Vol.2013-ARC-207 No.30
Vol.2013-HPC-142 No.30
2013/12/17

# 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～

稲富雄一[†1,2]　吉田匡兵[†1]　深沢圭一郎[†1,2]　上田将嗣[†1,2]　青柳睦[†1]
井上弘士[†1,2]

利用可能な電力バジェットを最重要資源とする電力指向型次世代スーパーコンピュータでは，ある電力バジェット内でアプリケーションプログラムを実行すると考えられる．そのような電力制約下で電力配分を調節することで性能を最適化する手法が必要となる．そこで，今回，量子化学計算のカーネルプログラムを用いて，電力制約下における性能最適化をインテル Xeon プロセッサに搭載されている電力制御インターフェイスを用いて行った．その結果，プログラム内の関数毎の電力バジェット配分を行うことで，数%～10 数%の性能向上を得ることができた．

# Performance Tuning of HPC Application Programs for Power-Budget Oriented Next-Generation Supercomputer — Case of A Quantum Chemical Calculation —

YUICHI INADOMI[†1,2]　KYOHEI YOSHIDA[†1]
KEI-ICHIRO FUKAZAWA[†1,2]　MASATSUGU UEDA[†1,2]　MUTSUMI AOYAGI[†1]
KOJI INOUE[†1,2]

An application program will be run under a power limit on the power budget-oriented next-generation supercomputer which is assumed an available power budget is the most important resources. Under such a power limitation, the technique to optimize a performance by regulating the power distribution is desired. Therefore, we carried out the performance optimization for the kernel program of a quantum chemical calculation under the power limitation by control the power distribution to the CPU and DRAM using the Running Average Power Limit interface equipped on the Intel Xeon processor. As a result, we obtained the ~10% performance enhancement by changing the power budget distribution for every function in the kernel program.

## 1. はじめに

近年の高性能大型計算機（スパコン）の性能向上は，トランジスタ集積度の向上に伴う計算ノード性能の向上と，計算ノード数の増加によるシステム規模の増大によって支えられてきた．しかしながら，2011 年に世界最高性能を達成した京コンピュータで約 13MW，2013 年現在での世界最速スパコンである Tianhe-2 では約 18MW という大きな電力を消費している[1]．米国 DARPA ( Defense Advanced Research Projects Agency )の報告[2]では現実的に供給可能な電力は 20MW とされており，今後のスパコン開発においては消費電力の増大が深刻な問題となっている．一方で，スパコン上で動作させるアプリケーションプログラムのシステムへの要求は多様化しており，演算性能で 100 倍，メモリ帯域で 1000 倍，メモリ容量で 1000 倍もの差が要求仕様に生じる，という報告もある[3]．したがって，次世代のエクサスケール・スパコンを実現するためには，アプリケーション特性に基づく様々な要求仕様に対応できる柔軟性を有し，その上で，与えられた電力バジェットを効率的に利用してアプリケーションの実行性能を最大化するための技術開発が不可欠となる．

このような課題に対し，現在我々は，電力バジェットこそが考慮すべき最重要資源であるとの考えに基づき，電力制約適応型システムに関する研究開発を進めている[4]．従来のスパコン設計法では，システム全体が稼働した際に消費される理論ピーク消費電力(定格消費電力，いわゆる熱設計電力) が制約を越えない範囲でハードウェア資源を投入する．これに対し，電力制約適応型システムでは，最大消費電力が制約を超過することを前提に大量のハードウェアを設置する．そして，アプリケーション特性に応じてシステム稼働時の「実効消費電力(実際に消費する電力)」が電力制約値を超えないよう制御する．

このように利用可能な電力バジェットに基づいた運用を行う「電力指向型」のスパコンでは，アプリケーションプログラムが実行時に利用可能な電力バジェットに何らかの制約が加えられていることが一般的である．そのようなシステムで効率的に動作させるためのアプリケーションプログラムの最適化は，従来の性能最適化とは大きく異なると考えられる．従来の性能最適化では，利用可能な計算機資源（CPU，メモリ，ネットワークなど）が与えられていて，

