Vol.2013-ARC-207 No.31
Vol.2013-HPC-142 No.31
2013/12/17

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

和田 康孝[1,a)]　カオ タン[1,2]　近藤 正章[1,2]　本多 弘樹[1]

概要：将来の HPC システムでは，消費電力がシステムの設計や実効性能を制約する最大の要因の一つになると考えられている．しかしながら，システムのピーク消費電力が電力制約を超えないことを保証する従来の設計思想では，アプリケーションを今後の大規模システムに対してスケールさせることは難しいと考えられる．我々はこの認識のもと，システムのピーク消費電力が制約を超過することを積極的に許容し，適切に電力性能ノブを調整することで，限られた電力資源を有効に使用して高い実効性能を得る電力制約適応型システムと，その実現に必要となる電力マネージメントフレームワークの研究開発を行っている．このような電力制約適応型システムにおいて，アプリケーションの最適化を効率よく行うためには，従来のプロファイルや実行トレース，ハードウェアカウンタ等の情報に加え，消費電力に関する情報も併せて取得し，可視化してプログラマにわかりやすく提示する仕組みが求められる．本稿では，既存のパフォーマンス解析ツールの消費電力情報取得機能について，主に観測オーバヘッドの削減および可視化や各種ツール間の可搬性の観点から比較検討を行った．その結果，既存ツールと比較して，可搬性の高い Open Trace Format Version2（OTF2）形式にて低オーバヘッドで消費電力情報を記録することが可能となった．

1. はじめに

将来の HPC システムでは，消費電力がシステム設計や実効性能を制約する最大の要因の一つになると考えられている．例えば，本稿執筆時点で世界最高性能を誇る Tianhe-2 は 33 ペタフロップス超の性能を達成するために 18MW 近い消費電力を必要としており，京は 10 ペタフロップスの性能を得るために 12MW 超の消費電力を要する [1] が，地球規模の省エネ要求や現在の大型計算機センターの電力設備状況や物理的な制約を鑑みると，将来的にもこれ以上の電力供給能力を持つセンターを設置することは難しい．つまり，2020 年頃に実現されるエクサスケール級のシステムでは，現在とほぼ同じ 20〜30MW 程度の電力で現在の世界トップクラスのスーパーコンピュータの 30〜50 倍近い性能を達成することが求められる．さらに，環境負荷低減の重要性が叫ばれる中，電力供給量が不安定である再生可能エネルギー利用が大型計算機センターにも拡大し，電力供給が時々変化するような状況に対応した運用が求められることも考えられる．

また，供給電力，あるいは熱設計消費電力制約の中でハードウェア資源を投入し，システムのピーク消費電力が制約を超えないことを保証する従来の設計思想では，様々な特性を持つアプリケーションを今後の大規模システムに対してスケールさせることは難しいと考えられる．このような背景のもと，我々はシステムのピーク消費電力が制約を超過することを積極的に許容し，ハードウェアが持つ電力性能ノブを調整することで限られた電力資源を計算・記憶・通信等の各要素に適応的に配分し，実効電力を制約以下に制御しつつ高い実効性能を得る電力制約適応型システムがポストペタスケール HPC システムのあるべき姿との認識に立ち，その実現に必要となる電力マネージメントフレームワークの研究開発を実施している [2].

このような電力制約適応型システムでは，アプリケーションの特徴やシステムの運用状況等に合わせた電力制御・電力管理が電力マネージメントフレームワークの最も重要な役割の一つとなる．適切な電力制御のためには，まずアプリケーション実行時の電力消費状況を観測することが必須となる．実際に TSUBAME2.0 は各計算ノード，ラック，及び計算機室の消費電力情報を監視するシステムを備えており [3]，IBM の Blue Gene/P や Blue Gene/Q はラックの AC/DC コンバータや各ノードボード，リンクカード等消費電力を一定間隔で取得・監視する機能を備えている [4].

