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

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．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，既存のパフォーマンス解析ツールである TAU を用い，性能情報と電力測定を同時に行う際のオーバヘッドについて評価を行った．さらに，その結果に基づき，パフォーマンス解析ツールによって得られる性能情報と，高精度な消費電力情報とを統合する手法を提案する．

Integrating Power Information to Performance Profile Data for HPC Application Optimization

Junpei Osaka[1]　Yasutaka Wada[2]　Masaaki Kondo[3]　Ikuo Miyoshi[4]　Hiroki Honda[1]

Abstract: Power consumption is expected to be one of the most important design constraints for future HPC systems. To tackle this problem, it is required to develop a power-constraint adaptive system, which adaptively controls power-performance knobs equipped in hardware devices in order to maximize effective performance within a power constraint. In order to optimize performance of applications for this kind of HPC systems, we need to consider power consumption of the applications together with its performance. This paper analyzes some problems in conventional performance analysis tools, which incur performance and power overhead in collecting the performance information. Based on this analysis, this paper also proposes an integration method, which combines performance information obtained by performance analysis tools and highly accurate power information given by another source. As the result, the proposed method gives us accurate execution traces with integrated power information.

[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

1. はじめに

現在の HPC システムの多くは，並列処理により複数のノードを同時に使用することで，つまり投入するハードウェア量・ノード数を増加させることで高い性能を得ている．しかし，このように多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 13[MW] [1] と一般家庭での使用電力量のおよそ 3 万世帯分にまで相当している．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は困難であると考えられる．

これに対し，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，限られた電力資源を適切に配分し，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 [2].

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という制約を守りつつ，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，プロファイルや実行トレースなどの性能情報に加え，実行時の電力消費を測定し，より正確にユーザに提供することが求められる．

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広く用いられている．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として広く用いられている TAU [3] 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）[4,5] を用い，実行トレースの取得と消費電力の計測を行った．プロファイリングツールを用いた情報収集では，アプリケーションの構造と性能の関係を解析するために必要な情報を取得できるが，情報の収集・記録のオーバヘッドが大きい．一方，RAPL による消費電力の測定では，高精度・低オーバヘッドで情報の取得が行えるものの，アプリケーションの構造・特性に関する情報との関連付けが難しい．

本稿では，このような，HPC アプリケーション向け性能プロファイリングツールを用いて消費電力の情報を収集する際に生じる問題点に述べるとともに，HPC アプリケーションの性能情報と消費電力情報を統合し，より正確にユーザに提示するための手法を提案する．

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および関連研究について，3 章ではパフォーマンス解析ツールによる性能・電力測定を行う際の問題点について，4 章ではパフォーマンス解析ツールによる性能情報と，別途取得した正確な消費電力情報を統合する手法について，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 [3] や Scalasca [6]，Score-P [7] 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おり，MPI や OpenMP における同期や通信等の情報を得ることができる．

また，消費電力に関する情報は 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・制御できる対象区分が 3 種類あり，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 [4].

2.2 パフォーマンス解析ツールを用いた実行トレースと電力情報の同時取得

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 [8]．また，PAPI（Performance Application Programming Interface）[9] 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 [10].

しかし，このようにパフォーマンス解析ツールを用い，性能の詳細情報取得機能や電力測定機能を埋め込んで情報取得を行うと，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 には独自の 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 [11, 12]．性能情報が予め判明してい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．その他，DVFS を適用するプログラムの領域を適切に分割することで，DVFS のオーバヘッドを低減しつつアプリケーションの消費電力を削減する手法 [13] も提案されている．

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 [14]．さらに，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 [15]．

3. パフォーマンス解析ツールを用いた消費電力測定における問題

本章では，パフォーマンス解析ツール 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生じる問題点について述べる．特に，パフォーマンス解析ツールを用いることによって生じ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比較を行うことで評価を行った．

表 1 実験環境

Processor	Intel(R) Xeon(R) CPU E5-2643
Num of Cores	8 cores x 2
Memory	256 GB
OS	Red Hat Linux Enterprise
Compiler	Intel Compiler 12.1.5
Num of Nodes	1 or 4

