Vol.2018-HPC-163 No.3
2018/2/28

DFS/DCT 制御による電力あたり性能の実行時最適化

三吉郁夫[†1]　三輪忍[†2]　井上弘士[†3]　近藤正章[†4]

概要：エクサスケールの HPC システムでは電力あたり性能の最適化が重要課題となるため，最適化実施の敷居を下げるべく，ジョブ実行の最中に最適化を行う手法を開発した．電力あたり性能を最適化するにあたり，本手法では，実行性能の低下許容度という指標を設け，ジョブ実行の機会増加や待ち時間短縮のメリットと，ジョブ自体の実行時間増加の間のトレードオフについて，ジョブ実行者が検討する余地を与える．HPCG ベンチマーク等を用いた評価の結果，本手法の汎用性や電力あたり性能の改善効果が確認された．特に汎用性に関しては，多くの測定で改善効果が確認されただけでなく，3 世代のインテル社製 Xeon プロセッに対して，1 種類のパラメータを使い分けるだけで十分対応できることが示された．

キーワード：電力あたり性能，実行時最適化，DFS 制御，DCT 制御

Run-Time Performance-per-Watt Optimization by DFS/DCT Techniques

IKUO MIYOSHI[†1]　SHINOBU MIWA[†2]
KOJI INOUE[†3]　MASAAKI KONDO[†4]

Abstract: Since optimization of “performance-per-watt” becomes more important issue on exascale HPC systems, we propose a technique of run-time optimization during each job’s execution in order to enable its easy adoption. Specifying range of acceptable performance degradation, trade-off between increasing opportunities of job execution and slightly longer execution time can be considered by users at the time of job submission. As a result of evaluation with HPCG and other benchmark programs, its effectiveness and wide applicability were validated. As for the applicability, the optimization technique was adapted to 3 generations of Intel Xeon processors by only one parameter, as well as showing the improvement of “performance-per-watt” in many cases.

Keywords: Performance-per-watt, Run-time optimization, Dynamic Frequency Scaling, Dynamic Concurrency Throttling

1. はじめに

エクサスケールの HPC システム開発では消費電力が主要な設計制約の 1 つとなるが，そのような制約に対する対応としてハードウェア・オーバープロビジョニングというアプローチが提案されている（図 1）．ハードウェア・オーバープロビジョニングの考え方で構築されたシステムでは，システムを構成する各コンポーネント（CPU，GPU，メモリ，インターコネクト等）がすべて最大負荷時電力で動作した場合にそれらの合計が電力制約を超過してしまうため，電力性能ノブを制御することで実効電力を制約以下に抑えるための仕組みが必要となる．その際，当然のことながら，性能の最大化も求められるため，本質的には電力あたり性能の最適化が重要課題となる．

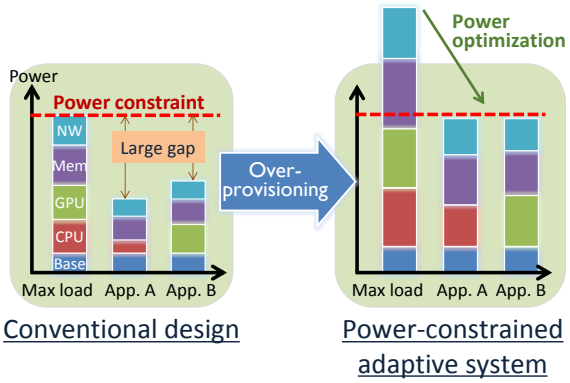

図 1　ハードウェア・オーバープロビジョニング
Figure 1　Approach of hardware-overprovisioning.

電力性能ノブとしては CPU の動作周波数を変更する方法（Dynamic Frequency Scaling, DFS）やスレッド並列処理の使用スレッド数を変更する方法（Dynamic Concurrency Throttling, DCT）などがあり，これらを適切に用いることで電力あたり性能を改善できることが知られている．例えば HPCG ベンチマークをインテル社製 Xeon プロセッサで実行する場合，CPU 動作周波数や使用スレッド数を適切に選択することによって，電力あたり性能を 4 割以上改善することが可能である（図 2 および図 3）．

†1 富士通株式会社
Fujitsu Limited
†2 電気通信大学
The University of Electro-Communications
†3 九州大学
Kyushu University
†4 東京大学
The University of Tokyo

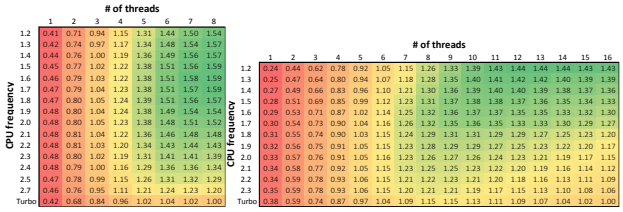

図 2　HPCG ベンチマークにおける電力あたり性能比（左図:Sandy Bridge，右図:Haswell）
Figure 2　Relative “performance-per-watt” of HPCG (left:Sandy Bridge, right:Haswell).

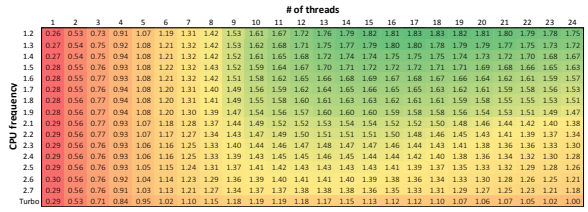

図 3　HPCG ベンチマークにおける電力あたり性能比（Skylake）
Figure 3　Relative “performance-per-watt” of HPCG (Skylake).