* †1 九州大学
Kyushu University
†2 科学技術振興機構/CREST
JST/CREST

Vol.2013-ARC-207 No.30
Vol.2013-HPC-142 No.30
2013/12/17

それを効率的に利用して計算時間を最短にするような最適化を行う．一方で，電力指向型スパコンを利用するアプリケーションの最適化では，利用可能な電力バジェットが与えられていて，アプリケーションプログラムの特性に合わせて，その電力バジェットを各資源にうまく配分することで，実行時間を最短にする，という最適化が可能になる．

このような電力バジェットを考慮した性能最適化については，これまで行われたことがなく，どのような場合にどのような最適化を行うか，という知見がほとんどない．そこで我々は，電力指向型スパコンでの動作を目指したアプリケーションプログラムの最適化をどのように行うかの指針を得るべく，既存のシステムを用いて電力制約下における性能最適化を試みた．本稿では，ターゲットアプリケーションとして，並列量子化学計算プログラム OpenFMO[5]のカーネルプログラム（SCF プログラム）を例にとり，インテル社製プロセッサに搭載されている電力制御インターフェイスである Running Average Power Limit (RAPL)[6]を用いて電力特性の測定，ならびに，CPU とメモリ（DRAM）に対する電力バジェット配分の調整による性能最適化を行ったので，その結果を報告する．

## 2. 評価環境

### 2.1 プラットフォーム

本実験を行った環境を（**表 1**）に示す．マザーボードに 2 つの CPU ソケットが備えられており，それぞれのソケットに Intel 社製 CPU の Xeon E5-2620（6 コア, 2.0GHz）が搭載されている．電力バジェット配分の実現には，Intel 社製 CPU に搭載されている RAPL を利用した．RAPL は，消費電力を管理するためのインターフェイスであり，CPU，DRAM の消費エネルギー計測や，消費電力制約値の設定などを行うことが可能である．今回用いたシステムには表 1 にあるように 6 コアのプロセッサが 2 ソケット搭載されているが，各種実験における SCF プログラムの実行では，パッケージ ID=0 の 1 ソケット（6 コア）のみを用いた．また，NUMA コントロールを利用することにより，計算を行うプロセッサに直接繋がれた DRAM をプログラムで使用するようにした．

図 1　実験環境

Table 1　Experimental environment

| | |
|---|---|
| # of node | 1 |
| Motherboard | SuperMICRO X9DRL-iF C602chipset |
| CPU | Intel Xeon E5-2620@2.00GHz<br>6 cores×2 sockets |
| Memory | 16[GB]×8（128GB） |
| OS | CentOS 6.4 64bit |
| Compiler | Intel Compiler Version 13.1.3 (icc, ifort) |

### 2.2 消費電力情報取得，消費電力制約値の設定，ならびに，性能情報取得

RAPL を用いた消費エネルギーや CPU や DRAM の熱設計電力（TDP）などのパラメタの取得，ならびに，CPU や DRAM に対する電力制約値の設定は，プロセッサの Model Specific Register（MSR）の特定番地への読み書きによって行う．実際の MSR の読み書きには，Linux カーネル（2.6.32 以降）が提供しているデバイスファイル（/dev/cpu/X/msr(X は論理プロセッサ ID)）に対する pread/pwrite 関数を用いたアクセスを行うことで行った．RAPL では CPU や DRAM の消費エネルギー値が得られるので，それを実行時間で割ることで，平均消費電力を算出した．

アプリケーションプログラムの実行における性能情報（ロード・ストア命令数, 浮動小数演算数など）の取得は，PAPI[7]などの汎用的なライブラリを用いることでも取得できるが，RAPL を用いた電力関係の情報取得，制御と同様に，今回は，MSR への読み書きによって必要な情報を取得した．

RAPL を用いた電力情報取得とその制御，および，性能情報取得を行うにあたり，我々は今回，それらを一定時間間隔で測定して保存して，プロファイルを出力するための関数（API）を作成した．その API をアプリケーションプログラムのソースコード内に挿入することで，簡便に実行プロファイルを取得できるようにした．

## 3. ベンチマークプログラム（OpenFMO）

フラグメント分子軌道（FMO）法は，たんぱく質や DNA などの大規模生体分子に対する電子状態計算（量子化学計算）を行うために開発された並列処理向きの計算手法であり，OpenFMO は九州大学，九州先端科学技術研究所でスクラッチから開発された並列 FMO プログラムである．OpenFMO は MPI と OpenMP を用いたハイブリッド並列化が施されており，マスタ-ワーカ方式を採っている．