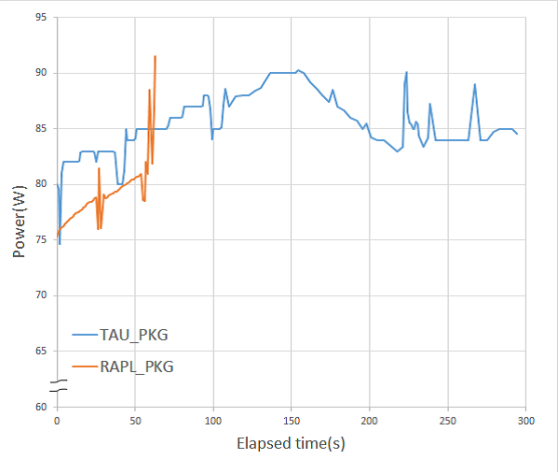

図 1 TAU 及び RAPL 単体による消費電力測定値の比較（HPCC ベンチマーク，シングルノード）

3.1 評価環境

表 1 に実験環境の概要を示す．ここでは，8 コアのプロセッサをノードあたり 2 ソケット搭載する環境を用い，シングルノードおよび 4 ノードの場合について実験を行った．実験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）ベンチマークである．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 とした．NPB の BT および ET はベンチマーク実行中のプログラム挙動がほぼ一定であるために消費電力の変化が少ないのに対し，HPCC ベンチマークは様々な種類のカーネルを含むため，実行するカーネルに応じて消費電力も大きく変化する．

これらのベンチマークに対し，TAU と RAPL を連携させて実行トレースと消費電力を同時に取得した場合（以下「TAU」）と，RAPL から消費電力情報を取得するのみとした場合（以下「RAPL 単体」）とを比較する．実行トレースおよび消費電力情報を取得する間隔は 1 秒とした．

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（16 物理コア，16 プロセス）で並列実行した HPCC ベンチマークと NPB の BT，EP に対し

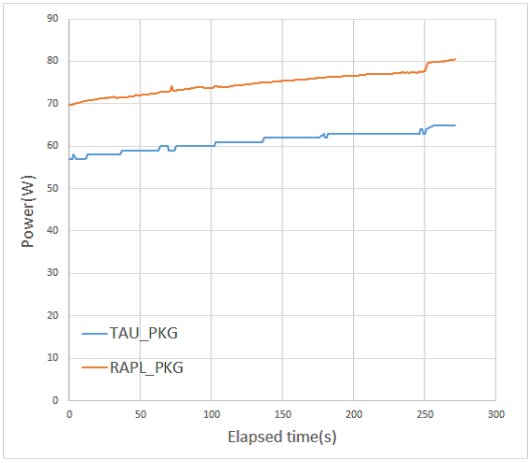

図 2 TAU 及び RAPL 単体による消費電力測定値の比較（NPB EP，シングルノード）

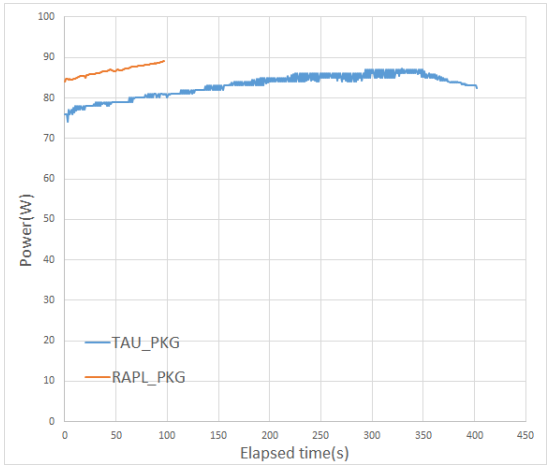

図 3 TAU 及び RAPL 単体による消費電力測定値の比較（NPB BT，シングルノード）

て測定を行った．

図 1，図 2 および図 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消費電力測定値（PKG 部分の消費電力）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おり，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 とした経過時間である．また，試行毎の時間の標準偏差は最大約 1.8，電力値の標準偏差は最大約 5.0 であり，RAPL 単体による測定における電力誤差はほぼ 0 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はおよそ 63 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 294 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およそ 4.67 倍に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．その他，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，RAPL 単体の測定による実行時間の増加はほぼ 0 であった．

