Vol.2015-HPC-150 No.28
2015/8/5

ノード内同時実行ジョブにおけるパフォーマンスカウンタによるプロセス毎消費電力のモデル化

寺西 賢人[1,a)]　野村 哲弘[1]　遠藤 敏夫[1]　松岡 聡[1]

概要：近年のスーパーコンピュータは大量に電力を消費するようになり，実用的なスーパーコンピュータの性能の向上には電力効率が課題となっている．消費電力の効率の良い制御のためにはより詳しい消費電力の計測を行う必要がある．しかし現状ノード毎の消費電力を計測することは可能だが，プロセス毎の消費電力の計測をすることはできない．本論文ではプロセス毎に計測可能なパフォーマンスカウンタを用いて消費電力をモデリングし，同一ノード内で同時にジョブを実行した場合のプロセス毎の消費電力の推定を提案する．作成したモデル式を用いた電力の推定実験を 1 プロセス時と 2 プロセス同時実行時についてそれぞれ行い，1 プロセス時は最大で誤差 5.16%，2 プロセス時は計測した組み合わせのうちの 84.8%が誤差 4%以内となった．

1. はじめに

近年のスーパーコンピュータでは大量に電力を消費するようになり，性能の向上には消費電力がボトルネックとなっている．消費電力を気にしなければ規模を大きくした分の性能向上が見込めるが，実際には電力バジェットの上限を設定する必要があり性能を引き出すことができない．したがって，消費電力をバジェットの上限まで有効活用するようにジョブを実行できればより性能を上げることができるはずである．

消費電力に重点を置いたジョブのスケジューリングをするためには，各ジョブがどれぐらいの電力を消費するのかを把握する必要がある．

複数のジョブを同時実行した時の各ジョブの消費電力を調べるにはプロセス毎の消費電力を知ることが必要になるが，現状計測できるのはノード毎の消費電力のみである．一方，近年計算機の消費電力を温度やプロセッサのパフォーマンスカウンタなどの指標から推定することが可能である．本論文ではプロセス毎の寄与分を按分できるパフォーマンスカウンタに着目し，プロセス毎の消費電力の寄与分の推定手法について述べる．

[1] 東京工業大学
Tokyo Institute of Technology
[a)] teranishi.k.aa@m.titech.ac.jp

2. 消費電力の推定方法

2.1 消費電力の推定

本論文ではプロセス毎の消費電力の推定を目指す．現状プロセス毎に消費電力の計測をすることは出来ないが，パフォーマンスカウンタならばプロセス毎に計測することが可能である．そこで，パフォーマンスカウンタを用いて消費電力の値を表すモデルを作成すればプロセス毎の消費電力を推定することが可能だと思われる．このモデル作成は統計的手法を用いて行う．まず多様なアプリケーションを用意し一つずつ実行時のパフォーマンスカウンタと消費電力のデータを計測し，そのデータを分析して関係式を導く．作成したモデルの有効性を調べるために，分析に用いていないアプリケーションについてのデータを計測する．モデルから推定した消費電力と実際の消費電力を比較し，どれぐらい一致しているかを調べる．

2.2 パフォーマンスカウンタと消費電力の計測

消費電力の推定のために必要なパフォーマンスカウンタと消費エネルギーのデータを計測する．まず何かアプリケーションを一つ実行し，そのアプリケーションでのパフォーマンスカウンタの値を計測する．同時に，実行中に消費したエネルギーを計測する．消費エネルギーを観測できる範囲は複数あるが，今回は全体の合計を用いてモデルを作る．同様に様々な種類・サイズのアプリケーションについて計測し，分析用のデータとしてまとめる．同じよう

