Vol.2014-ARC-213 No.7
Vol.2014-HPC-147 No.7
2014/12/9

HPCシステムにおける性能プロファイリングツールの電力測定精度の評価

大坂 隼平[1]　和田 康孝[2]　近藤 正章[3]　三吉 郁夫[4]　本多 弘樹[1]

概要：将来の HPC システムにおけるシステムの設計や実効性能を制約する主な原因の一つとして, 消費電力が挙げられる. よって HPC アプリケーションの性能向上を図る上では, 電力を意識することも重要である. ピーク電力が電力制約を超えずにシステムを動作させる従来の設計思想では, アプリケーションに対して限られた電力資源を適切にを配分することは難しい. よって, 我々はピーク消費電力が制約を超過する事を許し, 電力性能ノブを適切に調節することで限られた電力資源を有効に使用できるようにする電力制約適応型システムの実現を目指している. この電力制約適応型システムを実現させるためには, アプリケーションの実行時のプロファイルやトレーシングによるアプリケーションの性能情報に加え, 消費電力に関する情報をユーザに提示する必要がある. 本稿では, 既存のプロファイリングツールである TAU を用いて電力測定を行う際の測定精度評価を行った. その結果 TAU による電力測定は, 時間的オーバーヘッドが大きく, また, 電力情報も不正確であることが分かった. そこで, 電力方向と時間方向の両方に補正をかけ, 精度を向上させるための手法について検討・評価を行った.

1. はじめに

現在の HPC システムでは, 並列処理により複数のノードを同時に使用することで, 高い性能を得ている. しかし, この膨大な数のノードを用いた並列処理を行うシステムは電力の消費が激しく, 例えば高機能大型計算機 (スパコン) の一つである理化学研究所に設置された京コンピュータは, 消費電力がおよそ 13MW?と一般家庭での使用電力量のおよそ 3 万世帯分まで相当している. このことから, 将来の HPC システムではこれ以上ノードを増やす, すなわち消費電力を増加させることによる性能向上が困難となると予想される.

これに対し, 限られた電力を HPC アプリケーションの特性に応じて電力性能ノブを調節することで適切に配分し, 実効電力を抑えつつ高い実効性能を獲得する電力制約適応型システムの開発が進められている?.

[1] 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications
[2] 早稲田大学基幹理工学研究科
Graduate School of Fundamental Science and Engineering Waseda University
[3] 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo
[4] 富士通株式会社次世代テクニカルコンピューティング開発本部
Next Generation Technical Computing Unit, Fujitsu Limited

この電力制約適応型システムを実現させるためには, 電力性能ノブを適切に操作し, 与えられた消費電力の上限を超えないようにアプリケーションの実行性能を最大化する必要がある. このような最適化を正しく適用するためには, ユーザに提供するプロファイル情報や実行トレース情報に加え, 実行時の電力消費状況を実際の電力値となるべく誤差が出ないように測定を行えるのが望ましい.

このプロファイル情報や実行トレース情報を取得しユーザに提示する手段として, パフォーマンス解析ツールが用いられる. しかし, パフォーマンス解析ツールによるプロファイリングやトレーシングによって実行時間が増加し, これにより電力に影響を及ぼす可能性があり, その結果測定した電力測定値がパフォーマンス解析ツールを使わずに実行した際の実際の電力測定値と異なってしまう恐れがある. 電力測定値が実際の電力測定値と異なると, 限られた電力を適切に配分することが困難となり大きな問題となってくる.

本稿では，性能プロファイリングツールである TAU と, Intel 社のプロセッサに搭載されているインターフェースの RAPL(Runnning Average Power Limit)?とを連携させ, 性能と電力の測定を同時に行うことで, 電力制約適応型システムの実現に必要な情報の取得を試みた.

また, その際に取得した電力測定値の精度について, RAPL 単体での電力測定値と比較することで評価を行っ

た. その結果, RAPL を連携させた TAU による電力測定値の精度が十分ではないことが判明したため, RAPL を連携させた TAU の測定値を電力方向と時間方向の両方に補正をかけ, 性能情報とともに電力情報を高い精度で提示することができる手法について検討・評価を行った.

本稿の構成を以下に示す. 2 章ではパフォーマンス解析ツールについて, 3 章では TAU による性能・電力測定を行う際の精度の評価について, 4 章では TAU による測定結果に補正をかけ, 精度を向上させる手法について, 5 章ではまとめと今後の課題について述べる.

2. パフォーマンス解析ツールと関連研究

2.1 性能測定と電力測定

アプリケーションのボトルネックや負荷バランス等の特性は, TAU?や Scalasca?, Score-P?などのパフォーマンス解析ツールを用いることで取得が可能となる. これらは, 各関数の実行回数や各関数の実行時間が全体の実行時間に対してどれだけの割合を占めるかを示すプロファイル情報や, 各関数の開始・終了地点等のイベントやパフォーマンスカウンタの情報等タイムラインで逐次記録した実行トレース情報を取得できるツールである. HPC システムにおいては, 上記に示したツールが研究開発されていて, MPI や OpenMP の同期, 通信等の情報を得ることができる.