OpenFMO の実行において大きな割合を占める処理は，2 電子積分と，その結果を用いた 2 電子ハミルトン行列計算（G 行列計算）である（計算全体の 99.9%以上）．2 電子積分計算では，初期処理で作成したカットオフテーブルの値を参照した初期積分計算と，それに続く漸化計算で得られた値を主記憶上のバッファに格納する処理を行う．2 電子積分には複数の積分タイプ（今回の計算では 21 種類）が存在して，タイプごとに初期積分と漸化計算の割合や 1 回の積分計算で得られる積分数などが異なる．G 行列計算では，2 電子積分計算によってバッファに保存された積分値を読み取り，積和計算を行って G 行列（密対象行列）の生成を行う．本研究では，OpenFMO の主要計算部分を抜き出したベンチマークコード（SCF プログラム）を，性能評価等の実験で用いた．

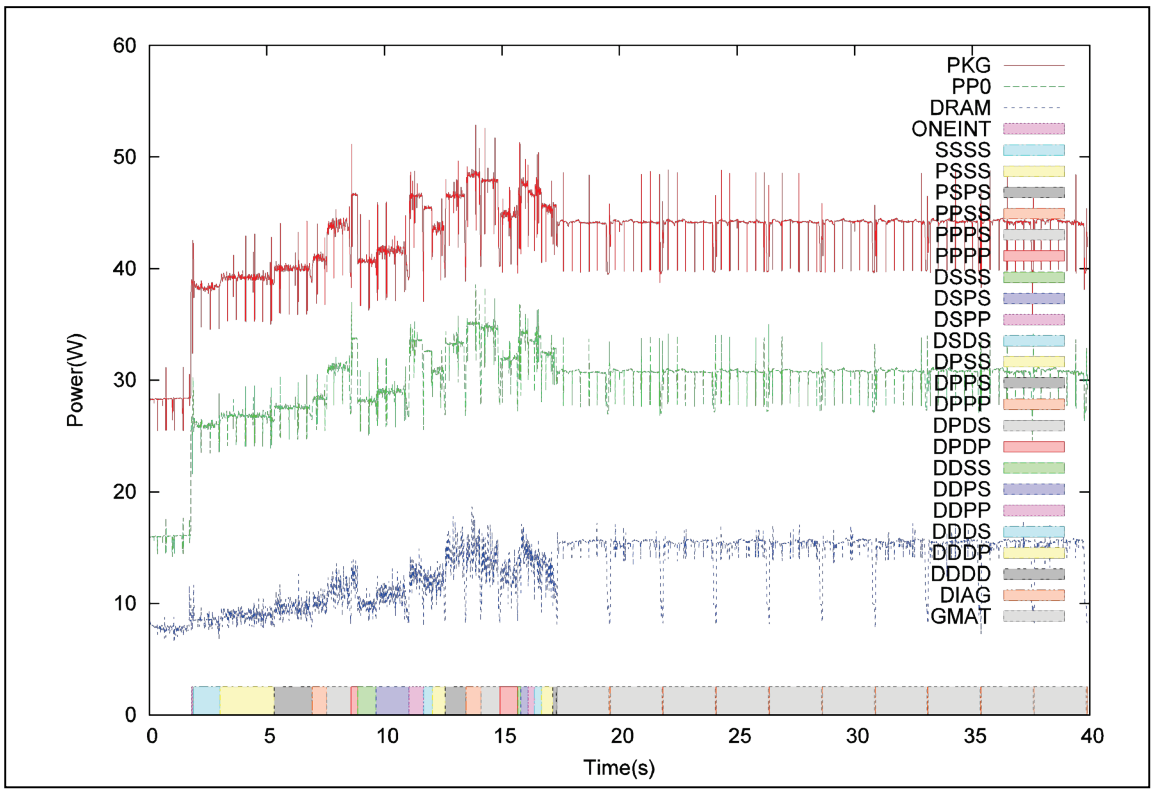


図 1　SCF プログラムの消費電力の推移

Figure 1　Profile of power consumption of the SCF program

## 4. 実験，および，評価

本節では，SCF プログラムに対して行った実験内容と結果，および，それに対する評価について述べる．

### 4.1 実行プロファイル取得

まず，SCF プログラムの消費電力が実行中にどのように変化するかを調べるため，消費電力に関する実行プロファイルを取得した．その結果を**図 1** に示す．この図は，CPU（”PKG”），CPU コア（”PP0”），ならびに，DRAM の消費電力がどのように推移するかを表したグラフであり（データは 10ms 間隔で取得），横軸が実行開始からの経過時間，縦軸が各区間の平均消費電力を示している．グラフの下部に表示してある帯は，その時間に実行されている関数を示している（関数名の詳細に関しては省略）．この結果を見ると，実行開始から 2〜17 秒くらいまでで行われている 2 電子積分計算部分で，消費電力が一定でないことが分かる．これは，同じ 2 電子積分でも，積分タイプ（関数名＝SSSS, PSSS, ..., DDDD）によって消費電力ことを意味している．また，既に報告されているとおり[8]，2 電子積分と G 行列計算部分でも電力性能が違っている．