なアプリケーションのみを分析用のデータにすると消費電力の推定が可能なアプリケーションの種類が限られてしまうので，できるだけ多くの種類のものを用いる．

2.3 消費電力のモデル化

計測したデータを分析する．消費電力の積算値，つまりアプリケーション実行中に消費したエネルギーはパフォーマンスカウンタを用いた一次式で表すことが出来ると考えられる．その式の全体を実行時間で割ることで，アプリケーションの実行中の平均の消費電力とパフォーマンスカウンタの関係式になる．したがって，まずはパフォーマンスカウンタを用いたエネルギーのモデル化を行う．

CPU は計測用プログラム実行時以外にも常に電力を消費している．この電力を Idle 時電力と呼ぶことにする．計測したエネルギーの中には Idle 時電力によるものも含まれるが，Idle 時電力は測定したアプリケーションとは無関係に常に存在しているので，プロセス毎のパフォーマンスカウンタのみを用いた式では表すことができない．Idle 時電力はほぼ一定であると考えられるので，そのエネルギーは実行時間に比例することがわかる．従って，パフォーマンスカウンタの値と実行時間を用いて消費エネルギーを表すモデルを作成することを目指す．説明関数に用いるパフォーマンスカウンタが n 種類の場合は以下のような関係式を重回帰分析を用いて調べる．

$$E = a_0 T_{\text{exe}} + \sum_{i=1}^{n} a_i X_i \tag{1}$$

表 1 式中のパラメータの説明

E	消費エネルギー
a_i	推定係数
T_{exe}	実行時間
X_i	パフォーマンスカウンタの計測値

この式全体を実行時間 T_{exe} で割ることで実行時の平均の消費電力 P の式になり，パフォーマンスカウンタを用いた消費電力のモデル化ができる．

$$P = \frac{E}{T_{\text{exe}}} \tag{2}$$

$$P = \frac{a_0 T_{\text{exe}} + \sum_{i=1}^{n} a_i X_i}{T_{\text{exe}}} \tag{3}$$

3. 計測に用いるソフトウェア

3.1 RAPL

Intel の Sandy Bridge マイクロアーキテクチャ以後のプロセッサには，RAPL(Running Average Power Limit) と呼ばれる電力監視インターフェースが備えられている．RAPL インターフェースではパフォーマンスカウンタや温度などを用いてプロセッサの消費電力を範囲で分けて別々に計測・制御を行うことができる．今回の実験では CPU を 2 つ搭載したマシン上で行うので，CPU1,2 それぞれでチップ全体，コア部分，DRAM の消費電力の計測が可能である．RAPL を用いることで高精度の電力計測が可能であり，RAPL で取得した電力はノード全体の電力と高い相関関係がありモデリングができることなどが判明している [1],[2]．よって，RAPL により計測できる消費電力を扱った実験をすることでノード全体の消費電力についての研究にも繋がることがわかる．

3.2 PAPI

ハードウェアのパフォーマンスカウンタの値を調べる手法として PAPI(Performance Application Programming Interface) というライブラリが存在し，近年のメジャーなマイクロプロセッサ上なら使用することができる．PAPI では各イベント毎に対応するカウンタ名が設定されており，例えば PAPI_L1_TCM というカウンタは L1 キャッシュミスの回数を表している．調べられるカウンタの種類は環境によって変化し，papi_avail コマンドにより実験環境でのカウンタの情報が把握できる．また同時に調べられるカウンタの組み合わせにも制限があるが，これは papi_event_chooser コマンドによって調べられる．PAPI のみでパフォーマンスカウンタの計測することは可能だが，今回は複雑なアプリケーションでの計測を行うため Score-P というソフトウェアを利用する．

3.3 Score-P

Score-P とはアプリケーションのパフォーマンスの測定を行うソフトウェアである．プログラムをコンパイルする際に，コンパイラの前に scorep コマンドを追加してコンパイルするだけで自動で様々な計測を行う機能が埋め込まれる．scorep コマンドを使ってコンパイルされたものを実行すると，各関数毎の実行時間や実行回数などを計測し結果をファイルとして生成する．計測する値に関する設定は環境変数によって変更でき，SCOREP_METRIC_PAPI という環境変数に PAPI のカウンタ名を設定することでパフォーマンスカウンタの計測を実装できる．Score-P はコンパイル時に用いるだけで計測機能の実装が可能と便利なツールであり，自力でのプログラムの改造が難しいアプリケーションでの計測も行うため今回の実験で採用した．