また, 電力情報は RAPL(Running Average Power Limit) インターフェースを介することで取得が可能となる. これは, Intel 製プロセッサの Sandy Bridge マイクロアーキテクチャ以降の世代のプロセッサに搭載された機能であり, この機能を用いることでプロセッサや DRAM の消費電力の測定や消費電力の制御を行うことができる. RAPL インターフェースは, 消費電力を測定・制御できる対象が 3 箇所あり, サーバ環境ではチップ全体 (Package,PKG), チップ上のコアの部分 (Power Plane 0,PP0), メモリ (DRAM) がそれに該当する. ユーザは MSR(Model Specific Register) を介してこれらの操作を行うことができる?.

2.2 TAU による実行トレースと電力情報の取得

HPC システム向けのパフォーマンス解析ツールの中でも特に TAU は PDT(Program Database Toolkit) と連携させることでより詳細で動的なプロファイルやトレース情報を取得することができる?. また, PAPI(Performance Application Programming Interface)?と連携させることで, RAPL の機能を使用することが可能となり電力情報の取得ができる. PAPI はアプリケーションプログラムのパフォーマンスに関する様々な情報の取得を可能にするもので, MSR にアクセスすることで RAPL から電力情報を取得することができる?.

しかし, このようにパフォーマンス解析ツールを用いて性能の詳細情報取得機能や電力測定機能を埋め込んで使用すると, 測定にかかる実行時間の増加は避けられない. このオーバーヘッドによりプロファイルやトレース情報, 電力情報が実際の値と大きく異なってしまう恐れがある. アプリケーションプログラムの挙動に対し, 適切に限られた電力を配分するためには, アプリケーションプログラムの性能情報と精度の高い電力情報が記載されたデータをユーザ側に提示できるシステムが必要となってくる.

また, TAU で性能情報や電力情報を取得してもそれをユーザ側に分かり易く提示しなくてはアプリケーションの性能や消費電力の最適化を行う上で非常に困難になることが予測される. よって, 取得した測定結果を提示するためのツールも必要となってくる.

従来から様々な実行トレース情報の記録を行うフォーマットが提案・利用され, 例えば TAU であれば Tau Trace format が利用されてきていた. しかし, このフォーマットでは利用できない可視化ツールも多く, 可搬性が乏しいためよりオープンなフォーマットを用いるのが望ましい. 中でも, OTF2(Open Trace Format Version 2) は多くの可視化ツールにサポートされているため, このフォーマットを利用することで様々な可視化ツールで実行トレース情報を作成・提示することが可能となる. TAU には Tau Trace Format から OTF2 に変換する機能が備えられており, TAU で測定した性能情報や電力情報を記録した OTF2 ファイルを作成することができる.

2.3 関連研究

電力制約適応型システムに関連する研究の中で, 限られた電力でアプリケーションプログラムの処理性能を落とさずに電力配分を調節する最適化手法の指標を得る研究が行われた?. 性能情報が予め判明してあるアプリケーションプログラムに対し, RAPL を用いて電力情報を取得し, それぞれの情報から電力を適切に配分したところ, 電力制約下でのアプリケーションプログラムの性能向上を達成した.

また, ベンチマークプログラムから性能情報や電力情報をパフォーマンス解析ツールを用いて取得を行う際, トレース情報や電力情報の取得を行う際に発生する観測オーバーヘッドの削減手法および, 取得した測定結果を可視化できるツールの検討を行った先行研究が行われた?. また, RAPL インターフェースにおいて, 電力の測定や制御の精度について, 電力メーターと比較評価が行われ, その結果 RAPL による電力の測定・制御は高い精度で測定できることが判明している?.

3. RAPL を連携させた TAU の精度評価

本章では，TAU を用いて実行トレース情報と電力情報の取得を行った際に発生するオーバーヘッドと, それが電力測定の精度に与える影響について, 実際に HPC 向けのベンチマークを用いて RAPL 単体による電力測定結果と

表 1　シングルノードによる実験での評価環境

Processor	Intel(R) Xeon(R) CPU E5-2643
Num of Cores	8
L1 Chache	32KB I + 32KB D cache per core
L2 Cache	256KB
L3 Chache	10240KB
OS	Scientific Linux 2.6.32-358.el6.x86_64
compiler	GCC 4.4.7
MPI	MPICH3.0.4

比較を行うことで評価を行った.

3.1　評価環境

RAPL と連携させた TAU(以下 TAU) で測定した電力の精度について調査を行うために TAU 及び RAPL 単体それぞれ用いて電力測定し, PKG ドメインを対象に比較評価を行った.