このように 2 電子積分の積分タイプごとに電力性能が異なることが分かったので，その詳細を調べるために，各タイプの積分計算関数（21 種類）の実行時における平均消費電力，ならびに，各種性能情報を取得した．G 行列計算部分と，SCF プログラムのもう 1 つの主要部分である固有値問題ソルバ（対角化）部分の性能測定も，あわせて行った．その結果を**図 2** に示す．2 つの実線は，CPU（”PKG”）と DRAM の平均消費電力（左軸）を，また，4 つの点線は，サイクルあたりのロード・ストア命令数（”LD/SR INST”），浮動小数命令数（”FP_INST”），総実行命令数（”INST”），および，Last-Level Cache（LLC）ミス（の 100 倍，”LLC MISSx100”）をそれぞれ表している．横軸に示した関数のうち，一番左側の”SSSS”から右から 3 番目の”DDDD”までは，種類の異なる 2 電子積分計算を行う 21 種類の関数である．また，”DIAG”は対角化を，また，”GMAT”は G 行列計算を行う関数を，それぞれ表している．

これを見ると，どの関数においても，平均消費電力が熱

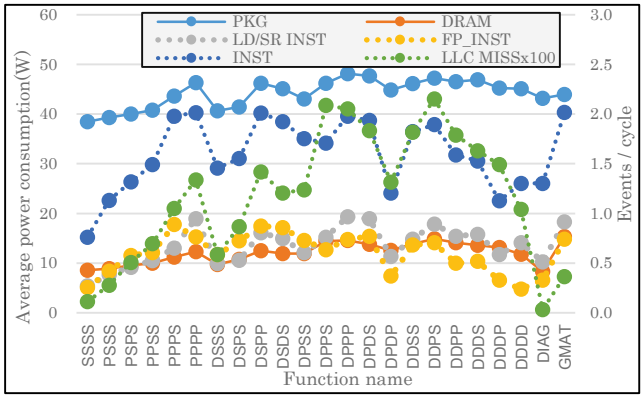


図 2　主要各関数の消費電力，および，性能情報

Figure 2　Power consumption and performance data for mainly used functions in the SCF program

設計電力（今回利用している Xeon E5-2620 では，CPU, DRAM で，それぞれ，95W，35W）よりもかなり小さいことがわかる. 例えば, 詳細データを測定した全関数のうち，最も消費電力が大きかった DPPP タイプの積分計算関数でも，CPU 平均消費電力は 48.0W であった（参考までに，Intel 社が提供している数値演算ライブラリ MKL(version 11.0.5)の行列積関数 DGEMM を実行した場合の CPU 平均消費電力は約 57.5W）．これは，2 電子積分アルゴリズムの複雑さ，ならびに，プログラムの最適化の不十分さから，プロセッサの性能を効率的に利用できていないことに対して，プロセッサの消費電力削減機構がうまく対応できていることを示している. また, 同じ 2 電子積分計算でも, DPPP のように 48W の電力消費をしている関数がある一方で, SSSS や PSSS のように CPU 消費電力が 40W に届かない関数があるなど, 関数間で電力消費に 10W 程度のばらつきがある.