さらに，このような電力監視機構に加え，電力制約適用

[1] 電気通信大学大学院情報システム学研究科
[2] 独立行政法人科学技術振興機構，CREST
[a)] wada@is.uec.ac.jp

型システムに適応するようにアプリケーションの性能と消費電力を同時に最適化するには，アプリケーションの性能に関わるプロファイル情報や実行トレース情報と消費電力の情報をより細かな時間粒度で取得し，プログラマに提示する必要がある．従来から，アプリケーションの性能を解析するためにプロファイル情報や実行トレース情報を取得してプログラマに提示するツールは多く開発されてきたが，パフォーマンス情報の取得にはある程度のオーバヘッドが必要となり，アプリケーションの実行状況に大きな変化が現れ，実際とは異なった挙動となることも考えられる．電力制約適応型システムにおいて消費電力に対する制御は実時間に即して行われなくてはならないため，このように，観測時の挙動と実際の実行時の挙動が異なることは，実効電力を制約以下に保つ上で大きな問題となりうる．

このような問題に対応するための初期段階として，本稿では，HPC システム上で動作するアプリケーションに対して，プロファイルや実行トレース情報，パフォーマンスカウンタの情報等を取得するパフォーマンス解析ツールについて，パフォーマンス情報および消費電力情報の取得を行う際のオーバヘッド，および可視化の利便性を考慮して比較検討を行った．

以降，2 節ではパフォーマンス解析ツールおよびその予備評価について，3 節では消費電力情報を取得する方法と可視化を考慮した検討について，4 節では消費電力観測に関わるオーバヘッドの評価についてそれぞれ述べる．

2. パフォーマンス解析ツール

アプリケーションの性能をより引き出すためにチューニングを行う際，その特性を解析しボトルネックとなる要素や負荷バランス等の情報を得るために，パフォーマンス解析ツールが用いられる．これは，各関数の実行回数や実行時間に占める割合を示すプロファイル情報に加え，各関数の開始・終了といったイベントやパフォーマンスカウンタの情報などを時系列で逐次記録した実行トレース情報等を得ることができるものであり，HPC システムを対象にしたものでは，MPI による通信や OpenMP によるワークシェアリング，同期，通信量等についても情報を得ることが可能である．

2.1 解析ツールを用いたプロファイリング・実行トレース取得

HPC システム向けのパフォーマンス解析ツールとしては，TAU[5]，Scalasca[6]，Score-P[7] などが研究開発されている．TAU は Fortran，C，C++，Java および Python 等で記述されたアプリケーションを解析可能であり，独自のツールによって様々な可視化方法を提供する．Scalasca は C，C++および Fortran で記述されたアプリケーションを扱うことができ，IBM BlueGene や Cray XT など大規模なシステムをサポートしている．Score-P は上記 2 つを含む様々なパフォーマンス解析ツールの研究者が連携して開発に参加しており [8]，これも C，C++および Fortran で記述されたアプリケーションを扱うことができる．また，Score-P は他の様々な解析ツールとの連携を考慮し，Open Trace Format Version 2（OTF2）という共通のフォーマットを提供している．

いずれのパフォーマンス解析ツールにも共通する点として，アプリケーション内の各関数が実行時間に占める割合を示すプロファイリング情報と，関数の開始・終了や通信のタイミング等をイベントとして時系列で記録していくトレース情報を取得するために，1）対象アプリケーションのソースコードを解析し，情報取得に必要な機能を埋め込む，2）通信やワークシェアリングに関する情報を実行時に取得するため，MPI や OpenMP の機能に対してラッパーを用意して適用する，という点が挙げられる．たとえば，TAU では Program Database Toolkit（PDT）と連携することで，より詳細かつ動的に情報を取得することを可能としている [9]．

このように，パフォーマンス解析ツールでは，アプリケーションの詳細の情報を取得するためのコードを対象アプリケーション中に埋め込むため，計測にかかるオーバヘッドが少なからず発生する．これは性能はもちろんのこと，消費電力特性を調べる上で特に大きな問題になる．