評価環境は表 1 の通りで, プロセッサは 8 コアを搭載した Intel(R) Xeon(R) CPU E5-2643 を使用した. 評価に用いたベンチマークは, Nas Parallel Benchmark(NPB) に含まれる BT と EP 及び HPC Challenge(HPCC) ベンチマークを用いた. NPB においてはクラス B の入力セットを使用し, HPCC ベンチマークは配列サイズ N を 5000 として評価を行った. 電力測定の対象となる区間は MPI 関数の開始 (MPI_Init 関数) 直後から通信の終了 (MPI_Finallize) の直前までとした. TAU による電力測定可能な最短電力取得間隔は 1 秒間隔毎であるため, TAU 及び RAPL 単体の電力取得間隔を等しく 1 秒として測定を行った.

なお, 可視化の可搬性を考慮して本稿では, TAU による性能情報と電力情報の取得は, OTF2 形式で取得したデータを用いることを前提とする.

3.2　シングルノードによる電力測定比較結果

まず, 予備評価として TAU 及び RAPL 単体で電力測定にどれだけ時間的・電力的に差が発生するのか比較するために, シングルノードによる 4 プロセスで並列実行した HPCC ベンチマークと NPB の BT および EP の測定を行った.

図 1 から図 3 に各ベンチマークで TAU 及び RAPL 単体で測定を行った際の PKG ドメインの rank0 における電力測定値の比較結果を示す. なお, TAU の電力測定結果は同じ条件でベンチマークを 5 回測定しその各時刻と電力の平均をとったグラフとなっていて, 縦軸は電力 (W), 横軸はプログラムの開始地点 (main 関数呼び出し直後) を時刻 0 とした経過時間としている. また, 試行毎の時間の標準偏差は, 最大約 3.0 電力値の標準偏差は最大約 0.20 と, ばらつきは少なかった. 一方で RAPL 単体による測定における電力誤差は殆ど 0 に近い値となった.

図 1 より, RAPL 単体による電力測定をしながらの実行

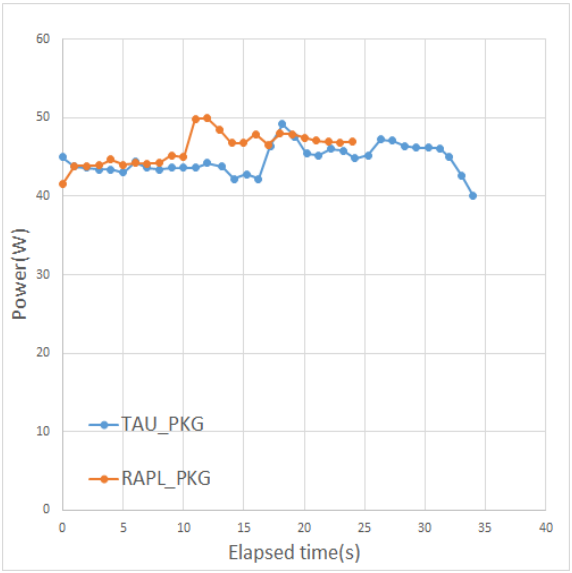

図 1　TAU 及び RAPL 単体を用いた, HPCC ベンチマークの消費電力測定値の比較

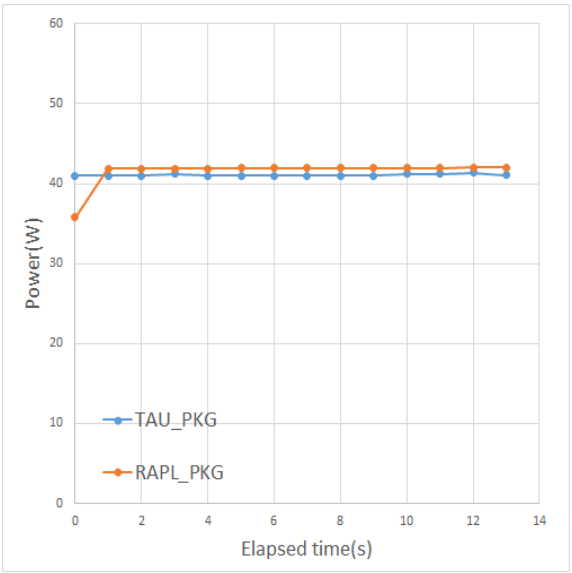

図 2　TAU 及び RAPL 単体を用いた, NPB の EP の消費電力測定値の比較

はおよそ 24 秒であるのに対し, TAU による電力測定を行った際の実行に時間はおよそ 35 秒であった. プログラムの特性によって誤差が一定とは限らないが, ベンチマーク全体では 44%近く実行時間が増加してしまっていることが判明した. また, 測定開始地点や終了地点に大きな電力の差が生じた. 他にも, TAU のグラフにおける 15 秒付近の電力値が下がっているのに対し, RAPL 単体の電力値にはそれに当たる箇所が無い等全体的に電力値の不一致が見受けられる. また, 解析ツールを使わずに普通に実行した場合の実行時間はおよそ 24 秒で, RAPL 単体の測定による実行時間の増加は殆どない結果となった.