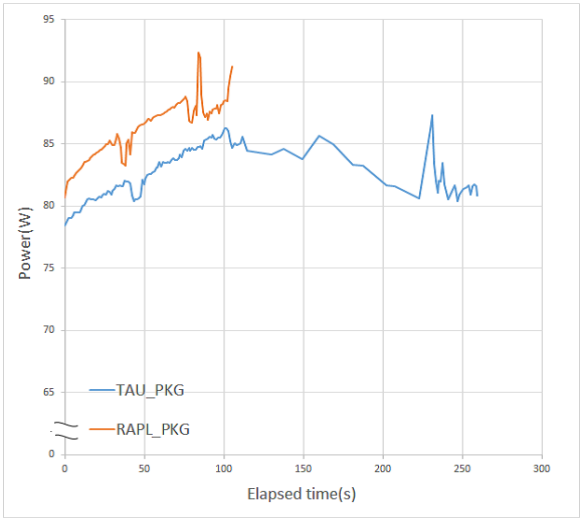

図 4 TAU 及び RAPL 単体による消費電力測定値の比較（HPCC ベンチマーク，4 ノード）

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 単体による測定で電力値に差が発生する結果となった．この原因としては，主として TAU を用いることによる情報取得・情報記録のためのオーバヘッドが考えられる．

3.3 複数ノードによる電力測定

次に，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．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 物理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．

以上より，パフォーマンス解析ツールを用いて性能情報と消費電力情報を同時に取得する方法は，簡単ではあるものの，実行時間のオーバヘッドおよびそれによって生じる消費電力の誤差が大きいという問題点があると言える．電力制約適応型システムにおいては，アプリケーション実行時の消費電力が制約を超えないことを保証しつつアプリケーションを最適化する必要がある．そのため，消費電力に関する情報が実際とは大きく異なる場合，アプリケーションを正しく最適化できず，システム全体の運用に悪影響を及ぼす．

4. 性能情報・消費電力情報の統合手法

3 章で述べた通り，パフォーマンス解析ツールを用いた性能情報・電力情報の取得では，アプリケーションの構造・特性と消費電力を関連付けてユーザに示すことができるが，そのオーバヘッドのために正しい消費電力の値を得ることができないという問題がある．一方，RAPL 単体での測定では，アプリケーションの実行に対するオーバヘッドは無視できる程度である．

つまり，RAPL 単体での測定によって得られる正確な電力情報と，パフォーマンス解析ツールによって得られるアプリケーション内部の情報とを統合してユーザに示すことができれば，電力制約適応型システムに向けたアプリケーション最適化が容易となる．本章では，パフォーマンス解析ツールによって得られる性能情報と，別途取得した正確な電力情報とを統合する手法について提案する．

本手法では，パフォーマンス解析ツールの利用によって生じた実行時間の増加の影響を低減するために，パフォーマンス解析ツールを用いた際の各計測ポイントと，正確な電力情報に含まれる各計測ポイントとを対応付け，時刻情報の統合を行った後，実際の電力情報の値について統合を行う．

4.1 計測ポイントの対応付けと時刻情報の統合

電力制約適応型システム上でアプリケーションのの最適化を行うにあたり特に必要となるものは，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可視化環境である．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．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 ファイルの情報を，精度の高い電力情報と統合することにより，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．

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統合を行う．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，各計測ポイントの対応付けを行う．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の情報統合手法を Algorithm 1 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によって修正し，統合する．

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統合を行った．HPCC ベンチマークでは異なる種類のベンチマークを順次実行しているため，これに基づき，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．この測定により得られたベンチマーク全体の実行時間における各関数の実行時間の内訳について，TAU 及び RAPL で比較した結果を図 5 に示す．図中，上側の棒グラフは RAPL 単体で解析した場合の各関数の実行時間の内訳，下側の棒グラフは 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 6 に表す．これは，RAPL で消費電力を測定した場合に対する，TAU を用いた場合の相対的な実行時間を示している．

図 5 および図 6 から，特に PTRANS 関数において，TAU による測定時の実行時間が RAPL による測定時の実行時間と比較し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．Algorithm 1 を用いることで，プログラム中の各箇所の特性によって生じる差異を考慮して統合を行うことができる．また，TAU による消費電力の測定点と，RAPL 単体による消費電力の測定点の対応付けを取ることができる．

この手法を適用して時間方向への統合を行った結果を図 7 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統合後の 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，両者はほぼ一致している．一方，電力値はまだ統合前であるため，パフォーマンス解析ツールによるオーバヘッドを含んだ不正確な値のままである．