2.2 解析ツールによるオーバヘッドの予備評価

パフォーマンス解析ツールによる性能への影響を調査するため，TAU および Score-P によって実行プロファイルの取得および実行トレース情報の取得を行った際のアプリケーションの実行時間の変化を評価した．評価に用いた環境を表 1 に示す．本予備評価では，16 コアを持つ 1 ノードにおいて NPB に含まれる BT，LU，SP の三種類のベンチマーク及び HPC Challenge（HPCC）ベンチマークを用いて比較を行った．アプリケーションの実行時間に応じてトレース情報の格納ファイル容量が大きくなってしまうため，評価環境上の制限から，NPB においてはクラス B の入力セットを用い，HPCC ベンチマークでは，配列サイズ N を 5120 とした．各ユーザ関数および MPI 関数の開始・終了時刻をトレース情報として取得した．なお，いずれのベンチマークも MPI による並列処理のみを適用し，3 回実行したうち最も実行時間の短いものを比較した．

図 1 にランク数 4 の場合の，図 2 にランク数 16 の場合の予備評価結果を示す．各図において，横軸は評価アプリケーションおよびプロファイル・トレース情報取得の有無，縦軸は解析ツールを使用しない場合に正規化した実行時間である．なお，図 2 の BT および HPCC において，Score-P を用いてプロファイル情報と実行トレース情報を取得する場合には実行時間が解析ツールを用いない場合に

表 1　評価環境

プロセッサ: Intel Xeon E5-2690	
コア数	8
L1 キャッシュ（コア毎）	32KB 命令，32KB データ
L2 キャッシュ（コア毎）	256KB
L3 キャッシュ（チップ毎）	20MB
マザーボード: SuperMicro MBD-X9DRL-IF-O	
CPU ソケット数	2
DIMM スロット数	8
メモリチャネル数	4
チップセット	Intel C602
LAN	Intel 82574L × 2
メモリ: DDR3-1600 TED316G1600C11DC	
容量	8GB × 8
レイテンシ	11-11-11-28
OS: CentOS 6.4（Kernel: 2.6.32-358）	
コンパイラ	GCC 4.4.7
MPI	MPICH2 1.2.1

対して 10 倍を超えたため，途中で計測を打ち切っている．

図 1 および図 2 から，BT および HPCC では計測オーバヘッドが大きく，例えば，TAU を用いて HPCC の解析を行った場合，プロファイル情報のみの場合はランク数 4 のとき約 37[%]，ランク数 16 のとき約 38[%] 増加している．実行トレース情報も合わせて取得した場合，これはさらに増加してそれぞれ約 48[%]，約 64[%] 実行時間が延びている．一方，LU および SP では実行時間の増加が比較的小さく，最大でも 10[%] 程度であることがわかる．これは，関数の呼び出し回数や MPI による通信の回数が大きく影響していると考えられる．また全体として，ランク数に応じてオーバヘッドも増大する傾向にあることが分かる．

このように，パフォーマンス解析ツールを用いてアプリケーションの詳細な情報を取得する際には大きなオーバヘッドがかかる場合があり，本来のアプリケーションの挙動とは大きく異なった情報を取得してユーザに提示してしまう．アプリケーションの性能を向上させることが目的の場合，プロセス間・スレッド間の相対的な関係や性能が保たれていれば，必要な部分の情報を選別して積算する等の処理を行うことである程度の対応が可能である．しかし，消費電力は処理の内容が変化することで大きく変わってしまう．今後，電力制約適用型システムにアプリケーションを適用し，消費電力と性能を同時に最適化することを考えると，このオーバヘッドを削減し，性能に関する情報と消費電力に関する情報が正しく得られる環境を構築する必要がある．