表 1　測定に使用した CPU
Table 1　CPUs used for measurements.

	Model	Details
Sandy Bridge	Xeon E5-2680	2.7GHz, 8 cores, TDP 130W
Haswell	Xeon E5-2698v3	2.3GHz, 16 cores, TDP 135W
Skylake	Xeon Platinum 8168	2.7GHz, 24 cores, TDP 205W

これに対し従来のジョブ実行では，実行全体を通じて 1 つの CPU 動作周波数を選択/固定し，その結果の実行時間を収集して，良い結果となった周波数を活用する程度の対処に留まっていた．この方法では，i）1 回のジョブ実行時間が長い場合に最適解が求まるまでの時間が長く掛かる，ii）ジョブ実行の内容（アプリケーションとしての計算内容）が同一でない場合の扱いが自明ではない，iii）実行させるハードウェア環境（特に CPU のアーキテクチャ世代）が異なると既存の収集結果を利用できない，という問題があり，広く実用に供されるには支障となっていた．

そこで本研究では，1 つのジョブの実行最中に電力性能ノブを最適化することで, これら問題点の解決を目指した．具体的には，アプリケーション開発者が性能確認などの目的で設定した実行区間をそれぞれ対象として，電力あたり性能の評価と電力性能ノブの制御を行い，実行性能の低下を一定範囲に抑えつつ，単一ジョブの実行最中に電力あたり性能を最適化する手法を開発した．

以下，本稿では，電力あたり性能の実行時最適化に向けた技術的課題を 2 章で述べ，その解決に向けた本研究の提案手法を 3 章で述べる．続いて 4 章では，HPCG ベンチマークや NAS Parallel Benchmarks，NICAM-DC ミニアプリに本提案手法を適用した結果を示し，電力あたり性能の改善効果や本手法の汎用性について評価する．その後 5 章で関連する既存研究について述べ，6 章でまとめる．

2. 課題

単一ジョブの実行最中に電力あたり性能を最適化するにあたり，主な技術的課題としては，i）電力あたり性能の観測，ii）最適設定の探索，の 2 つが想定される．また，これらの課題を検討するにあたり, 実行するジョブが HPC アプリケーションであるとして，検討の前提条件を設けた．

2.1 前提条件

最適化対象のジョブを HPC アプリケーション, 特に数値計算プログラムであると想定すると，それらは構造的に以下の特性を持つことが期待できる．

- 実行するプログラムは, 時間積分や反復解法のような, 繰返し計算を行う
- 実行するプログラムは, 例えば HPCG ベンチマークにおける関数 ComputeMG や ComputeSPMV のように, 計算内容の違いにもとづいて複数の実行区間に分割することができる

そこで以降では，これらの特性を前提として最適設定探索アルゴリズムを検討する．なお，これらの特性（特に 1 点目）を満たすものであれば，数値計算以外のプログラムにも本研究の内容は適用できるはずである．

2.2 最適設定の探索

技術的課題の 1 つめは，電力あたり性能が最適となる設定の探索である．

図 2 および図 3 に示したように, 同じアプリケーションを実行する場合でも，CPU のアーキテクチャ世代が異なると, 最高の電力あたり性能が得られる CPU 動作周波数および使用スレッド数は異なる．Sandy Bridge では全 CPU コアを使用し CPU 動作周波数が低め（1.7GHz），Haswell では CPU 動作周波数が最低で多めの CPU コア（16 コア中 12 コア）を使用，Skylake も CPU 動作周波数が最低だが少し多めの CPU コア（24 コア中 18 コア）を使用，が最高となる条件であった（表 2）．

表 2　HPCG ベンチマークで電力あたり性能が最高となる CPU 動作周波数および使用スレッド数
Table 2　CPU frequency and # of threads for the best “performance-per-watt” of HPCG.

	CPU frequency (GHz)	# of threads	Relative “performance -per-watt”	Relative performance
Sandy Bridge	1.7	8	1.59	0.80
Haswell	1.2	12	1.44	0.91
Skylake	1.2	18	1.83	0.94

一方，電力あたり性能すなわちジョブあたりの消費エネルギーを改善すると，一般にジョブの実行時間が延びることが知られている．例えば表 2 の例では，最高の電力あた

り性能と引き換えに, Sandy Bridge で 20%, Haswell で 9%, Skylake で 6%，の実行性能低下が見られる．

HPC アプリケーションの実行では従来，実行時間の増加はまったく望まれないものであった．しかし，本研究が対象としている，消費電力がジョブ実行の制約条件となるような HPC システムでは, 消費電力を削減することによって, ジョブ実行の機会増加や待ち時間短縮のメリットがあり，ジョブ実行者が消費電力と実行時間の間のトレードオフを前向きに検討することが期待される．

そのため，消費電力と実行時間の間のトレードオフを勘案しつつ，電力あたり性能を改善し，さらに，実行させるハードウェア環境が異なる場合でも適用可能な設定探索手法の開発が，第一の課題であると言える．

2.3 電力あたり性能の推定

電力あたり性能が最適となる設定を探索するにあたり，電力あたり性能を時間的, 空間的に精度よく観測できれば，観測値そのものを目的関数の値として使えばよい．現実に近年の CPU では, ハードウェアの消費エネルギーをソフトウェアから観測できる機能が提供されている．例えばインテル社製 Xeon プロセッサでは，Sandy Bridge 世代より RAPL インタフェースにて, CPU コアや DRAM の消費エネルギー値を取得することができる．

