめたものである．性能データは性能値（ピーク性能比），平均動作周波数（GHz），平均温度（℃），CPU 消費電力（W），DRAM 消費電力（W），実行時間（秒）の 6 項目である．平均動作周波数と平均温度は実行時に出力された値の平均値，消費電力は総消費電力量を実行時間で割った値である．p は実行パターンを表し，p1 は性能を逆転したパターン”pack m=15000 n=30000”，p2 は性能を逆転されたパターン”pack m=30000 n =6000”である．L は電力制約を表し，L1 は電力制約 1，L2 は電力制約 2，L3 は電力制約 3 である．

性能で降順ソートした電力制約 L1 では，性能は僅差であり，平均温度は 0.4 度差があるが，その他の項目はほとんど差がない．性能の逆転が発生した電力制約 L2 と L3 では，平均動作周波数や平均温度，CPU 消費電力は 2 つのパターンでほとんど差がない．

p1 と p2 は電力制約 L1 の元では若干 p1 の方が高性能である．一方電力制約 L2 と L3 を加えると p2 の方が高性能になる．p1 について電力制約 L1，L2，L3 における性能及び平均周波数を見ると，電力制約が増すにつれて平均周波数が低下し，それにしたがって性能が低下している．一方で p2 については，電力制約 L1 と L2 の間では平均動作周波数は低下しているものの性能はあまり低下していない．すなわち p2 は電力制約 L1 と L2 の間では，周波数の低下に対して性能低下が低い特性があるといえる．

表 4-3　性能データの比較

	性能	周波数 (GHz)	温度 (℃)	CPU (W)	DRAM (W)	時間 (秒)
p1-L1	0.549	1.44	70.18	224	59	10.59
p2-L1	0.547	1.41	69.78	225	59	10.61
p1-L2	0.509	1.35	69.30	208	59	11.42
p2-L2	0.535	1.34	69.56	209	58	10.86
p1-L3	0.473	1.25	68.50	193	55	12.91
p2-L3	0.502	1.23	68.24	194	59	11.58

4.4　電力制約による性能低下が発生しない事例

同一パターンで異なる電力制約下で性能が低下しない事例について，その原因の分析を試みる．表 4-4 は電力制約による性能低下が発生しない 3 つのパターンについて，

各電力制約下で性能データをまとめたものである．性能データは性能値（ピーク性能比），平均動作周波数（GHz），平均温度(℃)，CPU 消費電力(W)，DRAM 消費電力(W)，実行時間（秒）の 6 項目である．平均動作周波数と平均温度は実行時に出力された値の平均値，消費電力は総消費電力量を実行時間で割った値である．p は実行パターンを表し，p3 は”tile m=15000 n=30000”，p4 は”tile m=30000 n=30000”，p5 は”tile m=30000 n=12000”を表す．L は電力制約を表し，L1 は電力制約 1，L2 は電力制約 2，L3 は電力制約 3 である．

CPU 消費電力は電力制約に応じた値となっており，平均温度はいずれもあまり差がない．

p3，p4，p5 のいずれにおいても，電力制約が増すと平均動作周波数は低下している．一方で平均動作周波数が低下しても性能はほとんど低下していない．このことから，p3，p4，p5 は p2 と同様に動作周波数の低下に対して性能低下が低い特性があるといえる．

表 4-4　性能データの比較

	性能	周波数 (GHz)	温度 (℃)	CPU (W)	DRAM (W)	時 間 (秒)
p3-L1	0.622	1.43	70.24	222	62	9.34
p3-L2	0.622	1.33	70.26	210	60	9.35
p4-L1	0.621	1.42	70.74	221	61	9.36
p4-L2	0.621	1.33	69.67	210	60	9.35
p5-L2	0.561	1.35	69.82	205	58	10.34
p5-L3	0.560	1.25	69.30	194	59	10.38

5. おわりに

本稿では挙動の分析が容易なプログラムとして行列積を選び，異なる消費電力キャップ間における挙動の違いを分析した．特に消費電力による性能差の影響を受けやすい Intel 社の Xeon Phi (Knights Landing)を用いた．分析した結果，異なる消費電力キャップを設定した電力制約下で同じプログラムを実行しても性能が変化しない場合や，異なる実行パターンでは消費電力キャップによって性能の優劣が逆転する場合があることが明らかになった．これらの現象が発生する要因の究明には至っていないが，行列積計算という動作周波数への依存が高いアプリケーションにおいて，動作周波数低下するにも関わらず性能低下しないケースがあることが分かった．

さらに，これらの現象が発生する要因を究明するために，CPU に関する性能データを採取するために使われる PMC (Performance Monitoring Counter) による分析を試みることも考えられる．

参考文献

[1] Cochran, Ryan, et al. "Pack & Cap: adaptive DVFS and thread packing under power caps." Proceedings of the 44th annual IEEE/ACM international symposium on microarchitecture. ACM, 2011.
[2] A. Petitet, R. C. Whaley, J. Dongarra, and A. Cleary. HPL - a portable implementation of the high-performance linpack benchmark for distributed-memory computers. http://www.netlib.org/benchmark/hpl, September 2008.
[3] 黄巍 他：エネルギー効率を考慮した電力制約下でのスループット指向ジョブスケジューリング，HPCS2015 (2015)
[4] Intel: CHAPTER 14.9 PLATFORM SPECIFIC POWER MANAGEMENT SUPPORT, Intel Software Developer's Manual, Volume 3B, (April 2016)
[5] 今村智史 他：コードレベル性能最適化が電力効率に与える影響の分析，HPC-155 No.21 (2016)
[6] Srinivas Pandruvada : Running Average Power Limit, https://01.org/blogs/tlcounts/2014/running-average-power-limit-%E2%80%93-rapl
[7] カオタン 他 : RAPL インタフェースを用いた HPC システムの消費電力モデリングと電力評価，HPC-141 No.20 (2013)