Algorithm 1 計測ポイントの対応付け

Input: $start_{perf}[]$ :: 統合対象区間の実行開始時点に対応する計測ポイント（性能情報取得時）
$start_{power}[]$ :: 統合対象区間の実行開始に対応する計測ポイント（消費電力情報取得時）
$end_{perf}[]$ :: 統合対象区間の実行開始に対応する計測ポイント（性能情報取得時）
$end_{power}[]$ :: 統合対象区間の実行開始に対応する計測ポイント（消費電力情報取得時）
$T^{perf}[]$ :: 性能情報取得時の各計測ポイントの時刻
$T^{power}[]$ :: 消費電力情報取得時の各計測ポイントの時刻
N^{perf} :: 性能情報取得時の計測ポイント数
Output: $T^{perf}_{revised}[]$ :: 統合後の各計測ポイントの時刻（性能情報取得時）

```
for i = 0 to N^perf do
    section ← 計測ポイント i が属する計測区間 ID
    tmp_perf ← (T^perf[i] − start_perf[section])/(end_perf[section] − start_perf[section])
    for j = start_power[section] to end_power[section] do
        tmp_power ← (T^power[j] − start_power[section])/(end_power[section] − start_power[section])
        if tmp_perf ≈ tmp_power then
            T^perf_revised ← T^power[j] ;; 両者を対応付け，時刻情報を統合する
            break
        end if
    end for
end for
```

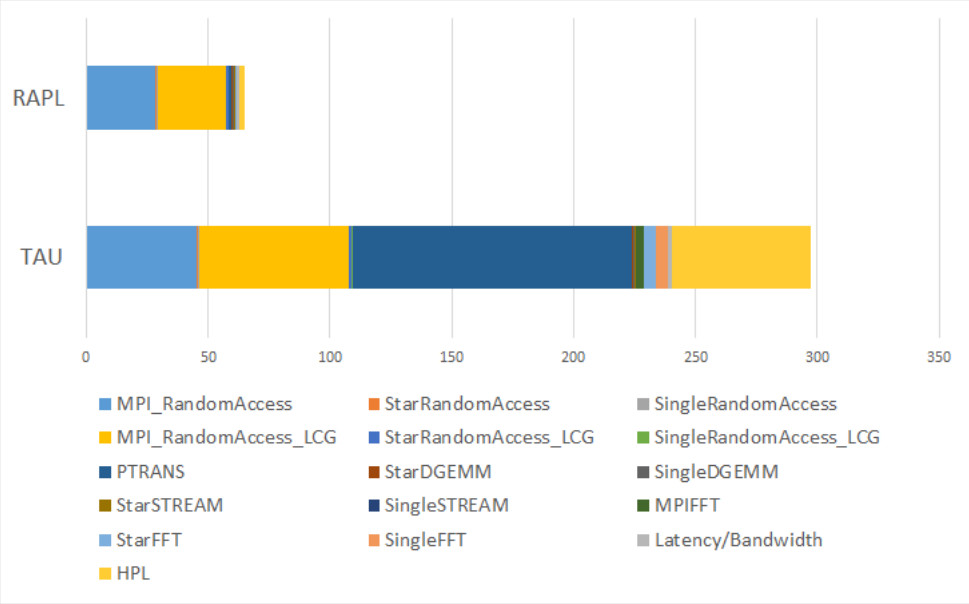

図 5 HPCC ベンチマークにおける各関数の実行時間の内訳比較

4.2 消費電力情報の統合

4.1 節に示す通り，TAU を用いた測定結果に対して時刻情報の統合を行い，実行時の時刻とプログラム上の各地点とをより正確に対応づけることができた．本節ではさらに，時間方向に計測ポイントを対応付け・統合された結果に対して，RAPL 単体での電力測定結果を統合することで，実行時間・計測値ともに正確な計測結果としてまとめる．

情報の統合を行う際には，TAU および RAPL 単体の計測情報のうち，Algorithm 1 によって各計測ポイントの時刻の対応付けを行った結果を用い，TAU によって取得した性能情報と RAPL 単体による消費電力値とを統合する．このようにして，正確な消費電力情報とその他の性能情報と統合・統一してユーザに提示することが可能となる．消費電力情報の統合を行った結果を図 8 に示す．