しかし RAPL インタフェースでは，取得できる消費エネルギー値の更新間隔が 1 ミリ秒程度であり，更新回数の過不足による誤差を 1%以下に抑えるためには，観測の時間的粒度を 100 ミリ秒以上にしなければならない. すなわち，アプリケーションの実行区間が切り替わって計算内容やその性能特性が変化する時間的粒度に対して，RAPL インタフェースで観測できる粒度は粗すぎる, という問題がある．

また，RAPL インタフェースで取得できる消費エネルギー値は CPU 毎の値であるが, アプリケーションレベルの視点や粒度で最適化するには MPI プロセスを単位として観測できることが望ましい．すなわち，一般的に 1 つの CPU に複数の MPI プロセスを割り当ててアプリケーションを実行することを考慮すると，RAPL インタフェースは空間的粒度の面でも使いにくい．

そのため，仮に RAPL インタフェースが利用できるとしても，時間的および空間的粒度の面で適切に補正などの処置を行うことにより，アプリケーションレベルの視点や粒度で最適化するのに適した目的関数（電力あたり性能の推定値）を実現することが，第二の課題である．

3. 提案手法

2 章で述べた課題に対して，本研究で用いた解決手法を以下で述べる．なお，解決手法の検討にあたり，消費電力と実行時間の間のトレードオフについて，ジョブ実行者の許容性について以下の仮定を設けた．

3.1 仮定

2.2 節でも触れたように，消費電力がジョブ実行の制約条件となるような HPC システムでは, 消費電力を削減することによって，ジョブ実行者に実行機会増加や待ち時間短縮のメリットがある．

また，現実のジョブ実行においては，IO 性能のブレやその他の外的環境要因（ハードウェアの冷却状況なども含む）によって実行時間が多少変動することは一般的であり，ジョブ実行者はそのことを理解して，実行時間の見積もりに 1 割程度のマージンを設けると聞いている．

そこで，ジョブ実行の機会増加や待ち時間短縮のメリットを認識することによって，例えば 1 割程度の実行性能低下であれば，ジョブ実行者は性能低下を許容可能である，と以降の議論では仮定する．

この仮定により，例えば Skylake における HPCG ベンチマークの実行では，図 4 中の赤枠内部が，ジョブ実行者にとって許容可能な CPU 動作周波数および使用スレッド数の範囲となる．

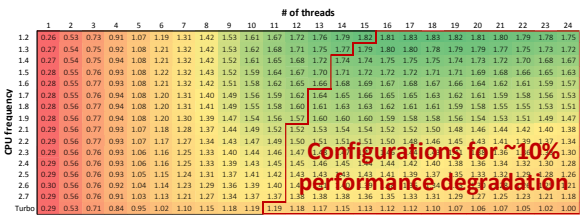

図 4 HPCG ベンチマークにおいて許容可能な実行性能低下の範囲 （Skylake）
Figure 4 Range of acceptable performance degradation on HPCG (Skylake).

3.2 消費エネルギーの推定

2.3 節で述べた，電力あたり性能の推定の課題に対しては，CPU の PMU（Performance Monitoring Unit）機能によって得られるカウンタ値から推定するアプローチを採った．

電力あたり性能すなわち FLOPS-per-watt は，その定義から，消費エネルギーあたりの浮動小数点演算数と等価である．ここで，アプリケーションの各実行区間を 1 回通過する際の浮動小数点演算数が一定であると仮定すると，実行区間を 1 回通過する間の消費エネルギーの逆数によって，その実行における電力あたり性能を相対値として推定することができる．

そこで本研究では，アプリケーションの各実行区間を 1 回通過する間の消費エネルギーを精度良く推定することを目標として，以下の推定式を用いることとした．なお，この推定消費エネルギーは，CPU 1 つとそこに接続されたメモリの分である．

推定消費エネルギー
＝CPU クロックサイクル数
×（1＋（使用スレッド数－1）
×使用スレッド数補正係数
×CPU あたり MPI プロセス数）

ここで,「使用スレッド数」は当該実行区間を通過した際の MPI プロセスあたりの使用スレッド数,「使用スレッド数補正係数」は CPU の製品モデル毎に固有の補正係数,「CPU あたり MPI プロセス数」は当該ジョブ実行において 1 つの CPU に割り当てた MPI プロセスの数，をそれぞれ表す．また,「使用スレッド数補正係数」については，すべての CPU 動作周波数および使用スレッド数で STREAM ベンチマークを実行した結果から，消費エネルギー実測値に対する CPU クロックサイクル数および使用スレッド数の回帰分析を行い，表 3 のとおり，CPU の製品モデル毎の補正係数を求めた．なお，回帰分析の対象とした消費エネルギー実測値は, CPU 1 つとそこに接続されたメモリについて RAPL インタフェースを用いて計測した結果である．

表 3　消費エネルギー推定のための使用スレッド数補正係数
Table 3　Correction factors for # of threads to estimate energy consumption.