3. パフォーマンス解析ツールを用いた消費電力情報取得と可視化

消費電力の変化をアプリケーションの特性として取得する場合，そのアプリケーションの構造と照らし合わせる必要がある．アプリケーションからノードやその構成要素が消費する電力の情報を得る方法として，外部の電力計から情報を取得する，各部品に電力を供給するラインを直接計測するなどの方法が考えられる．しかしながら，外部の電力計では観測の間隔が 1[s] 以上と比較的長い場合が多いという問題があり，また直接計測する方法は一般的な環境で利用することが難しい．アプリケーションの特性と消費電力の関係を正しく把握するためには，細かい時間粒度の情報がユーザレベルで取得できる必要があり，本稿では，この条件を満たす，Intel のプロセッサが持つ RAPL インタフェース [10], [11] を用いることを考える．

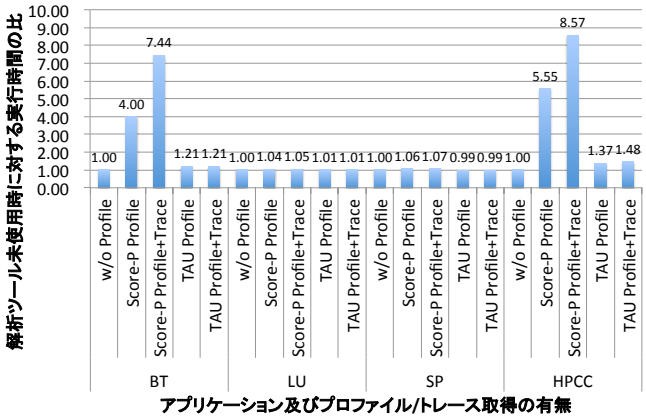

図 1　TAU および Score-P の予備評価結果（ランク数 4）

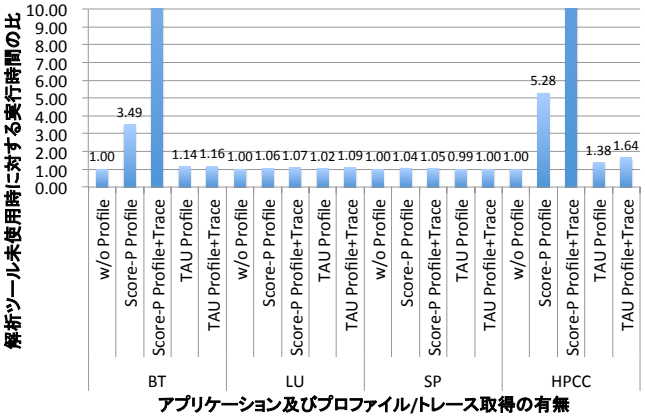

図 2　TAU および Score-P の予備評価結果（ランク数 16）

3.1 RAPL インタフェースを利用した消費電力情報の取得

RAPL（Running Average Power Limit）インタフェースは Intel 製プロセッサにおいて Sandy Bridge マイクロアーキテクチャより搭載された機能であり，このインタフェースを介してプロセッサおよび DRAM の消費電力に関する情報を取得したり，消費電力の上限を設定することができる．プロセッサは，パフォーマンスカウンタや温度などの情報を基に消費電力の見積りと記録を行い，与え

られた消費電力の上限を超えないように動的に制御を行う [10], [11]．RAPL による消費電力の情報は精度が高く，また小さい時間粒度で取得可能であるため [12]，パフォーマンス解析ツールで用いるに適していると言える．

RAPL では消費電力を観測・制御する単位が 3 種類定義されており，サーバ環境では，チップ全体（Package, PKG），チップ上のコア部分（Power Plane 0，PP0）およびメモリ（DRAM）がそれにあたる．Linux OS 上では，MSR（Model Specific Register）を介して消費電力の取得，消費電力の上限設定などの操作を上記の各ドメイン毎に適用することができる [13]．また，これらの操作は CPU ソケット毎に個別に行うことができる．