一方 NPB の測定結果は, 図 2, 図 3 の通り殆ど電力値の推移が一定となる結果となったが, TAU 及び RAPL 単体

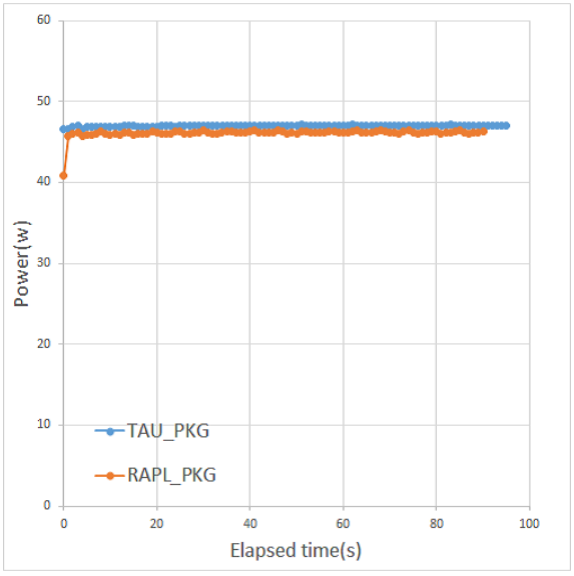

図 3　TAU 及び RAPL 単体を用いた, NPB の BT の消費電力測定値の比較

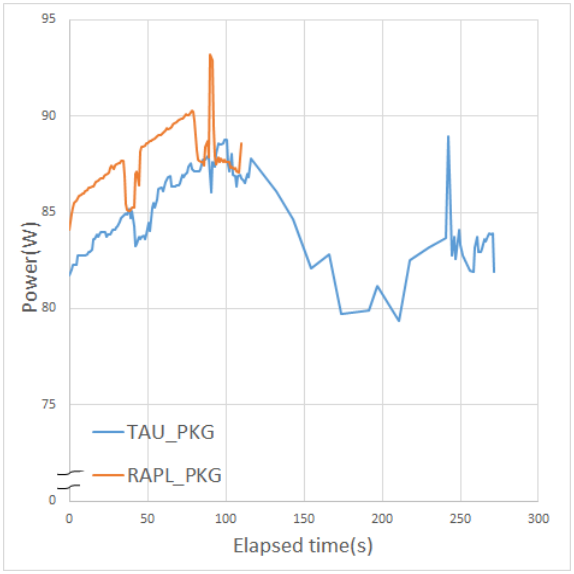

図 4　TAU 及び RAPL 単体を用いた, HPCC ベンチマークを 4 ノードで実行した際の電力測定値の比較

表 2　複数ノードによる実験での評価環境

Processor	Intel(R) Xeon(R) CPU E5-2643
Num of nodes	1476
Num of Cores	8 x 2
OS	Red Hat Linux Enterprise
compiler	Fujitsu, Intel(Fortran, C, C++), PGI Fortran

による測定で電力値に差が発生する結果となった．これは, RAPL 単体による電力測定は小数点第 6 位までの電力値を測定可能であるのに対し, TAU による電力測定は小数点以下を切り捨ててしまうため, 差が発生していると思われる．

3.3　複数ノードによる電力測定

次に, 大規模な計算システムを使用して複数ノードによる電力測定で電力値の推移に影響があるのか調査を行った．評価環境を表 2 に示す．シングルノードによる実験と同様に HPCC ベンチマークに対し 16 コアのマシンを用いて 4 ノード 64 プロセスで並列実行し, TAU 及び RAPL 単体での電力測定比較を行った．測定比較結果を図 4 に表す．図 4 では, 各チップ毎の電力の平均をとっており, RAPL 単体における各チップ毎の電力値の差は最大でおよそ 11W であった．図 4 から, 複数ノード TAU による測定にかかった時間的オーバーヘッドが大幅に大きくなり, これにより TAU で測定した電力値が, RAPL 単体で測定した場合の電力値よりも大幅に下がる結果となった．

4.　電力補正手法の提案・評価

本章では, TAU を用いてアプリケーションの電力測定を行った際に生じた誤差の補正・解消を行う．また, 実行時間や消費電力, その他トレース情報を正確にユーザに提示するための手法を提案する．