	Correction factor
Sandy Bridge	0.13
Haswell	0.60
Skylake	0.65

本推定式の構成は，後述する最適設定探索アルゴリズムにおける，RAPL インタフェースを用いて消費エネルギーを実測した場合の相対電力効率の振る舞いを参考に，吟味し選択した結果である（図 5 参照）．すなわち，回帰分析によるパラメータ決定の作業コストを低減するべく，消費エネルギーを絶対値で推定することに固執せず，電力あたり性能の相対的な良し悪しが判定できることを要件とした．また，回帰分析に用いるデータ自体も単純化するべく，敢えて STREAM ベンチマークの結果を用いることとした．このようなアプローチで十分な推定精度が得られるかどうかについては，4.2 節で評価する．

3.3 推定消費エネルギーにもとづく最適設定探索

2.2 節で述べた，電力あたり性能が最適となる設定を探索する課題に対しては，前節の推定消費エネルギーをもとに最適な CPU 動作周波数および使用スレッド数を順次探索するアプローチを採った．

本研究が扱う電力あたり性能の最適化問題では，実行性能の低下について許容範囲の制約を課すため，最高性能が期待される条件，すなわち CPU 動作周波数が最高かつ全 CPU コアを使用，が自ずと初期条件となる．また，一般的な HPC アプリケーションでは CPU あたりの MPI プロセス数が 1 であることは珍しい点を考慮すると，CPU 動作周波数よりも使用スレッド数（CPU あたり MPI プロセス数が乗じられる）の変化による消費エネルギーの変動幅がより大きいと期待されるため，使用スレッド数に関する探索を優先する．ただし，インテル社製 Xeon プロセッサには turbo モードがあり, 消費電力や発熱の状況によって CPU 動作周波数が変動するため, 当該モード下での探索は独立に扱う．なお，設定探索の終盤，すなわち実行性能の低下が許容範囲を超えつつある状況下では，CPU 動作周波数と使用スレッド数を適宜切り替えながら許容範囲内の最適設定を探索する．

より具体的に，最適化の手順を示すと以下のとおりである．この手順は，アプリケーションの各実行区間に対してそれぞれ独立に適用される．

ステップ 1: 所定の繰返し数だけ実行し，その際の実行時間および推定消費エネルギーを，最適化の基準値として測定する．

ステップ 2: turbo モードにおいて，使用スレッド数を順次減らす方向で，実行時間の増加が許容範囲であり，かつ相対電力効率が最大となるスレッド数を探索する．

ステップ 3: 非 turbo モードで最高の CPU 動作周波数にて，ステップ 2 と同様に，相対電力効率が最大となる使用スレッド数を探索する．

ステップ 4: ステップ 2 および 3 で相対電力効率が最大となった使用スレッド数にて，CPU 動作周波数を順次低下させる方向で，実行時間の増加が許容範囲であり，かつ相対電力効率が最大となる周波数を探索する．

ステップ 5: ステップ 4 の探索において実行時間の増加が許容範囲を超えた場合，使用スレッド数を 1 つ増やした上で，ステップ 4 と同様の探索を続ける．

なお，ここでも前節と同様に，アプリケーションの各実行区間を 1 回通過する際の浮動小数点演算数は一定であると仮定し，実行性能の低下幅は実行時間の逆数から分かるものとした．また，同様の仮定から,「相対電力効率」はそれぞれの「推定消費エネルギー」に対するステップ 1 の「推定消費エネルギー」の比とし,「推定消費エネルギー」を最小にする設定が「相対電力効率」を最大化するものとした．

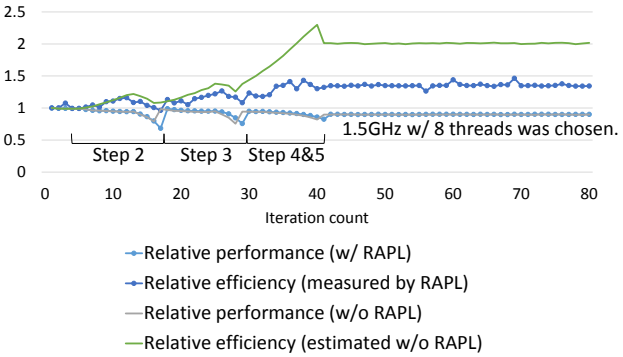

図 5　HPCG ベンチマーク ComputeSPMV 区間における最適設定探索の振る舞い　（Haswell）
Figure 5　Optimization behavior for ComputeSPMV region of HPCG (Haswell).

HPCG ベンチマークの ComputeSPMV 区間に対して上記アルゴリズムを適用した例を図 5 に示す．RAPL インタフェースを用いて消費エネルギーを実測した場合と，前節で述べた手法により消費エネルギーを推定した場合では，振れ幅は異なるものの，相対電力効率の振る舞いは十分似ており，結果として選択された CPU 動作周波数や使用スレッド数が両者で一致したことが分かる．

4. 評価

3 章で提案した手法について有効性や汎用性を確認するために，当該アルゴリズムを実装した実行時最適化ライブラリを作成し，表 4 のベンチマークプログラムを用いて評価を行った．なお，各実行区間を識別するために行ったソースコード変更の例を図 6 に示す．また，測定に使用した環境は表 5 のとおりである．

表 4　評価に使用したプログラム
Table 4　Programs used for evaluations.

