Vol.2016-HPC-155 No.17
2016/8/9

電力制約型スーパーコンピュータにおける性能モデリング

稲富 雄一[1,†1,2]　垣深 悠太[1]　小野 貴継[1,2]　井上 弘士[1,2,a)]

概要：スーパーコンピュータ（スパコン）における消費電力問題は演算性能の増大に伴い年々深刻化している．今後は電力性能効率を最大化することが極めて重要となり，そのためにはマイクロプロセッサ・チップや DRAM といったハードウェア資源のみならず，消費電力資源の効果的利用が必要不可欠となる．たとえば，消費電力を考慮した効率的なジョブスケジューリングを行うことで，限られた供給電力の中でシステム・スループットを向上させる取り組みも行われている．システムの電力性能効率を高めるためには，与えられる電力制約がアプリケーション実行時間に与える影響を把握する必要がある．これまでに，スーパーコンピュータを対象とした様々な性能推定技術が提案されてきたが，その殆どは消費電力制約を考慮していない．そこで本研究では，電力制約型スーパーコンピュータを対象とした性能推定技術を提案する．本方式の特徴は，マイクロプロセッサ・チップや DRAM の製造ばらつきを考慮した電力制約時実行時間の推定を簡便に行なう点にある．1,920 個のマイクロプロセッサ・チップを搭載したスーパーコンピュータを対象に定量的評価を行った結果，電力制約下におけるアプリケーション実行時間を 15%程度の誤差で推定できることが分かった．

1. はじめに

将来のスーパーコンピュータ（以降，スパコンと略す）では，期待される性能に対して利用できる電力が大きく制約される．たとえば，米国エネルギー省 (DOE: Depertment of Energy) は 20MW 以下の消費電力でエクサフロップスを達成することを目標にしており，電力効率の大幅な改善が必要不可欠であることを示唆している [2,4,16]．このようにスパコンの消費電力問題が注目されている中で，将来のスパコンにおいて限られた電力で高い性能を発揮できるようにするためには，電力効率の良いハードウェアの開発だけでなく，決められた電力バジェットのもとでアプリケーションプログラム（以降，アプリと略す）の実効性能を最大化することが重要になる．また，アプリの消費電力特性と実行時間を調整してジョブスケジューリングを的確に行うことで，決められた消費電力の範囲内でシステム全体のスループットを最大化することも必要になる．このような電力性能最適化を実施するには，電力制約下におけるアプリ実行性能を正確に推定することが必要となる．従来，スパコンを対象とした様々な性能推定技術が提案されてきた [3,17]．しかしながら，消費電力制約は考慮されておらず，電力性能効率を考慮した新しい性能推定技術が必要となる．

これまでに我々は，将来のスパコンの消費電力問題に着目し，電力供給量を超えないように電力制約を適用しながら運用する「電力制約適応型スパコン」に関する研究を行ってきた．特に，同一カタログスペックのマイクロプロセッサ・チップであっても製造プロセスのばらつきにより消費電力に差が生じる問題が指摘されており [5,7,9,19]，これが電力制約下における並列アプリ性能に極めて深刻な悪影響を与えることを示した [10]．そこで本研究では，製造ばらつき特性を考慮した電力制約下での並列アプリ実行時間推定技術を提案する．電力制約を適用しない場合の実行時間，予め取得した製造ばらつき情報，ならびに，小規模ノードを用いた簡易な電力特性測定結果を入力とし，電力制約下での並列アプリ実行時間を出力する．1,920 個のマイクロプロセッサ・チップを搭載したスパコンを用いた大規模実験を行った結果，さまざまな電力制約下における並列アプリ実行時間を 15%程度の誤差にて推定することができた．

本稿の構成は以下の通りである．まず，第 2 節では本研究で用いた実験環境を説明する．次に，第 3 節にて電力制約時実行時間推定の方針や，その際に用いる消費電力/実行時間モデルの詳細を示す．第 4 節では提案手法による電力制約時実行時間推定の精度評価の結果を報告し，最後に第 5 節にてまとめと今後の課題を述べる．