図 8 より，時間方向の統合とそれに基づく計測値の統合を行ったことで，トレース情報に含まれる消費電力情報が，RAPL 単体によるものとほぼ一致している．これにより，アプリケーションの構造や性能情報とともに，正確な消費電力情報をユーザに提供することが可能となった．図 8 における差の主な原因は，TAU と RAPL 単体それぞれの計測ポイントのタイミングが必ずしも一致しないこと，また，時間方向の統合を適用する区間内で，TAU による計測オーバヘッドが常に一定ではないということが考えられる．

また，本稿においては，比較評価のため TAU を用いた場合でも消費電力情報を取得しているが，実際に提案手法を用いて情報を統合する際には，各情報源から重複なく性

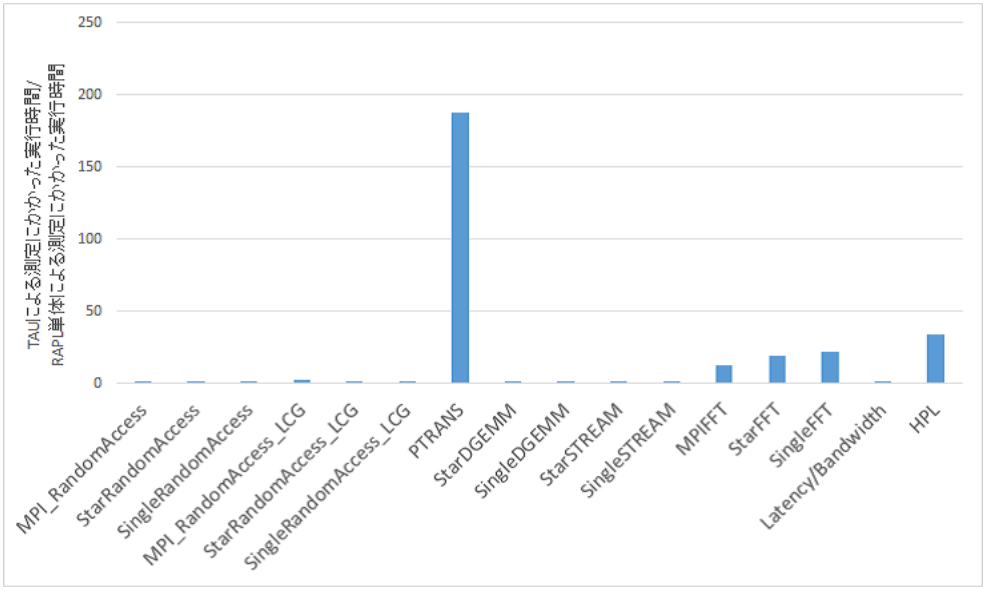

図 6 HPCC ベンチマークにおける各関数の RAPL 単体による実行時間と TAU による実行時間の相対値

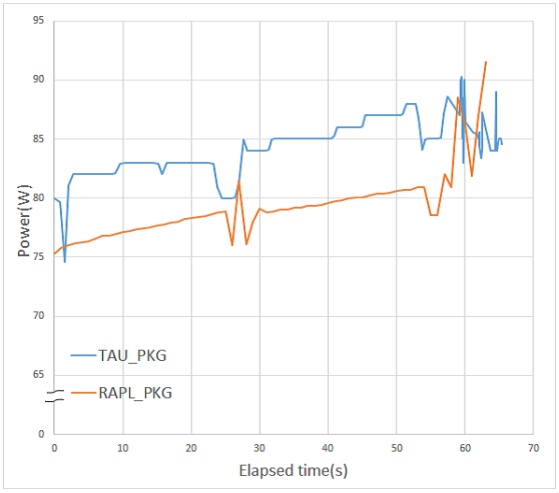

図 7 時間方向への統合を行った結果（HPCC ベンチマーク）

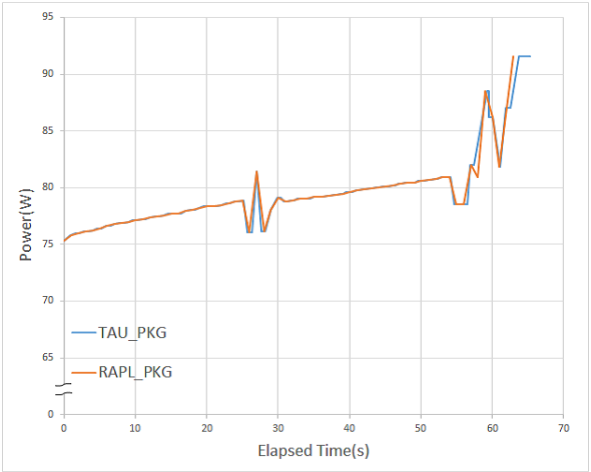

図 8 時間方向の情報統合後，電力情報の統合を行った結果（HPCC ベンチマーク）

能・消費電力に関する情報を取得すればよい．本手法は，トレース情報を保存する共通フォーマットである OTF2 で保存された性能情報に対して適用することができるため，OTF2 をサポートする Scalasca や Score-P といった多くのパフォーマンス解析ツールに対しても有効である．つまり，電力制約適応型システムに対してアプリケーションの性能最適化を行う際，複数の情報源から得られた情報をまとめ，統一した環境でユーザに提供することが可能となる．

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 アプリケーションを最適化する上では，実行性能とともに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容し，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．

本稿では，電力制約適応型システムにおけるアプリケーション最適化に必要となる，アプリケーション実行時の性能情報と消費電力情報を合わせて取得する際の問題点について述べた．また，これに基づいて，パフォーマンス解析ツールによって取得できる性能情報と，高精度な消費電力情報とを統合する手法を提案した．本手法を用いることで，消費電力情報と性能情報，プログラムの構造を関連付け，より正確な情報に基づいてアプリケーションの解析や最適化を行うことが可能となる．

今後の課題としては，統合を行う単位となるプログラム上の区間を適切に選択する手法の開発や，情報取得・統合を自動的に行う環境の構築等が挙げられる．

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] PomPP (POwer Management Framework for Post Peta-Scale Computing) Project: `http://www.hal.ipc.i.u-tokyo.ac.jp/research/pompp/`.
[3] Shende, S. S. and Malony, A. D.: The TAU Parallel Performance System, *International Journal of High Performance Computing Applications*, Vol. 20, No. 2, pp. 287–331 (2006).
[4] 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).
[5] 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 (2012).