4.1　時間方向への補正手法

電力制約適応型システム実現の上で必要なものはアプリケーションの性能に関わるパフォーマンスカウンタの情報や実行トレース情報, 精度の高い電力情報をユーザに提示できるようにする可視化ファイル, 今回の場合は, 様々な可視化ツールに対応した OTF2 ファイルがあれば望ましい．よって, TAU による測定で生成されたプロファイル情報やトレース情報, 精度の悪い電力情報が記録された OTF2 ファイルの電力情報を, 精度の高い電力情報に書き換える．これにより, 電力最適化を行う上で必要な情報が記載された OTF2 ファイルをユーザに提供可能となる．

まず, TAU による電力測定によって発生した実行時間の増加分を取り除くため, TAU による測定でかかった実行時間を RAPL 単体による測定でかかった実行時間と合うように, TAU の測定時刻に対して補正を行う．NPB の BT のようなプログラムの構造が大きく変化しないアプリケーションは, TAU の測定で発生するオーバーヘッドも一定であるため, RAPL 単体による測定にかかった全体の実行時間と TAU による測定にかかった全体の実行時間から比を求め時刻の補正を行う．しかし, HPCC ベンチマークように, 実行中にプログラムの特性が変わるアプリケーションの場合は, TAU による測定で発生するオーバーヘッドも一定ではないと予測される．

この様なアプリケーションを対象とした時間方向への補正手法を図 5 中段に表す．TAU を用いた電力測定で記録された時刻に対して, 実行時間の伸び率が一定であると思われる区間を設け, その区間毎の TAU 及び RAPL 単体で測定を行った際の各実行時間を測定し, 比を求め TAU による測定でかかった時刻を RAPL 単体による測定でかかった時刻と合わせる．これにより, TAU による測定でずれてし

OTF2 ファイルの内容を次のように変更する．
当該プログラムのすべての補正対象区間 seg[i] に対して：
・全ての P(t_TAU[j]) を P(t_RAPL[k]) に書き換える
ただし，t_TAU[j]，t_RAPL[k] は次の条件を満たす値とする．

$$ts\_TAU(seg[i]) < t\_TAU[j] < te\_TAU(seg[i])$$
$$ts\_RAPL(seg[i]) < t\_RAPL[k] < te\_RAPL(seg[i])$$
$$(t\_TAU[j] - ts\_TAU(seg[i])) / (te\_TAU(seg[i]) - ts\_TAU(seg[i]))$$
$$\fallingdotseq (t\_RAPL[k] - ts\_RAPL(seg[i])) / (te\_RAPL(seg[i]) - ts\_RAPL(seg[i]))$$

・ts_TAU(seg[i])：TAU による測定における関数 seg[i] の実行開始時刻
・te_TAU(seg[i]): TAU による測定における関数 seg[i] の実行終了時刻
・ts_RAPL(seg[i])：RAPL による測定における関数 seg[i] の実行開始時刻
・te_RAPL(seg[i]): RAPL による測定における関数 seg[i] の実行終了時刻
・P(t_TAU[j]): TAU による測定における時刻 t_TAU[j] での電力測定値
・P(t_RAPL[k]): RAPL による測定における時刻 t_RAPL[k] での電力測定値

図 5 TAU の電力補正手法

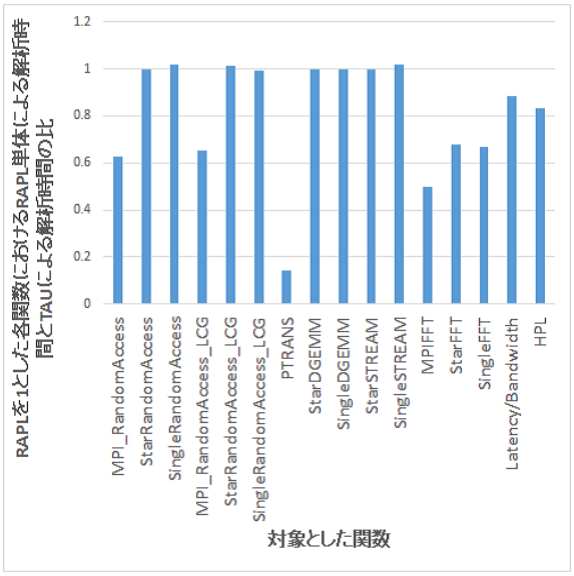

図 7 HPCC ベンチマークにおける各関数の
RAPL 単体による実行時間と TAU による実行時間の比

まった時刻をパフォーマンス解析ツールを用いずに実行した場合の時刻と近い値に補正することができる．

例として，シングルノードで測定した HPCC ベンチマークに対し時間方向の補正を行った．補正区間は HPCC ベンチマークを構成している各関数毎とした．各関数における実行時間の比を求めるため，TAU 及び RAPL 単体各関数の時刻・時間の測定を行った．時刻の測定範囲は各関数の呼び出し直前から終了直後までであり，rank0 を基準として経過時刻の取得を行った．また，測定開始地点は測定の対象となる区間と同じである MPI_Init 関数の開始直後とした．この測定により判明した全体の実行時間における各関数の実行時間の内訳を TAU 及び RAPL で比較した結果を図 6 に表す．上の棒グラフは RAPL 単体で解析した場合の各関数の実行時間の内訳，下の棒グラフは TAU で解析を行った場合の各関数の実行時間の内訳を示している．また，各関数において RAPL の実行時間と TAU の測定にかかった実行時間との比を図 7 に表す．これは，図 6 で求めた各関数の RAPL で測定した場合の実行時間を TAU で測定した場合の実行時間で割った値を示している．