4. 実装

4.1 消費電力の測定方法

消費電力の計測には RAPL インターフェースを用いる．RAPL では各 CPU のチップ全体，チップ上のコア部分，DRAM の消費電力を調べることができる．まず消費エネルギーを用いてモデリングをするので，あるアプリケーションの実行中の消費エネルギーを RAPL で計測するよ

うなプログラムを作る．今回は CPU を 2 つ搭載のマシンで計測するので**表 2** の 6 種類の消費エネルギーが計測できる．

表 2 RAPL を用いて計測できる消費エネルギー

cpu1pkgenergy	cpu1 のチップ全体のエネルギー
cpu1pp0energy	cpu1 のコア部分のエネルギー
cpu1dramenergy	cpu1 での DRAM のエネルギー
cpu2pkgenergy	cpu2 のチップ全体のエネルギー
cpu2pp0energy	cpu2 のコア部分のエネルギー
cpu2dramenergy	cpu2 での DRAM のエネルギー

計測プログラムを作成するために，まず RAPL で消費エネルギーを計測し続けるプログラムを作る．この消費エネルギー計測プログラム上で子プロセスを作り，親プロセスで調べた値を共有メモリを用いて各プロセスで共有できるようにする．共有メモリとは別々のプロセスで値を共有することができる変数のことである．子プロセスでは計測したいアプリケーションを実行し，その実行前後の消費エネルギーの差から実行中に消費したエネルギーの値を求める．また，消費エネルギーの計測と同時に実行時間も計測する．

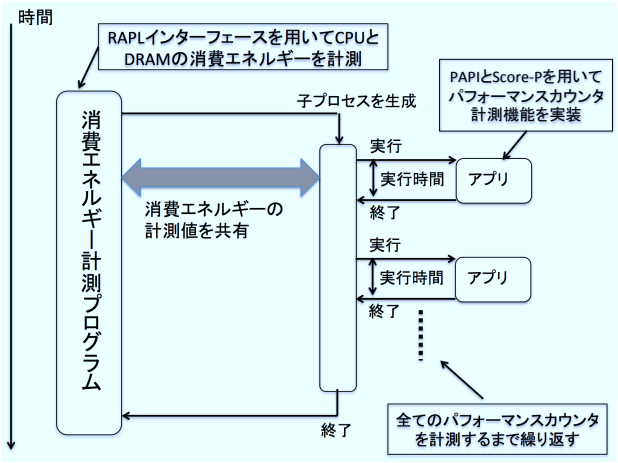

図 1 消費エネルギーとパフォーマンスカウンタの計測方法

4.2 プロセス毎パフォーマンスカウンタの計測方法

PAPI によって与えられるパフォーマンスカウンタを計測する．PAPI のみでもパフォーマンスカウンタの計測は可能だが，多種類のアプリケーションでの計測を重視し Score-P というツールを用いる．Score-P を使ってコンパイルしたアプリケーションを上記の子プロセスで実行することにより，アプリケーション実行中の消費エネルギーとパフォーマンスカウンタを同時に計測するようなプログラムを実現できる．計測するパフォーマンスカウンタの種類は環境変数によって設定できるが，1 回の実行で得られるパフォーマンスカウンタの数や組み合わせは限られる．よって，**図 1** のようにすべてのパフォーマンスカウンタの値を得るために環境変数を変えて繰り返し実行することになる．