	Region definition	Comments
HPCG	According to timing reports in YAML output	Tuned by Intel based on V3.0
NAS Parallel Benchmarks	According to “timer_start/ stop” calls	Only iterative kernels as class C in V3.3.1 by OpenMP
NICAM-DC-MINI	According to “DEBUG_ rapstart/rapend” calls	V1.0.0 with gl05rl00 z40pe10 dataset

```
int CG(...) {
  ...
  REGION_BEGIN(“ComputeMG”);
  ComputeMG(...);
  REGION_END(“ComputeMG”);
  ...
  REGION_BEGIN(“ComputeSPMV”);
  double localPAp = ComputeSPMV(...);
  REGION_END(“ComputeSPMV”);
  ...
}
```

図 6　実行区間識別のためのソースコード変更例
Figure 6　Example of source code modification for distinguishing regions.

表 5　測定環境
Table 5　Measurement environments.

	Sandy Bridge	Haswell	Skylake
CPU model	Xeon E5-2680	Xeon E5-2698v3	Xeon Platinum 8168
# of CPUs	2		
Turbo Boost	Enabled		
Hyper-Threading	Disabled		
OS	RHEL 6.3	RHEL 6.5	RHEL 7.4
Compilers & MPI library	Intel Parallel Studio XE 2017 Update 4		
PAPI library	Version 5.6.0 with RAPL component		
Used SIMD instructions	AVX	AVX2	AVX-512

4.1 HPCG ベンチマークにおける効果

まず，本提案手法の有効性を確認するために，本手法を HPCG ベンチマークに適用して測定を行った．使用したソースコードは，Intel Math Kernel Library 2017 Update 3 に付属の最適化されたコード/ライブラリを元に，実行区間識別のためのコード変更を行ったものである．ただしその結果，本手法を適用し実行最中に使用スレッド数を変更すると実行が終了しなくなる等の不都合があったため，ComputeMG および ComputeSPMV 区間については使用スレッド数を最大数に固定して比較を行った．また，通信による他プロセスからの性能影響を排除して評価するべく，MPI プロセス数は 1 とし，1 つの CPU の全コアを使用して実行した

実行性能低下の許容範囲を 10%に設定し，Sandy Bridge，Haswell および Skylake の 3 世代の CPU で測定した結果を図 7 に示す（「Optimized w/o RAPL」と表記）．いずれの CPU でも電力あたり性能が向上し，特に Skylake では 46%の改善が見られた．

また，探索結果の妥当性を比較確認するために，本手法を用いず，フェアな比較のために使用スレッド数を最大数に固定した条件で，すべての CPU 動作周波数についてそれぞれ周波数固定した状態で実行する測定も行った（これらは図 2～図 3 の測定の一部である）．それらの結果から性能低下 10%以内で最善の電力あたり性能となった CPU 動作周波数を探すと，Sandy Bridge では 2.1GHz，Haswell と Skylake では 1.2GHz であり，本手法の探索結果とほぼ合致した．なお，これらの周波数固定による電力あたり性能の改善幅は，Sandy Bridge で 48%，Haswell で 41%，Skylake で 76%となり，本手法を用いた場合の結果を上回ったが，この違いは本手法が実行最中に CPU 動作周波数を細かく切り替えたことによる影響と考えられる．

なお，図 7 には「Optimization w/ RAPL」として，3.2 節で述べた手法で消費エネルギーを推定するのではなく，RAPL インタフェースを用いて実測した場合の結果も示した．その結果は，消費エネルギーを推定した場合の電力あたり性能と同等であり，推定消費エネルギーの精度に問題は無かったと言える．

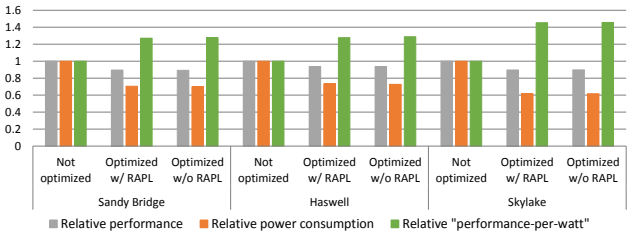

図 7　HPCG ベンチマークに対する実行時最適化結果
Figure 7　Results of run-time optimization for HPCG.

4.2 汎用性および消費エネルギー推定の検証

次に，本提案手法の汎用性や消費エネルギーの推定精度を検証するために，本手法を NAS Parallel Benchmarks に適用して測定を行った．ソースコードは OpenMP 版を用い，1 つの CPU の全コアを使用して実行した．ただし，最適化

対象の粒度を上げることで検証結果を分かりやすくするべく，各ベンチマークの性能評価区間に直接対応するそれぞれ 1 つの実行区間のみを最適化の対象とした．なお，2.1 節で述べた「繰返し計算」の特性を満たさないため，DC および EP ベンチマークは除外した．また，使用スレッド数を陽に意識して動作する IS ベンチマークも，評価対象から除外した．

実行性能低下の許容範囲を 10%に設定し，Sandy Bridge，Haswell および Skylake の 3 世代の CPU で測定した結果を図 8〜図 14 に示す．図中 Brute-force search の結果は，本手法を用いず，すべての CPU 動作周波数および使用スレッド数について，それぞれ実行条件を固定して得られた結果の中から，性能低下 10%以内で最善の電力あたり性能となったケースのものである．そのためこの値は，多くの実行コストと引き換えに明らかになった，限界性能に相当するものと言える．