図 6 および図 7 から HPCC ベンチマークにおける PTRANS 関数の TAU による測定にかかった実行時間が RAPL による測定にかかった実行時間比べて他の関数よりも大幅に増加しているのが分かる．PTRANS 関数は大規模な配列データの転送速度を測定する関数で通信回数が多くトレーシングによる情報量が他の関数に比べ多いため実行時間の増加が膨大になってしまうと考えられる．この様に，関数毎に TAU による測定にかかった実行時間の増加量が異なる区間を対象に，TAU による測定にかかったの実行時間を RAPL 単体による測定にかかった実行時間と一致するように時間方向への補正を行った．

この手法を適用して時間方向への補正を行った結果を図 8 に表す．対象とした関数以外の箇所で発生したオーバーヘッドの影響で補正後の TAU 及び RAPL 単体の電力測定における実行時間に多少の誤差はあるが，補正後の TAU の最終電力取得時刻は 24.45 秒，RAPL は 24.00 秒とほぼ一致させることができた．電力値の推移に焦点を当ててみると，両方とも大体 30 秒付近で電力が急激に上がっていることが分かる．ただし，依然として消費電力のデータに誤差があり，これを更に補正する必要がある．

4.2 電力方向への補正

TAU による測定にかかった実行時間に対して時間方向への補正を行い，RAPL 単体による測定にかかった実行時

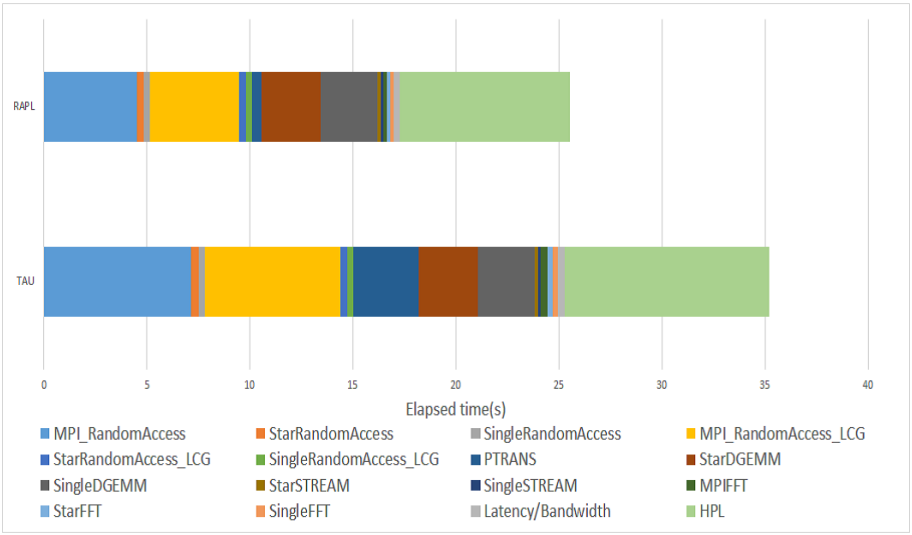

図 6 TAU 及び RAPL 単体を用いた HPCC ベンチマークにおける
それぞれの全体の実行時間に対する各関数の実行時間の内訳比較

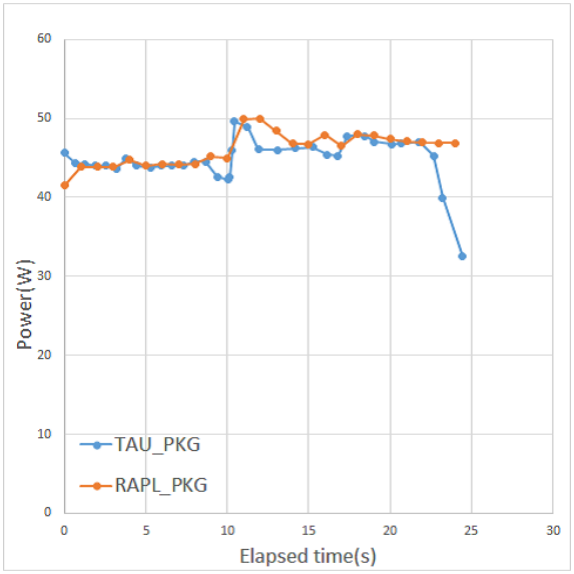

図 8 TAU 及び RAPL 単体を用いた HPCC ベンチマークにおける
時間方向・電力方向への補正を適用した電力値測定比較

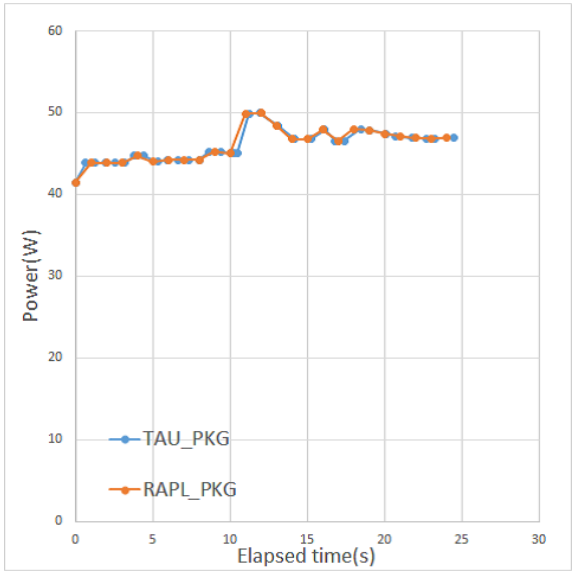

図 9 HPCC ベンチマークにおける電力方向への補正を行った TAU
及び RAPL 単体での電力値測定比較

間とほぼ一致したことにより, 各関数の測定で発生した実行時間の誤差をほぼ解消することが可能となった. しかし, 図 8 から分かる通り, TAU を用いて性能情報を取得した影響で電力値がが変わってしまった箇所が多く見受けられる.