4.3 計測結果の分析

測定したデータを統計ソフトの R を用いて重回帰分析し，次のようなモデル式を作る．

$$E = a_0 T_{\text{exe}} + \sum_{i=1}^{n} a_i X_i \tag{4}$$

$$P = \frac{a_0 T_{\text{exe}} + \sum_{i=1}^{n} a_i X_i}{T_{\text{exe}}} \tag{5}$$

表 3 式中のパラメータの説明

E	消費エネルギー
P	消費電力
a_i	推定係数
T_{exe}	実行時間
X_i	パフォーマンスカウンタの計測値

複数のパフォーマンスカウンタの計測のために何度もプログラムを実行しているので，それぞれの消費エネルギーの平均をこのプログラムの消費エネルギーの値とする．実行時間も同様に平均を用いる．RAPL により計測できる範囲全体の消費エネルギーは CPU1 のチップ全体と DRAM，CPU2 のチップ全体と DRAM の和であるので，

$$\begin{aligned} E \ = \ & \text{cpu1pkgenergy} + \text{cpu1dramenergy} \\ & + \ \text{cpu2pkgenergy} + \text{cpu2dramenergy} \end{aligned} \tag{6}$$

を目的変数とする．説明変数は計測したパフォーマンスカウンタの中の数種類と実行時間を用いる．また，定数項は 0 とする．以上の条件に従って重回帰分析を R で行う．分析で作成したモデルの両辺を実行時間で割ることで，プログラム実行中の平均消費電力についてのモデル式となる．

4.4 電力推定

新たなアプリケーションについてモデル作成時と同様にパフォーマンスカウンタ，実行時間，消費エネルギーの計測をする．次にパフォーマンスカウンタと実行時間を作成したモデルに当てはめて消費電力を推定する．計測した消費エネルギーから求めた実際の消費電力の値と，推定した消費電力の値を比較し精度を確認する．

4.5 複数プロセスでの電力推定

複数のアプリケーションを同時に実行してプロセス毎の消費電力を推定する．その後各プロセスで推定した消費電力から全体の消費電力を推定し，実測値と比較し有効性を調べる．まず 2 プロセスで同時にアプリケーションを実行してそれぞれのパフォーマンスカウンタ，実行時間，消費

エネルギーを計測する．計測するアプリケーションをアプリケーション A，アプリケーション B として，以下の図のようにデータの計測中は常にもう片方のプロセス上でもアプリケーションが実行されているような仕組みで計測プログラムを作る．

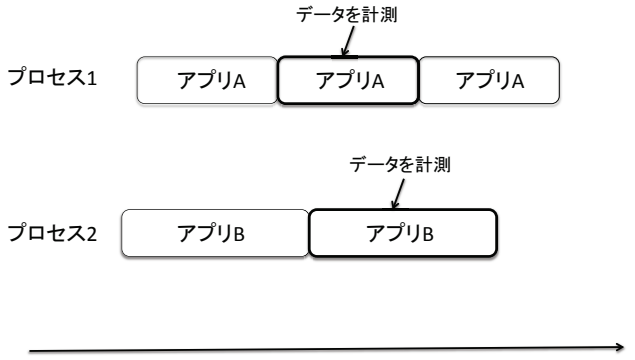

図 2 2 プロセス同時実行での計測方法

パフォーマンスカウンタはプロセス毎の値が，消費エネルギーは全体の値が得られる．式（5）の a_0 が係数の項は Idle 時電力である．2 プロセスでそれぞれ得られたカウンタの値を用いて全体の電力の推定を行うが，各プロセスの推定値の和を単純にとると Idle 時電力が二重に積算されてしまう．したがって，各プロセスで推定された消費電力と RAPL で計測した全体の消費電力 P は以下のような関係があると思われる．

$$\hat{P_{\mathrm{A}}} = P_{\mathrm{A}} - P_{\mathrm{Idle}} \tag{7}$$

$$\hat{P_{\mathrm{B}}} = P_{\mathrm{B}} - P_{\mathrm{Idle}} \tag{8}$$

$$P = \hat{P_{\mathrm{A}}} + \hat{P_{\mathrm{B}}} + P_{\mathrm{Idle}} \tag{9}$$

表 4 式中のパラメータの説明

P	全体の消費電力
P_{A}	アプリケーション A の消費電力の推定値
P_{B}	アプリケーション B の消費電力の推定値
P_{Idle}	Idle 時電力