まず，手法の汎用性の観点からこれらの測定結果を確認すると，Haswell および Skylake 上の FT と，Sandy Bridge 上の UA において，本手法は電力あたり性能をほとんど改善できておらず，Brute-force search の結果と異なる．ただし Haswell および Skylake 上の FT については，最適化対象の実行区間を 20 回しか通過しなかったために最適設定の探索が終了していないことが原因であり，本手法の不備が本質的に表れたわけではない．一方，Sandy Bridge 上の UA については，最適化対象の実行区間の 1 回あたり実行時間が，本手法を適用しない状態でも平均値に対して±5%以上変動するため，3.3 節で設けた「実行区間を 1 回通過する際の浮動小数点演算数は一定」の仮定が崩れている疑いがあり，その影響で実行性能の低下幅を誤って判定した可能性が考えられる．そのため，本手法は現状のままでも十分汎用的と言えるものの，上記仮定を見直し，例えば CPU の PMU 機能を用いて浮動小数点演算数を計測し利用することを検討する余地がある（一部の Xeon プロセッサではこの値を正しく計測できないため，本研究では当初，この方向を断念した）．

一方，消費エネルギーの推定精度の観点からこれらの測定結果を確認すると，Skylake 上の LU のみにおいて，RAPL インタフェースを用いて実測した場合（「Optimized w/ RAPL」と表記）と，3.2 節の式で推定した場合（「Optimized w/o RAPL」と表記）で，最適化結果が異なる．このとき，実測では 2.4GHz の 24 スレッド実行，推定では 2.7GHz の 23 スレッド実行が選択されており，消費エネルギーの推定に誤差があった可能性が高い．しかし，その他のケースにおいては，実測と推定で最適化結果は良く一致しており，3.2 節の推定式は簡素であるにも拘らず，十分精度があると言える．

なお，電力あたり性能の改善幅の観点からこれらの測定結果を確認すると，十分効果は見られるものの，Skylake 上の CG，各 CPU 上の MG，Sandy Bridge および Haswell 上の SP では，本手法の結果が Brute-force search の結果に達していない．消費エネルギーを実測した場合でも同様の差があることから，推定消費エネルギーの精度は原因でないと考えられるが，3.3 節で述べた最適設定探索アルゴリズムに問題があるのか，それとも実行時最適化によるアプローチの限界であるかは，精査の余地がある．

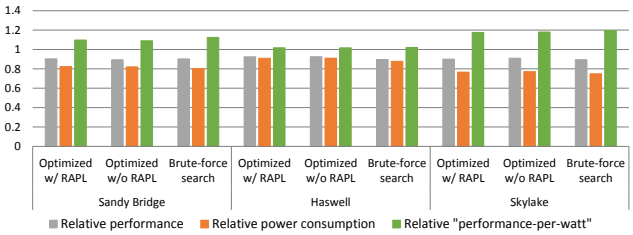

図 8　NPB BT ベンチマークに対する実行時最適化結果
Figure 8　Results of run-time optimization for NPB BT.

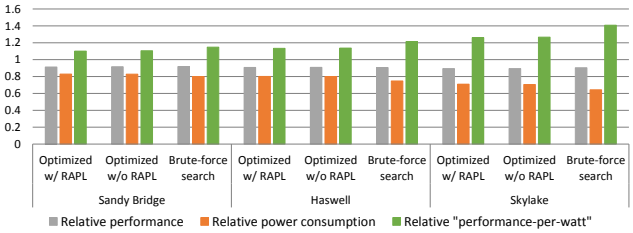

図 9　NPB CG ベンチマークに対する実行時最適化結果
Figure 9　Results of run-time optimization for NPB CG.

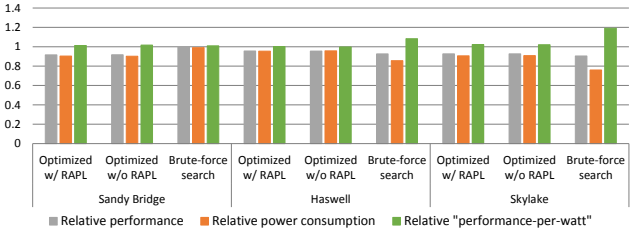

図 10　NPB FT ベンチマークに対する実行時最適化結果
Figure 10　Results of run-time optimization for NPB FT.

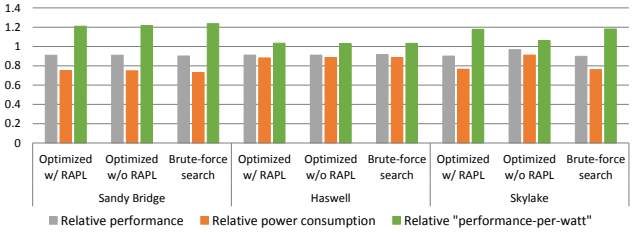

図 11　NPB LU ベンチマークに対する実行時最適化結果
Figure 11　Results of run-time optimization for NPB LU.

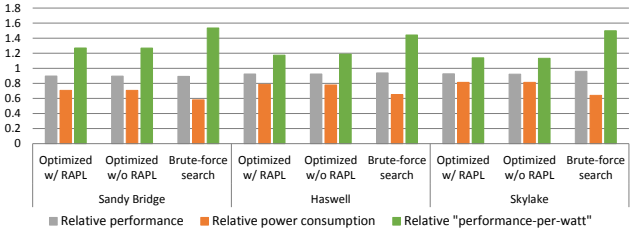

図 12　NPB MG ベンチマークに対する実行時最適化結果
Figure 12　Results of run-time optimization for NPB MG.