メモリの消費電力については，CPU ほどのばらつきはないものの，G 行列計算で 15.4W 消費しているのに対して，SSSS タイプの積分計算や対角化などでは, その消費電力が 9W 未満である.

各種性能情報も，関数毎に大きく異なることが分かる. 例えば，G 行列計算ではサイクルあたり命令数（IPC）が 2 程度であるが，SSSS 関数では 0.76 と，かなり小さな値であった．また，サイクルあたり LLC ミス数（の 100 倍）に関しても，ほぼ 0 の対角化から，2 を超えている DPPS, DPPP，および，DDPP などとの間に大きな差があることが分かった.

これらの結果から，PPPP や DPPP など IPC が大きな関数のほうが，IPC の小さな SSSS や PSSS などに比べて平均消費電力が大きい傾向が見られる．しかし，今回性能測定した 23 関数の中で IPC が最も高い G 行列計算よりも 0.5 ほど IPC が低い DDDS 関数のほうが，平均消費電力が高いなど，必ずしも IPC と平均消費電力との相関が高いわけではない.

同様に, LLC ミス率と DRAM 消費電力の間にも相関があるようだが，G 行列計算の 4 倍以上の LLC ミス率を持つ DDPS 関数よりの方が G 行列計算よりも DRAM 消費電力が低いなど，こちらも特に関係性が高いわけではない．現在，CPU や DRAM での平均消費電力と性能情報との関係性について，更なる解析を行っているところである.

### 4.2 電力制約下での実行性能変化

電力制約時の SCF プログラムの実行性能の概要を知るために, RAPL を用いて CPU に対して様々な電力制約値を適用して，その状況下での SCF プログラム全体の実行時間，および，CPU と DRAM の平均消費電力を測定した．その結果を**図 3** に示す. 横軸は CPU に対する電力制約値で，今回の測定では，CPU の TDP である 95W から 20W まで 5W 間隔で CPU の電力制約値を変化させた．この図には，CPU

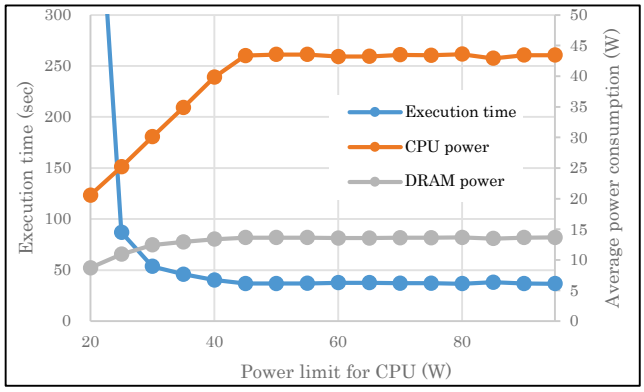


図 3 CPU 電力に制約をかけた場合の実行時間，および，CPU，メモリ平均消費電力の変化

Figure 3 Change of average power consumptions of CPU and DRAM when various power limits for CPU were applied

に対する各電力制約値における SCF プログラムの実行時間（”Execution time”），および，CPU と DRAM の平均消費電力（それぞれ，”CPU power”と”DRAM power”）がプロットされている.

この結果を見ると，電力制約値が 95～45W では，実行時間，および，CPU，DRAM の消費電力がほとんど変化しないことがわかる．これは，先に示したとおり，電力制約を掛けていない場合における SCF プログラムの消費電力が，もともと，45W 前後と低いことによる．一方で，CPU の電力制約値が 45W を下回ると，制約値に応じて CPU の平均消費電力が減少しており，RAPL による電力制約機能がうまく動作していることが分かる．また，CPU 消費電力の減少に伴って，あらわには制約を掛けていない DRAM 平均消費電力も減少している．これは，演算速度が低下したことで DRAM へのアクセス要求速度も低下したためであると考えられる.

CPU 電力制約に伴う実行時間の増加についてみると，電力制約なし時の平均消費電力を下回る電力制約値を適用したあたりから，実行時間も増加している．また，CPU の電力制約値が 20W の場合では，以前の報告[8]にもあるように，CPU 平均消費電力の低下よりも極端に大きな割合で実行時間が増加することが確認できる.