各プロセスでの Idle 時電力を含まない消費電力を推定するモデル式は，式（5）のパフォーマンスカウンタに関する項の和で表せて次のようになる．

$$\hat{P_{\mathrm{A}}} = \frac{\sum_{i=1}^{n} a_i X_i}{T_{\mathrm{exe}}} \tag{10}$$

推定した各プロセスの消費電力と Idle 時電力の合計と，RAPL で計測した全体の消費電力を比較することでプロセス毎の消費電力推定の有効性を調べる．

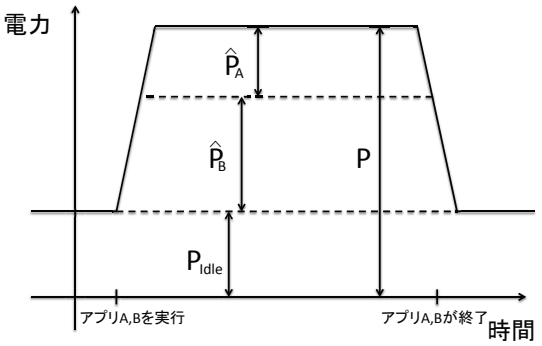

図 3 2 プロセスでの計測時の各消費電力の関係

5. 評価

5.1 実験環境

本研究では，**表 5** に示す実験用サーバを用いて実験を行った．

表 5 マシンの仕様

CPU	Intel Xeon CPU E5-2640 v2(2GHz) × 2
メモリ	128GB
物理コア数	8 × 2
カーネル	2.6.32-358.el6.x86_64
OS	CentOS6.4

パフォーマンスカウンタの計測には, PAPI-5.3.2 と Score-P-1.3 を用いた．

5.2 消費電力のモデル化

アプリケーション実行中の消費エネルギーとパフォーマンスカウンタを計測を行う．用意したアプリケーションの中で計算のサイズを変更できるものは変更して複数の実行パターンのデータを取る．今回の実験では**表 6** のアプリケーションを分析用データ計測に用いた．

表 6 分析用に用意したアプリケーション

mat_mult_ijk mat_mult_ikj mat_mult_jik mat_mult_jki mat_mult_kij	行列積計算 ijk などはプログラム中での ループ計算を行う順序
diffusion	拡散計算
advection	移流計算
stream	メモリ帯域幅と計算速度の計測 [3]
FFVC-MINI	熱流体解析プログラム [4]
sleep300	300 秒間 sleep するプログラム

計測したデータを統計ソフトの R で分析し，より有意な結果となるパフォーマンスカウンタを選ぶ．説明変数と

して用いたパフォーマンスカウンタと R での分析結果は**表 7**，**表 8** に示すとおりとなった．

表 7 説明変数に用いたパフォーマンスカウンタ

PAPI_L1_TCM	Level 1 cache misses
PAPI_L2_TCM	Level 2 cache misses
PAPI_SR_INS	Store instructions
PAPI_LD_INS	Load instructions
PAPI_DP_OPS	Floating point operations; optimized to count scaled double precision vector operations
PAPI_TOT_INS	Instructions completed
PAPI_TOT_CYC	Total cycles

表 8 回帰分析の結果

説明変数	推定係数
PAPI_L1_TCM	-4.037×10^{-7}
PAPI_L2_TCM	3.594×10^{-7}
PAPI_SR_INS	9.422×10^{-8}
PAPI_LD_INS	-8.961×10^{-9}
PAPI_DP_OPS	-1.058×10^{-8}
PAPI_TOT_INS	1.839×10^{-9}
PAPI_TOT_CYC	7.811×10^{-9}
実行時間	5.535×10