ユーザは MSR にアクセスして消費電力の情報を直接取得することも可能であるが，Performance Application Programming Interface（PAPI）の機能を用いてこれを行うことが可能である．PAPI はアプリケーションに対して各種プロセッサが備えるパフォーマンスカウンタを利用するための統一的なインタフェースを与えるもので，これを利用することによって，アプリケーションのパフォーマンスに関する様々な情報を取得することができる．本稿執筆時点における最新バージョンは 5.2.0 であり，NVIDIA CUDA GPU への対応や，RAPL インタフェースへの対応がなされている [14]．内部の実装としては，MSR にアクセスして RAPL から消費電力に関する情報を得る形をとっている．

3.2 既存ツールにおける消費電力情報のロギング

3.1 節で述べたように，RAPL インタフェースを利用することで消費電力に関する情報を取得可能であるが，アプリケーションの特性と消費電力の変化を対応づけてチューニングに用いるためには，同時に実行トレース情報と消費電力の変化を取得していく必要がある．

ここでは，既存のパフォーマンス解析ツールにおける消費電力情報取得のための機能について述べる．

3.2.1 TAU における消費電力情報取得機能

本稿執筆時点において，TAU の正式リリースバージョンは Ver.2.22.2 であるが，このリリースバージョンでは電力情報を取得する機能は取り入れられていない．ただし，入手可能なバージョンである Ver.2.22.3b4 においては，PAPI を経由して RAPL の情報を取得し，消費電力の情報を集計あるいはロギングする機能が実装されている．ここでは Ver.2.22.3b4 を対象として議論する．

TAU における消費電力情報取得は PAPI の計測機能を利用したものであり，プログラム中に計測開始・終了のための関数を追記することでその間の消費電力測定が可能となる．測定区間内はタイマとシグナルを用いて一定間隔で観測のための関数が呼び出され，PAPI の機能を介して RAPL の値を取得する．プロファイル情報のみを取得する指定をした場合には計測対象区間全体の最大・最小・平均の消費電力が記録され，実行トレース情報を取得する場合には，一定間隔毎に他の実行トレース情報とともに時系列で独自フォーマットにて記録される．ただし，現状では設定可能な計測間隔が秒単位であるため，アプリケーションの動作・特性の変動に対して計測の粒度が大きくなりすぎる恐れがある．

3.2.2 Score-P における消費電力情報取得機能

本稿執筆時点において，Score-P の最新バージョンは 1.2.1 となっており，このバージョンにおいて電力情報取得機能は実装されていない．しかしながら，PAPI を経由してパフォーマンスカウンタの値を取得する機能を持っているため，PAPI の機能を使用する際に RAPL に関連するイベントを指定することで，消費電力に関する値を取得可能と考えられる．ただし，現時点では RAPL の値を扱うことを想定した実装にはなっていない．

通常 Score-P において実行トレース情報は各ユーザ関数および MPI・OpenMP ライブラリコールの前後で取得され，PAPI に関する情報も同様に取得される．そのため，RAPL 内部の情報がアップデートされるよりも短い時間間隔で消費電力の情報を取得・ロギングしてしまい，正しい情報が得られない，あるいは過剰な計測オーバヘッドがかかる可能性が考えられる．

3.3 可搬性の考慮

3.1 および 3.2 節で述べたように，RAPL インタフェースおよびこれを利用する様々なツールを用いて，アプリケーションの消費電力情報を取得することができる．この情報に基づいて効率よくアプリケーションの性能と消費電力をチューニングするためには，取得した情報の可視化が重要である．