そこで, TAU の電力測定結果に対して, RAPL 単体での測定結果と一致するように電力方向への補正を行った. 手法として, 図 5 に示す通り., 時間方向に補正を行った TAU の各時刻における電力情報をその時刻に最も近い RAPL の時刻の電力情報に書き換えた. 電力方向への補正を行った結果を図 9 に示す.

図 9 より, 補正を行ったことで, RAPL 単体による測定結果のグラフとほぼ一致したことが分かる. RAPL 単体による測定結果は先行研究より, 性能解析を行わない分精度の高い電力測定が可能であるため, これにより TAU による測定で生成した OTF2 ファイルの電力データが記載された箇所を, 補正後の電力値に書き換えることで, より精度の高い電力値が記載された情報を取得することが期待できる.

5. おわりに

将来の HPC システムでは, システムの設計や実効性能を制約する主な原因の一つとして, 消費電力が挙げられる. よって, HPC アプリケーションのコードを最適化する上では, 電力を意識することも重要である. そこで我々は, ピーク消費電力が制約を超過する事を許し, 電力性能ノブを適切に調節することで限られた電力資源を有効に使用できるようにする電力制約適応型システムの実現を目指す. 本稿では, 電力制約適応型システムの実現に必要な実行トレー

ス情報と精度の高い電力情報を取得するための手法を提案・評価した.

評価結果から, 提案手法により性能プロファイリングツールである TAU による性能測定の影響で精度が悪くなった電力測定値を精度の高い電力測定値に補正することができた. 今回は, HPCC ベンチマークが構成する各関数を補正区間として時間方向への補正手法を適用したが, 電力測定結果と性能測定結果を照らし合わせ補正を行う最適な区間を決定するためのアルゴリズムを考案する必要がある.

今後の課題としては, 上記アルゴリズムの考案と, そのアルゴリズムを利用して今回提案した電力補正手法を自動的に行えるようなシステムの作成が挙げられる.

謝辞 本研究の一部は，JST CREST 研究課題「ポストペタスケールシステムのための電力マネージメントフレームワークの開発」の支援により行われたものである．本研究に対してツールのサポートを頂いた九州大学院システム情報科学研究院の皆様に感謝致します.