5.3 作成したモデルを用いた電力推定

5.3.1 1 プロセス

消費電力の実測値とモデル式を用いた推定値を比較する．まず，1 プロセスでアプリケーションを実行したときの電力を推定する．使用するアプリケーションは分析に用いたものとは異なるものにする．各アプリケーションについてのパフォーマンスカウンタと消費電力を計測し，パフォーマンスカウンタをモデルに当てはめて推定した消費電力と実際に計測した消費電力を比較する．

表 9 検証用に用意したアプリケーション

mat_mult_kji	行列積計算
CCS QCD	格子量子色力学計算 [4]
NTChem-MINI	分子化学計算 [4]

表 10 1 プロセスでの電力推定結果

	電力		
アプリケーション名	推定値	実測値	誤差
mat_mult_kji	66.04	67.91	-2.75%
CCS QCD	71.67	68.16	5.16%
NTChem-MINI	69.95	72.40	-3.38%

実験の結果，最大で 5.16%の誤差で消費電力の推定ができた．電力推定の精度はパラメータ決定時に用いたアプリケーションと内容的に近いものほど高くなると考えられる．mat_mult_kji はパラメータ決定時に用いたアプリケーションとほとんど同じものなので誤差が少ないことに対して，CCS QCD はプログラムの構造が大きく異なるため誤差が大きくなったと思われる．

5.3.2 2 プロセス

2 プロセスで同時にアプリケーションを実行した時の消費電力を推定する．アプリケーションは 1 プロセス時に分析用と検証用に用いた全てのものを組み合わせて実験を行った．各プロセスで推定した消費電力から全体の消費電力を推定し，その値と実測値を比較する．

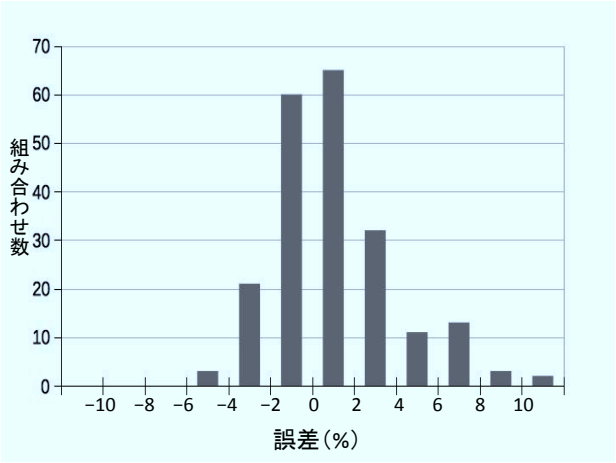

図 4 2 プロセスでの消費電力推定結果

実験の結果は**図 4** に示すとおりとなった．調査した組み合わせのうち 84.8%が誤差の絶対値が 4%以内となり，標準偏差は 2.96 となった．また，推定値が実測値より大きくなる傾向があることがわかった．

6. 関連研究

従来様々な手法で消費電力を推定する研究が行われてきた．ここで既存の研究での消費電力のモデル化について記す．

Karan Singh らはパフォーマンスカウンタを用いて CPU のコアの消費電力のモデル化を行っている [5]．使用したカウンタは，L2_CACHE_MISS:ALL, RETIRED_UOPS, RETIRED_MMX_AND_FP_INSTRUCTIONS:ALL, DISPATCH_STALLS の 4 種類である．CPU コアの消費電力とサイクルあたりの各カウンタの値を計測し，区分線形関数を用いて消費電力をモデル化している．作成したモデルを複数のベンチマークで検証した結果，最大で誤差 7.2%という結果となった．

長坂らはパフォーマンスカウンタを用いて GPU の消費電力のモデル化を行っている [6]，[7]．説明変数をパフォーマンスカウンタ，目的関数を消費電力に設定した線形回帰分析で GPU の消費電力を推定するモデルを作成している．この研究ではパフォーマンスカウンタの取得に CUDA