従来から様々な実行トレース情報の記録フォーマットが提案・利用されてきた．例えば，TAU であれば独自の Tau Trace Format が，Scalasca であれば EPILOG フォーマットが利用されてきたが，2.1 節で述べたように，Score-P の一部として Open Trace Format Version 2（OTF2）が開発され，現在では様々なツールが OTF2 をサポートしている．OTF2 では，実行トレース情報のフォーマットとともにこれを操作するための API およびライブラリが提供されており，これらを利用することで，様々な可視化ツールで利用できる実行トレース情報を作成可能である．つまり，OTF2 形式を介して様々なツールの機能を利用できる．例えば，TAU は Tau Trace Format から OTF2 に変換するツールを提供しており，Scalasca はバージョン 2.0 から Score-P の機能を利用し，OTF2 フォーマットで情報を記録することが可能となっている．

実行トレース情報の可視化ツールとしては，Vampir[15] や TAU に付属する Paraprof などがあるが，これらも上述

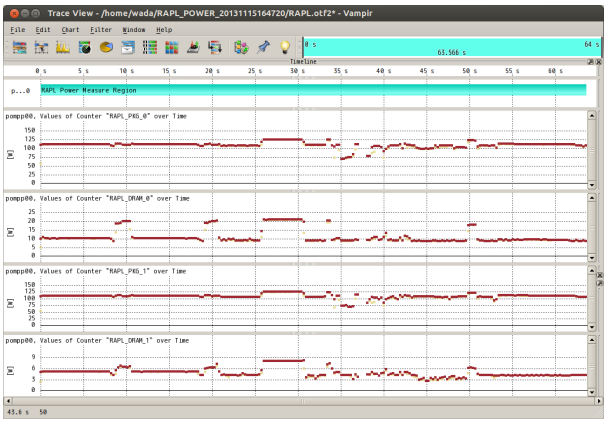

図 3 Vampir による消費電力可視化の例

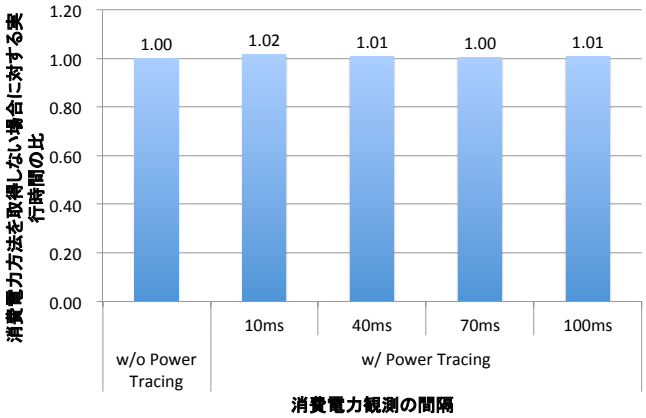

図 4 提案手法の評価（HPCC，ランク数 16））

の通り直接あるいは間接的に OTF2 のサポートが進んでいる．特に Vampir は OTF2 が標準の入力フォーマットとなっている．

3.4 可視化ツールに対する可搬性とオーバヘッドを考慮した電力ロギング

3.2 節で述べたように，TAU ではタイマとシグナルによって定期的かつプログラムの動作とは非同期に消費電力を観測することができ，プログラムの構造とは独立して観測の時間粒度を考慮できるという点では優位性がある．しかしながら，同時に取得されている他の実行トレース情報とデータの登録先アーカイブを共有しているため，オーバヘッドが増大することが考えられる．そこで，TAU 等のパフォーマンス解析ツールによる実行トレースとは個別に消費電力情報を取得・格納する方法を考える．また，PAPI を経由せずに RAPL インタフェースに直接アクセスすることで，余分な観測オーバヘッドを削減することも可能となる．

以上をふまえ本稿では，シグナルおよびタイマによって一定間隔ごとに MSR にアクセスして RAPL の情報を取得し，実行トレース情報として記録する機能をもつライブラリを作成した．このライブラリでは，3.3 節で述べたように，可視化ツールおよび解析ツール間の可搬性やデータの統合・加工のしやすさを考慮して OTF2 の API とライブラリを利用してデータを出力するものであり，これによりデータ取得後の情報統合や可視化が容易になると期待できる．