[6] Geimer, M., Wolf, F., Wylie, B. J. N., Ábrahám, E., Becker, D. and Mohr, B.: The Scalasca Performance Toolset Architecture, *Concurrency and Computation: Practice and Experience*, Vol. 22, No. 6, pp. 702–719 (2010).
[7] Knüpfer, A., Rössel, C., an Mey, D., Biersdorff, S., Diethelm, K., Eschweiler, D., Geimer, M., Gerndt, M., Lorenz, D., Malony, A., Nagel, W. E., Oleynik, Y., Philippen, P., Saviankou, P., Schmidl, D., Shende, S., Tschüter, R., Wagner, M., Wesarg, B. and Wolf, F.: Score-P: A Joint Performance Measurement Run-Time Infrastructure for Periscope, Scalasca, TAU, and Vampir, *Proceedings of the 5th International Workshop on Parallel Tools for High Performance Computing*, pp. 79–91 (2011).
[8] 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 with Templates, *Proceedings of the 2000 ACM/IEEE Conference on Supercomputing*, p. 49 (2000).
[9] Weaver, V. M., Johnson, M., Kasichayanula, K., Ralph, J., Luszczek, P., Terpstra, D. and Moore, S.: Measuring Energy and Power with PAPI, *Proceedings of the 41st International Conference on Parallel Processing Workshops (ICPPW)*, pp. 262–268 (2012).
[10] Intel Corporation: Intel® 64 and IA-32 Architectures Software Developer's Manual (2013).
[11] 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-142, No. 30, pp. 1–6 (2013).
[12] 稲富雄一，和田康孝，深沢圭一郎，青柳 睦，近藤正章，三吉郁夫，井上弘士：MPI 並列アプリケーションの電力最適化，研究報告ハイパフォーマンスコンピューティング（HPC），Vol. 2014-HPC-146, No. 6, pp. 1–8 (2014).
[13] 木村英明，佐藤三久，堀田義彦，今田貴之：影響の少ないインスツルメント手法と電力最適化のためのプログラム領域分割，情報処理学会論文誌コンピューティングシステム（ACS），Vol. 48, No. SIG13(ACS19), pp. 247–259 (2007).
[14] 和田康孝，稲富雄一，井上弘士，三吉郁夫，近藤正章，本多弘樹：高性能計算環境向け電力配分自動最適化のためのコンパイラ環境の構築，研究報告ハイパフォーマンスコンピューティング（HPC），Vol. 2014-HPC-145, No. 11, pp. 1–8 (2014).
[15] 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，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 (2013).