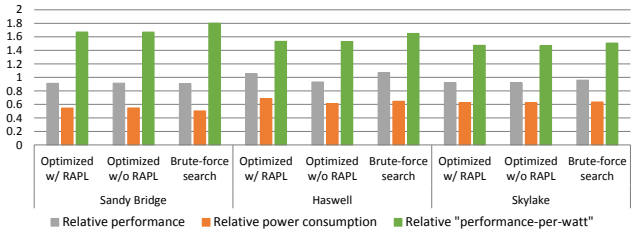

図 13　NPB SP ベンチマークに対する実行時最適化結果
Figure 13　Results of run-time optimization for NPB SP.

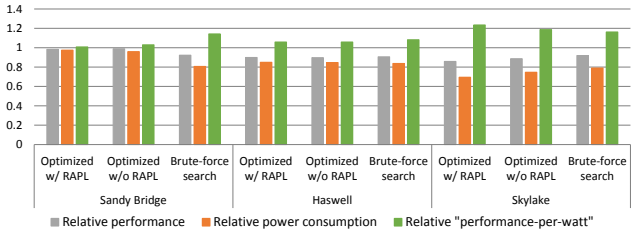

図 14　NPB UA ベンチマークに対する実行時最適化結果
Figure 14　Results of run-time optimization for NPB UA.

4.3 実行性能低下許容度の影響

続いて，本提案手法において実行性能の低下許容度が最適化結果に与える影響を確認するために，再度 HPCG ベンチマークを用いて測定を行った．測定の条件は 4.1 節と同様だが，性能低下の許容範囲を 10%，5%，3%，1%の 4 通りに設定した結果を図 15 に示す．

この測定結果を見ると，性能低下の許容度と電力あたり性能の改善幅の間には相関があり，前者を大きく許容すると後者の改善幅も拡大する傾向にあると言える．ただし，設定した許容範囲と実際の性能低下の関係を見ると，Sandy Bridge では上手く制御できていると言える一方，Haswell や Skylake の結果は十分とは言えない．その原因はまだ分析できていないが，この問題が解決すれば，消費電力と実行時間の間のトレードオフに関して，ジョブ実行者の検討の自由度が増えるため，解決が期待される課題であると言える．

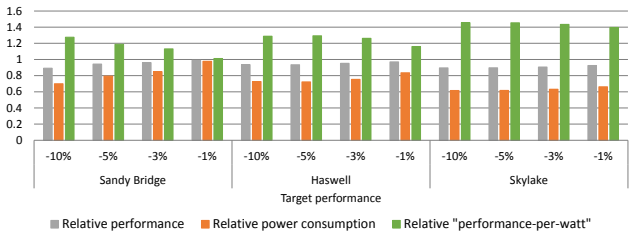

図 15　最適化結果に対する実行性能低下許容度の影響（HPCG ベンチマーク）
Figure 15　Optimization results depending on acceptable performance degradation (HPCG).

4.4 NICAM-DC ミニアプリによる評価

最後に，より実用に近いアプリケーションで有効性を確認するために，本手法を NICAM-DC ミニアプリに適用して測定を行った．このプログラムは，ポスト京の開発に向けて整備されたミニアプリの 1 つであり，全球雲解像度モデル NICAM の力学コアにもとづくものである．なお今回の測定では，Sandy Bridge，Haswell，Skylake それぞれで 2 つの CPU を搭載した 1 ノードの環境を使用するべく，10 プロセスで実行可能な小規模な入力データを用いた．

実行性能低下の許容範囲を 10%に設定し，Skylake で実行した際に計測された，CPU あたりの消費電力（メモリ分も含む）の時間的推移を図 16 に示す．本手法を適用することによって，平均消費電力が 169W から 123W に削減されている．なお，本手法を適用した場合に消費電力が細かく変動しているのは，主に 5 つの実行区間が最適化の対象と識別され，異なる CPU 動作周波数および使用スレッド数が選択された影響と理解できる．

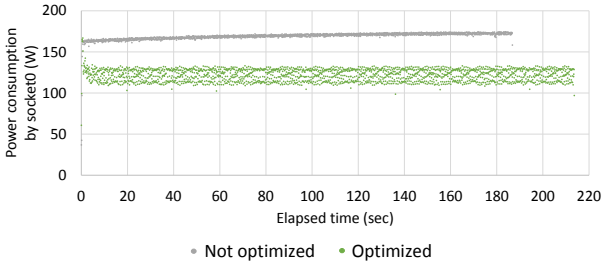

図 16　提案手法適用時の消費電力の時間的推移（NICAM-DC ミニアプリ）
Figure 16　Difference of power consumption by the proposed algorithm (NICAM-DC-MINI).

同様の測定を Sandy Bridge と Haswell についても行い，纏めた結果を図 17 に示す（「Multiple regions」と表記）．なお比較のために，最適化の対象を Atmos 区間のみに限定した場合の結果（「Single region」と表記）も記載した．この測定結果を見ると，最適化対象の実行区間を限定したほうが消費電力が少なく，結果として電力あたり性能も良いことが分かる．

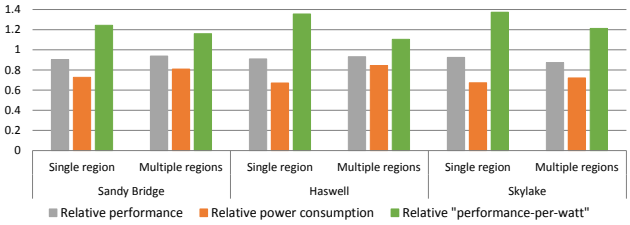

図 17　NICAM-DC ミニアプリに対する実行時最適化結果
Figure 17　Results of run-time optimization for NICAM-DC-MINI.