参考文献

[1] Top 500 Supercomputer Sites: http://www.top500.org/.
[2] http://www.postpeta.jst.go.jp/reserchers/kondo24.html.
[3] David, H., Gorbatov, E., Hanebutte, U. R., Khanna, R. and Le, C.: RAPL: Memory Power Estimation and Capping, *Proceedings of 2010 ACM/IEEE International Symposium on Low-power Electronics and Design, pp.189-194(2010)*
[4] Rotem, E., Naveh, A., Rajwan, D., Ananthakrishnan, A. and Weissmann, E.: Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge, *IEEE Micro, Vol. 32, No 2, pp. 20-27.*
[5] Shende,, S.S. and Malony, A. D.: The Tau Parallel Performance System, *International Journal of High Performance Computing Applicationsns, Vol. 20, No. 2, pp.287-331(2006).*
[6] Geimer, M., Wolf, F., Wylie, B. J. N., Abraham, E., Becker, D. and Mohr, B.: The Scalasca Performance Toolset Architecture, *Concurrency and Computation: Practice and Experience, Vol. 22, No 6, pp. 702-719*
[7] Knupfer, A., Rossel, C., an Mey, D., Biersdorff, S., Diethelm, K., Eschweiler, D., Geimer, M., Gerndt, M., Lorenz, D., Malony, A., Nagal, W. E., Oleynik, Y., Philippen, P., Saviankou, P., Schmidl, D., Shende, S., Tschuter, R., Wagner, M., Wesarg, B. and Wolf, F.: Score-P: A Joint Performance Measurement Run-Time Infrastructure for Periscope, Scalasca, TAU, and Vampire, *Proceedings of the 5th International Workshop on Parallel Tools for High Performance Computing, pp.79-91(2011).*
[8] David, H., Gorbatov, E., Hanebutte, U.R., Khanna, R. and Le, C.: RAPL: Memory Power Estimation and Capping, *Preceedings of 2010 ACM/IEEE International Symposium on Low-power Electronics and Design, pp.189-194(2010).*
[9] Lindlan, K. A., Cuny, J., Malony, A. D., Shende, S., Mohr, B., Rasmussen, C. and Rivenburgh, R.: A Tool Framework for Static and Dynamic Analysis of Object-Oriented Software wih Templates, *Proceedings of the 2000 ACM/IEEE Conference on Supercomputing, p. 49(2000).*
[10] Weaver, V. M., Jhonson, M., Kasichayanula, K., Ralph, J., Luszczec, P., Terpstra, D. and Moore, S.: Measuring Energy and Power with PAPI, *Proceedings of the 41st International Conference on Parallel Proceedings Workshops(ICPPW), pp. 262-268(2012).*
[11] Intel Corpolation: *Intel64 and IA-32 Architectures Software Developer's Manual(2013).*
[12] 稲富雄一, 吉田匡兵, 深沢圭一郎, 上田将嗣, 青柳睦, 井上弘士:電力指向型次世代スーパーコンピュータを想定した HPC アプリケーションの性能最適化　～量子化学計算の場合～, 情報処理学会研究報告ハイパフォーマンスコンピューティング (HPC), Vol. 2013-HPC-207. No. 30, pp.1-6.
[13] 和田康孝, カオタン, 近藤正章, 本多弘樹,:HPC システムにおける電力・性能可視化ツールに関する比較検討, 情報処理学会研究報告ハイパフォーマンスコンピューティング (HPC), Vol. 2013-HPC-207. No. 31, pp.1-7.
[14] カオタン, 和田康孝, 近藤正章, 本多弘樹:RAPL インターフェースを用いた HPC システムの消費電力モデリングと電力評価, 情報処理学会研究報告ハイパフォーマンスコンピューティング (HPC), Vol. 2013-HPC-141, No. 20, pp. 1-8.

正誤表

論文題目:HPC システムにおける性能プロファイリングツールの電力測定評価

1 頁

左段 **7** 行目

- (誤)13MW?
- (正)13MW[1]

左段 **15** 行目

- (誤) 進められている?.
- (正) 進められている [2].

右段 **20** 行目

- (誤)RAPL(Running Average Power Limit)?
- (正)RAPL(Running Average Power Limit) [3,4]

2 頁

左段 **14** 行目

- (誤)TAU?や Scalasca?, Score-P?
- (正)TAU[5] や Scalasca[6], Score-P[7]

左段 **33** 行目

- (誤) これらの操作を行うことができる?.
- (正) これらの操作を行うことができる [8].

左段 **38** 行目

- (誤) 取得することができる?.
- (正) 取得することができる [9].

左段 **39** 行目

- (誤)PAPI(Performance Application Programming Interface)?
- (正)PAPI(Performance Application Programming Interface)[10]

左段 **44** 行目

- (誤) 取得することができる?.
- (正) 取得することができる [11].

右段 **29** 行目

- (誤) 研究が行われた?.
- (正) 研究が行われた [12].

右段 **37** 行目

- (誤) 研究が行われた?.
- (正) 研究が行われた [13].

右段 **41** 行目

- (誤) 判明している?.
- (正) 判明している [14].

3 頁

左段表 **1, 2** 行目

- (誤)Num of Cores 8
- (正)Num of Cores 4 x 2

4 頁

左段表 **2, 2** 行目

- (誤)Num of nodes 1476
- (正)Num of nodes 4

左段表 **2, 5** 行目

- (誤)compiler Fujitsu, Intel(Fortran, C, C++), PGI PGI Fortran
- (正)compiler Intel Compiler 12.1.5

6 頁

左段図 **8,** キャプション **2** 行目

- (誤) 時間方向・電力方向への
- (正) 時間方向への