図 3 に，今回作成したライブラリを用いて取得した消費電力情報を Vampir を用いて可視化した例を示す．図 3 では，本稿で用いた評価環境（表 1）において，各ソケットのチップ全体の消費電力および各ソケットに属する DRAM の消費電力を示している．

4. オーバヘッド評価

本節では，3 節で述べた消費電力情報を取得しロギングする方法に関して，オーバヘッドを評価した結果について述べる．なお，評価環境は 2.2 節同様，表 1 の通りである．

4.1 作成したライブラリのオーバヘッド

本評価では，3.4 節で詳細を述べたライブラリ（以下，提案手法）のオーバヘッドを評価するため，HPCC ベンチマークにこのライブラリを適用し，消費電力情報を取得する間隔を 10[ms]，40[ms]，70[ms]，100[ms] と変化させたときの実行時間の変化を測定し，消費電力の観測を行わない場合と比較した．

図 4 に評価結果を示す．図 4 において，横軸は消費電力測定の有無および測定時の測定間隔，縦軸は測定を行わない場合に正規化した実行時間である．提案手法では，ほぼオーバヘッド無しで OTF2 形式の消費電力情報を得ることができる．

4.2 実行トレース取得時の評価

2.2 節における予備評価から，実行トレース情報取得のオーバヘッドが大きかった NPB の BT および HPCC ベンチマークについて，TAU および提案手法それぞれで消費電力情報の取得とロギングを行った際のアプリケーションの実行時間を測定した．ここでは，実行トレース情報の取得および消費電力情報ロギングのオーバヘッドを比較するために，下記 3 通りについて評価を行う．

- 実行トレース情報と消費電力情報のロギングを TAU の機能を用いて行った場合
- 実行トレース情報を取得せず，消費電力情報のみ提案手法を用いて取得した場合
- 実行トレース情報は TAU の機能を用いて取得し，消費電力情報を提案手法用いて取得した場合

ここで，TAU で消費電力情報を取得する間隔は設定可能な最小値である 1[s]，提案手法においては 100[ms] とした．

図 5 にランク数 4 の場合の評価結果を，図 6 にランク数 16 の場合の評価結果を示す．各図において，縦軸は解析ツールを使用しない場合に正規化した実行時間，横軸はア