今回の結果から, CPU への電力制約値を加えた場合には，制約値がある一定の値に達するまでは, 計算時間や CPU や DRAM の平均消費電力，ならびに，実行時間にほとんど変化が見られないが，電力制約値がアプリケーションプログラム本来の平均消費電力を下回ると，CPU や DRAM の消費電力も低下するとともに，実行性能も悪くなることが分かった.

### 4.3 電力バジェット配分の違いによる実行性能変化

前述のとおり，SCF プログラム中の関数間，特に，同じ 2 電子積分関数間でさえも，消費電力に差が見られることが明らかとなった．これは，消費電力に制約を加えた際の実行性能への感受性が，関数毎に異なる可能性があることを

Vol.2013-ARC-207 No.30
Vol.2013-HPC-142 No.30
2013/12/17

示唆している．そこで，CPU と DRAM へ電力バジェット配分の仕方によって各関数の性能がどのように変化するかを調べた．まず，CPU と DRAM に対する電力制約値の合計（全電力バジェット）を設定して，その全電力バジェットの CPU，DRAM への配分方法を変化させて，各関数の実行時間を測定した．また，全電力バジェットは 60W から 35W まで 5W 毎に設定して，それぞれの全電力バジェットについて，電力バジェット配分を変えた実行時間測定を行った．

その結果を**図 4** に示す．この図には，60W から 35W までの各全電力バジェットにおいて，性能が最もよくなる，すなわち，計算時間が最短となる電力配分を行った際の DRAM 電力バジェット配分（電力制約）値を，関数毎にプロットしたものである．この結果から，同じ全電力バジェットでも，計算時間が最短になる電力バジェット配分が関数毎に異なることが分かる．例えば全電力バジェットが 60W の場合で見ると，G 行列計算は CPU，DRAM の電力制約値が 18W，42W（=60-18）の場合に計算時間最短になっているが，2 電子積分の 1 つである SSSS 関数では DRAM，CPU の電力制約値が 9W, 51W が最良の電力バジェット配分となっている．また，同じ 2 電子積分関数間でも最適な電力バジェット配分が異なっている．

一方で，全電力バジェットが異なる場合の DRAM への最適な電力制約値を関数毎に見ると，その傾向が類似していることが分かる．例えば，G 行列計算（”GMAT”）については，同じ全電力バジェット下では DRAM 電力制約値が他の関数よりも少し高い傾向があり，逆に，SSSS 関数では最適な DRAM 電力制約値が他の関数よりも低い．このような関数毎に最適な電力バジェット配分値が異なるのは，各関数の性能の違いに起因すると考えられる．現在，最適な電力バジェット配分が，どのような性質（B/F 値やサイクルあたり命令数など）と関係性があるかを調査している状況である．

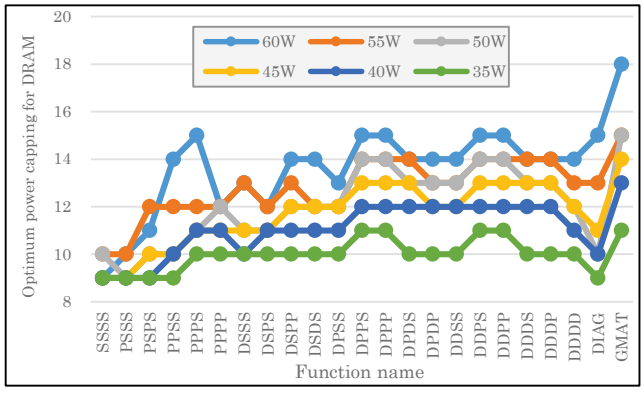


図 4　各電力バジェットにおける最適な DRAM 電力制約値

Figure 4　Optimal power limit for DRAM with various total power budget

### 4.4 SCF プログラムの電力最適化

図 4 に示した結果により，60W から 35W の全電力バジェットにおける各関数への最適な電力バジェット配分が分かった．そこで，その結果を元に RAPL を用いて関数毎に最適な CPU と DRAM への電力制約値を設定して，各全電力バジェットにおける SCF プログラムの最適化を行い，その計算時間を測定した．今回の実験では，最適（と思われる）電力バジェット配分も含めた 3 つの電力配分方式を適用して，各電力配分方法によって SCF プログラムの実行時間がどのように変化するのかを調べた．以下に，その 3 つの電力配分方法について記す．