そこで改めて，Skylake において Atmos 区間のみを最適化した場合の消費電力について，時間的推移を図示すると図 18 のとおりであった．この結果では，最適化した場合の消費電力が 114W 程度で一定となり，複数の実行区間をそれぞれ最適化した場合（図 16）の消費電力よりも少なく，それでいて実行性能の低下幅も小さい．すなわち，最適化対象の実行区間を細かく区別し過ぎると，消費電力の面でも実行性能の面でも，逆効果になる場合があることが分かった．この点は，本手法による電力あたり性能の改善精度

や効果を高め，実行時最適化の適用を推し進めるために，引き続き重要な課題であると推測される．

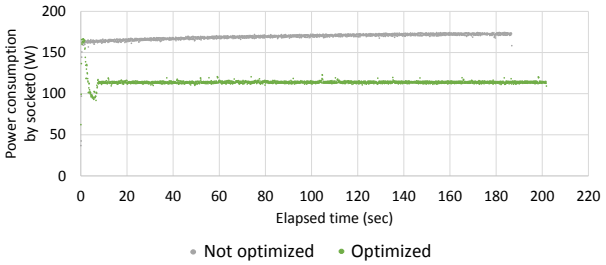

図 18　Atmos 区間のみを最適化した場合の消費電力（NICAM-DC ミニアプリ）
Figure 18　Power consumption by optimizing only “Atmos” region (NICAM-DC-MINI).

5. 関連研究

ドレスデン工科大学他によって進められている READEX プロジェクト[2]では，エネルギー効率を改善するための自動チューニング手法を提案している．ただし本研究とは異なり，プロダクションランに先立って，代表的データを用いたアプリケーションの事前実行を必要とする．そのため，様々なアプリケーションが実行されるような状況下では，この手法は適用の敷居が高いと考えられる．

ノースカロライナ大学の Bhalachandra らは，MPI 集団通信で挟まれた演算区間を対象として，演算時間の長短に応じて CPU 動作周波数等を調整し，消費電力を削減する実行時ライブラリを開発した[3]．しかしこのアプローチは，負荷不均衡がある場合に消費電力を削減できるものの，そうでない場合の効果は少ないと考えられる．

インテル社による GEOPM[4]は，エネルギー管理ソリューションのための研究用 OSS フレームワークである．その応用例の 1 つである power-balancing プラグインは，ジョブ実行のクリティカルパス上にあるノードにて，RAPL インタフェースによる電力制限を緩和し，ジョブの実行時間を短縮する．しかしこのアプローチは，実行時間は削減できるものの，消費電力は改善せず，電力あたり性能の向上に限界があると考えられる．

6. まとめ

エクサスケールの HPC システムでは電力あたり性能の最適化が重要課題となるため，最適化実施の敷居を下げるべく，ジョブ実行の最中に最適化を行う手法を本研究では開発した．電力あたり性能を最適化するにあたり，本手法では，実行性能の低下許容度という指標を設け，ジョブ実行の機会増加や待ち時間短縮のメリットと，ジョブ自体の実行時間増加の間のトレードオフについて，ジョブ実行者が検討する余地を与える．

HPCG ベンチマークや NAS Parallel Benchmarks，NICAM-DC ミニアプリを用いた評価の結果，本手法の汎用性や電力あたり性能の改善効果が確認された．特に汎用性に関しては，多くの測定で改善効果が確認されただけでなく，Sandy Bridge，Haswell および Skylake の 3 世代の CPU に対して，使用スレッド数補正係数を使い分けるだけで十分対応できることが示された．

一方で，評価の結果として，以下の点が引き続き課題であると特定された．

- 「実行区間を 1 回通過する際の浮動小数点演算数は一定」の仮定が崩れ，実行性能の低下幅を誤って判定する可能性がある．
- Haswell や Skylake では，実行性能低下の許容度設定値と実際の性能低下幅が合致しない場合がある．
- 最適化対象の実行区間を細かく区別し過ぎると，消費電力の面でも実行性能の面でも，逆効果になる場合がある．

特に 3 点目は，本手法による電力あたり性能の改善精度や改善効果を高め，実行時最適化の適用を推し進めるために，重要な課題であると思われる．

謝辞　本研究は，JST CREST「ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出」の研究課題「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」の一部として行われた．

参考文献

[1] Curtis-Maury, Matthew, et al. "Prediction models for multi-dimensional power-performance optimization on many cores." Proceedings of the 17th international conference on Parallel architectures and compilation techniques. ACM, 2008.
[2] Oleynik, Yury, et al. “Run-time exploitation of application dynamism for energy-efficient exascale computing (READEX).” Computational Science and Engineering (CSE), 2015 IEEE 18th International Conference on. IEEE, 2015.
[3] Bhalachandra, Sridutt, et al. "An Adaptive Core-specific Runtime for Energy Efficiency." Parallel and Distributed Processing Symposium (IPDPS), 2017 IEEE International. IEEE, 2017.
[4] Eastep, Jonathan, et al. “Global Extensible Open Power Manager: A Vehicle for HPC Community Collaboration on Co-Designed Energy Management Solutions.” International Supercomputing Conference. Springer, Cham, 2017.