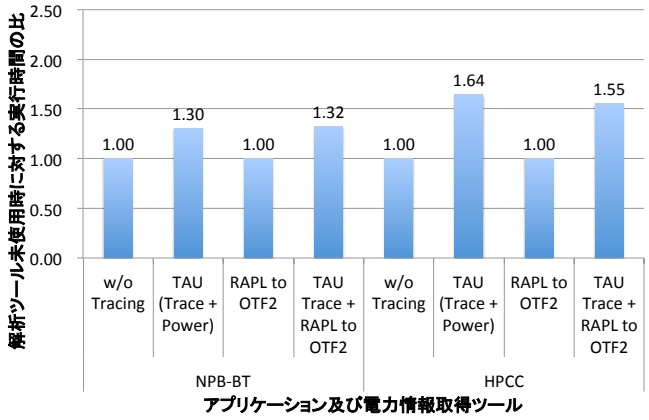

図 5 電力情報ロギング性能の評価（ランク数 4）

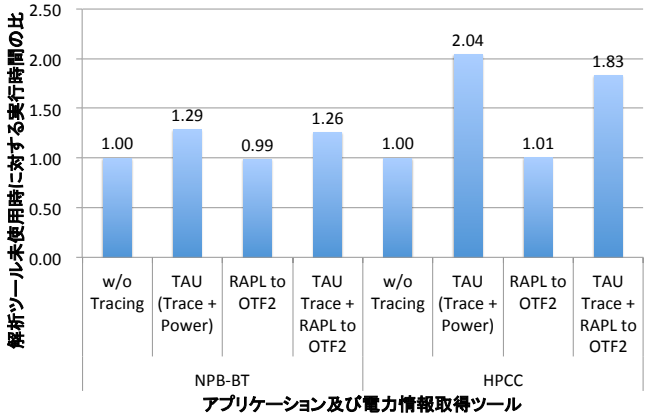

図 6 電力情報ロギング性能の評価（ランク数 16）

プリケーションおよび情報の取得方法であり，”TAU(Trace + Power)” は TAU の機能を用いて実行トレース情報と消費電力情報のロギングを行ったもの，”RAPL to OTF2” は実行トレース情報を取得せず，消費電力情報のみ提案手法で取得した場合，”TAU Trace + RAPL to OTF2” は実行トレース情報を TAU の機能を用いて取得し，消費電力ロギングを提案手法で取得した場合を示す．

図より，提案手法ではほぼオーバヘッド無く消費電力情報の取得とロギングができていることがわかる．そのため，TAU では実行トレース情報のみを取得し，本手法で消費電力情報を記録することで，消費電力の観測にかかるオーバヘッドを削減することができる．例えば，HPCC をランク数 16 で実行した場合，TAU で消費電力情報を取得した場合と比較して 10[%] 以上，ランク数 4 の場合でも 6[%] 程度，パフォーマンス解析を含めた実行時間が短縮されている．加えて，TAU よりも細かい時間粒度を設定可能である．

5. まとめ

将来的のポストペタスケールシステムでは消費電力が最大の制約になり，性能とともに消費電力を意識したアプリケーション最適化を行うことが重要であると考え，本稿では，パフォーマンス解析ツールによる消費電力情報の取得および可視化について検討と初期評価を行った．また，本稿では各解析ツールおよび可視化ツールでの利便性と観測の時間粒度，および観測オーバヘッドを考慮した消費電力のロギング方法についても検討を行った．

評価の結果，パフォーマンス解析ツールのみで実行トレース情報と消費電力情報を取得するのではなく，外部から平行して取得し別途記録していくことで，情報取得の時間粒度を考慮しつつオーバヘッドを削減可能であることが示された．また，様々な解析ツールや可視化ツールでサポートが行われている OTF2 を出力形式とすることで，利便性を確保することができている．

今後の仮題としては，より大規模なアプリケーションや HPC システムでの評価，および更なる観測オーバヘッドの削減などが挙げられる．

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」の助成を受け行われた．本研究に関して多くのご助言，またツールのサポートを頂いた九州大学大学院システム情報科学研究院の井上准教授，ならびに井上グループの皆様に感謝いたします．

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] 松岡 聡：グリーンなスパコンはエクサスケールの夢を見るか。 - TSUBAME2.0 を例にして，第 10 回 PC クラスタシンポジウム招待講演 (2010).
[4] Yoshii, K., Iskra, K., Gupta, R., Beckman, P., Vishwanath, V., Yu, C. and Coghlan, S.: Evaluating Power-Monitoring Capabilities on IBM Blue Gene/P and Blue Gene/Q, *Proceedings of 2012 IEEE International Conference on Cluster Computing*, pp. 36–44 (2012).
[5] 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).
[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] Virtual Institute - High Productivity Supercomputing (VI-HPS): http://www.vi-hps.org/.
[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 with Templates, *Proceedings of the 2000 ACM/IEEE Conference on Supercomputing*, p. 49 (2000).
[10] 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).
[11] 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).
[12] 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).
[13] Intel Corporation: Intel® 64 and IA-32 Architectures Software Developer's Manual (2013).
[14] 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).
[15] Knüpfer, A., Brunst, H., Doleschal, J., Jurenz, M., Lieber, M., Mickler, H., Müller, M. S. and Nagel, W. E.: The Vampir Performance Analysis Tool-Set, *Proceedings of the 2nd International Workshop on Parallel Tools for High Performance Computing*, pp. 139–155 (2008).