**(1) naïve な電力バジェット配分（naive）**

naïve な電力バジェット配分として，CPU と DRAM への電力配分を，その定格消費電力（TDP）の比率と等しくする方法，と定義する．ただし，CPU や DRAM には，ある値以下に設定すると実行性能が極端に悪化する，という値が存在することが，実験で判明しているため，その限界値を「最低消費電力」とし，この最低消費電力を CPU と DRAM へ割り振った残りを，定格消費電力の比で分配する，という方針で電力配分を行う方法を，本稿では採用した．その具体的な電力バジェット配分の式を以下に示す．

$$P_{\mathrm{CPU}} = \frac{P_{\mathrm{total}} - \left(limit_{\mathrm{CPU}} + limit_{\mathrm{DRAM}}\right)}{\left(TDP_{\mathrm{CPU}} - limit_{\mathrm{CPU}}\right) - \left(TDP_{\mathrm{DRAM}} - limit_{\mathrm{DRAM}}\right)} \times \left(TDP_{\mathrm{CPU}} - limit_{\mathrm{CPU}}\right) + limit_{\mathrm{CPU}}$$

$$P_{\mathrm{DRAM}} = \frac{P_{\mathrm{total}} - \left(limit_{\mathrm{CPU}} + limit_{\mathrm{DRAM}}\right)}{\left(TDP_{\mathrm{CPU}} - limit_{\mathrm{CPU}}\right) - \left(TDP_{\mathrm{DRAM}} - limit_{\mathrm{DRAM}}\right)} \times \left(TDP_{\mathrm{DRAM}} - limit_{\mathrm{DRAM}}\right) + limit_{\mathrm{DRAM}}$$

ここで，$P_{\mathrm{CPU}}$，$P_{\mathrm{DRAM}}$ は全電力バジェット $P_{\mathrm{total}}$ の場合の CPU，DRAM への電力制約値（電力バジェット配分値）であり，$TDP_{\mathrm{CPU}}$，$TDP_{\mathrm{DRAM}}$ は CPU，DRAM の定格消費電力，$limit_{\mathrm{CPU}}$，$limit_{\mathrm{DRAM}}$ は CPU，DRAM の最低消費電力を，それぞれ表している．本稿で用いる定格消費電力と最低消費電力（プロセッサあたり）は，**表 2** に示している．

表 2　定格消費電力および最低消費電力

Table 2　Thermal design power and lowest power

| | 定格消費電力 | 最低消費電力 |
|---|---|---|
| CPU | 95[W] | 20[W] |
| DRAM | 35[W] | 10[W] |

**(2) 静的な電力バジェット配分（static）**

静的な電力バジェット配分とは，SCF プログラムの実行を通して，一定の電力バジェット配分を行うことをさす．本稿では，そのような電力バジェット配分のうち，各全電力バジェットの下で実行時間が最短になった結果を用いている．

Vol.2013-ARC-207 No.30
Vol.2013-HPC-142 No.30
2013/12/17

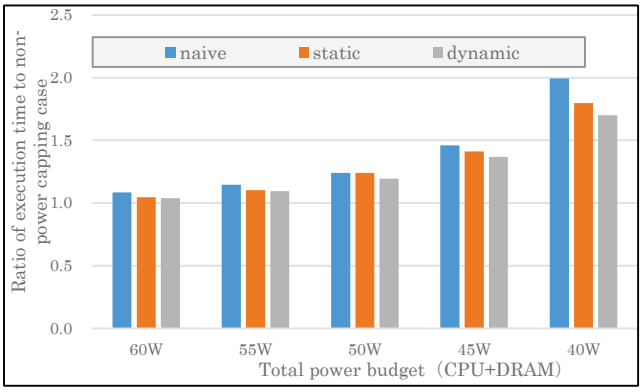


図５ 電力制約時における非電力制約時との計算時間の比
Figure 5 Ratios of execution time under power limit to no-power limit case

(3)**動的な電力バジェット配分（dynamic）**

動的な電力バジェット配分とは，各関数の実行時間が最短になるような電力バジェット配分を，SCF プログラム内で関数毎に変更する配分方法である．各関数の最適な電力バジェット配分は，図 4 に示している結果を用いている．