Prodiler を使用している．取得したカウンタから説明変数とするカウンタを選び，作成したモデルで消費電力の推定を行い平均誤差率 7.2%という結果となった．

カオタンらは RAPL の有効性を調べ，RAPL での消費電力の計測値からノード全体の消費電力を推定するモデルの作成をしている [1]．RAPL の有効性を調べるための実験では，RAPL の計測値と外部の電力測定値を比較することで RAPL が高精度で電力の計測が行えることを示した．RAPL ではプロセッサソケットと DRAM のみが電力計測の対象であるが，実験によりノード全体の電力とも高い相関関係があることがわかり，RAPL 計測値からノード全体の電力を推定するモデルの作成も行っている．得られたモデルでは，データポイントの 88.33%が誤差 2.5%以内，9.13%が誤差 2.5%から 5.0%の範囲に収まっており，ほとんどの場合で誤差 5%以下と高い精度でノード全体の電力が推定可能なことが判明した．

これらはいずれも物理的な単位での消費電力のモデル化であり，プロセスという論理的な単位でのモデル化を行った我々の手法とは異なる．

7. まとめと今後の課題

本研究ではパフォーマンスカウンタを用いたプロセス毎の消費電力のモデル化を行った．作成したモデル式を用いた電力の推定実験を 1 プロセス時と 2 プロセス同時実行時についてそれぞれ行い，1 プロセス時は最大で誤差 5.16%，2 プロセス時は計測した組み合わせのうちの 84.8%が誤差 4%以内となった．

今後の課題として，分析に使うアプリケーションの種類を増やして精度の向上を目指す．複数プロセス時には推定値が大きくなる傾向があるので，その原因を調査しモデル式の改善を行う．また，今回は 2 プロセスで実験を行ったがプロセスをさらに増やした場合の消費電力についての推定実験を行っていく．

謝辞 本研究の一部は科学研究費補助金基盤研究 (S)23220003 「10 億並列・エクサスケールスーパーコンピュータの耐故障性基盤」及び科学技術振興機構戦略的想像研究推進事業「EBD:次世代の年ヨッタバイト処理に向けたエクストリームビッグデータの基盤技術」による．

参考文献

[1] カオタン, 和田康孝, 近藤正章, 本多弘樹. 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, sep 2013.
[2] 大坂隼平, 和田康孝, 近藤正章, 三吉郁夫, 本多弘樹. Hpc アプリケーションの消費電力最適化に向けた性能・消費電力情報の統合手法. ハイパフォーマンスコンピューティングと計算科学シンポジウム論文集, 第 2015 巻, pp. 159–166, may 2015.
[3] John D. McCalpin. Stream:sustainable memory bandwidth in high performance computers. Technical report, University of Virginia, Charlottesville, Virginia, 1991-2007. A continually updated technical report. http://www.cs.virginia.edu/stream/.
[4] 小村幸浩, 鈴木惣一朗, 三上和徳, 滝澤真一朗, 松田元彦, 丸山直也. Fiber ミニアプリの性能評価, jul 2014.
[5] Karan Singh, Major Bhadauria, and Sally. A McKee. Real time power estimation and thread scheduling via performance counters. *SIGARCH Comput. Archit. News*, Vol. 37, No. 2, July 2009.
[6] 長坂仁, 丸山直也, 額田彰, 遠藤敏夫, 松岡聡. GPU における性能と消費電力の相関性の解析. 情報処理学会研究報告, No. 27, pp. 1 – 5, jul 2009.
[7] 長坂仁, 丸山直也, 額田彰, 遠藤敏夫, 松岡聡. GPU におけるモデルに基づいた電力効率の最適化. 情報処理学会研究報告, No. 2, pp. 1 – 6, dec 2010.