以上の電力バジェット配分を適用して，SCF プログラムの実行時間を測定した結果を**図 5** に示す．この図には，全電力制約値が 60W～40W の場合の，3 つの電力バジェット配分方法における実行時間と，非電力制約時の実行時間との比を示している．この数値が 1 を超えると，非電力制約時に比べて実行時間が長くなったことを表す．

この結果を見ると，全電力バジェットが小さくなるにしたがって実行時間が伸びているが，電力バジェット配分を naïve な手法から変更することで, 性能が改善されていることが分かる．また，SCF プログラム全体で一定の電力バジェット配分（静的電力バジェット配分）を行った場合よりも，関数毎に最適な電力バジェット配分を設定（動的電力バジェット配分）したほうが，より大きな性能向上を達成していることが分かる．例えば，全電力バジェットが 55W の場合で見ると，静的配分では naïve な配分の場合に比べて約 3.6%の性能向上であるが, 動的配分の場合には約 4.5%の性能向上を示している．また，性能向上の程度は，全電力バジェットが小さい場合に顕著になる傾向があり，全電力バジェット 40W の場合には，naïve な配分の場合に比べて，静的配分の場合で 9.9%，動的配分の場合には 14.7%の性能向上を達成している．

今回の結果から，その電力バジェット配分方法に対する工夫を行うことで，電力制約下における性能最適化が可能であり，その程度は，今回用いたアプリケーション（SCF プログラム）と Xeon を用いたシステムでは数%～10 数%であることが分かった．

## 5. まとめ

我々は，消費電力の問題から，近い将来のスパコン運用では，ノード数などの計算機資源ではなく，電力バジェットをシステムがユーザに割り当てて，その電力バジェットのもとでアプリケーションプログラムを動作させる，という形態を採ると想定して，そのような「電力指向型スパコン」を利用する場合のアプリケーションプログラムの最適化は，これまでの性能最適化とは異なるものになると考えている．そのような電力バジェット制約がある中で，アプリケーションプログラムの最適化をどのように行うか，ということを考え，その指針を得るために，今回は，量子化学計算プログラムを対象として，電力特性や性能情報の取得，および，電力バジェット配分を適切に行うことによる性能最適化を行った．その結果，電力特性は，１つのアプリケーションプログラム内でも大きく異なっており，その特性に応じた電力バジェット配分を行うことで，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できることが分かった．また，電力特性と IPC や LLC ミスなどの性能情報との間に，緩いながらも，相関があることが明らかとなった．今後は，現在得られている情報の解析でなく，他のアプリケーションプログラムに対する情報取得，解析を行い，最適な電力バジェット配分と性能情報などとの関連性を明らかにして，電力制約下における性能最適化の自動化などに取り組む予定である．

## 参考文献

1) Hans, M., Erich, S., Jack, D. and Horst, S.: Top500 Supercomputer sites. http://www.top500.org/
2) Bergman, K., Borkar, S., Campbell, D., Carlson, W., Dally, W., Denneau, M., Franzon, P., Harrod, W., Hill, K., Hiller, J. et al.: Exascale computing study: Technology challenges in achieving exascale systems, Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO), Tech. Rep (2008).
3) Ishikawa, Y., Maruyama, N. et al.: HPCI 技術ロードマップ白書 (2012).
4) 科学技術振興機構：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.
5) Inadomi, Y., Takami, T., Maki, J., Kobayashi, T. and Aoyagi, M.: RPC/MPI Hybrid Implementation of OpenFMO, Parallel Computing: From Multicores and GPU's to Petascale, Vol. 19, p. 220 (2010).
6) Intel: Intel 64 and IA-32 Architectures Softeware Developer's Manual (2012).
7) Performance Application Programming Interface PAPI http://icl.cs.utk.edu/papi/index.html
8) 吉田 匡兵，佐々木 広，深沢 圭一郎，稲富 雄一，上田 将嗣，井上 弘士，青柳 睦，CPU と主記憶への電力バジェット配分を考慮した HPC アプリケーションの性能評価，第 141 回ハイパフォーマンスコンピューティング研究発表会（沖縄）2013.10.1