[1] 九州大学, Kyushu University
[2] CREST, JST
[†1] 現在，（株）チーム AIBOD
[a)] inoue@ait.kyushu-u.ac.jp

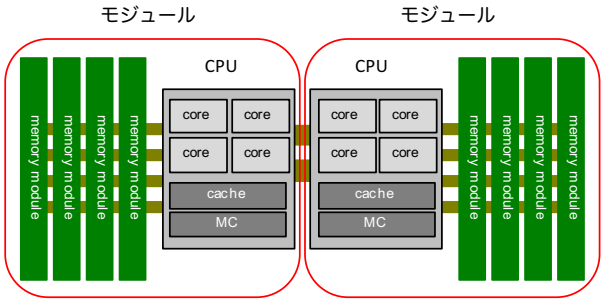

図 1　CPU とモジュール

表 1　計算機環境

ノード数	960（965 ノード中）
CPU	Intel Xeon E5-2697 v2@2.7GHz 12 コア ×2 ソケット/ノード
主記憶	256GB (DDR3-1600)/ノード
インターコネクト	InfiniBand FDR (片方向 6.78GB/s)
OS	Red Hat Linux Enterprise 6
コンパイラ	Intel C++/Fortran Compiler (version 15.0.3)
MPI ライブラリ	Intel MPI (version 5.0)
数値演算ライブラリ	Intel Math Kernel Library (version 11.2.3)

2. 実験環境

2.1 用語の定義

本稿では以下に示す用語を用いる．

- **CPU**（マイクロプロセッサ・チップ）：（複数の）コア，キャッシュ，メモリコントローラなどが搭載された物理的なマイクロプロセッサ・チップ．
- モジュール：CPU とそれに直接接続された DRAM の組（図 **1** 参照）
- 正規化実行時間：非電力制約時の実行時間に対する電力制約時実行時間の比．電力制約を施すことで生じる実行時間増加率を表す．

2.2 電力，動作周波数の制御・測定用ライブラリ

電力制約や動作周波数制約を適用したアプリの実行，ならびに，アプリ実行時の消費電力や実行時間，平均動作周波数，各種性能カウンタ値を測定するために専用ライブラリ（RIC ライブラリと呼ぶ）を開発した．RIC ライブラリでは CPU への電力制約や CPU と DRAM の消費電力（消費エネルギー）を測定するために，SandyBridge 以降のインテルプロセッサに搭載されている Running Average Power Limit(RAPL) [11] を利用している．RAPL では CPU 全体とコア部分，CPU に直接接続された DRAM 消費エネルギーをそれぞれ測定することができる．また，一定時間間幅（デフォルトでは約 1ms）にて平均消費電力の制約値を指定することが可能である．本研究では，このような電力キャッピング機能を用いて CPU への電力制約を行う．

一方，CPU 動作周波数の制御に関しては Linux カーネルでサポートされている cpufreq の機能を利用し，RIC ライブラリ経由で制御できるようにした．なお，RAPL の仕様では CPU だけではなく DRAM にも電力制約を指定することができるが，本研究で使用したスパコンでは DRAM への電力制約指定がサポートされていない．そのため，直接的に電力制約を施す（RAPL を経由して電力制約値を明示的に指定する）のは CPU のみとした．ただし，電力制約値そのものは，CPU と DRAM の消費電力相関を考慮してモジュール単位で（つまり，CPU と DRAM の合計消費電力を考慮して）制御する．したがって，消費電力バジェットの配分はモジュール単位となる．

2.3 プラットフォーム

本研究では，九州大学情報基盤研究開発センターの HITACHI HA8000-tc/HT210 を占有利用して実験を行った．本スパコンの諸元を 表 **1** に示す．12 コアの Intel Xeon プロセッサ 2 ソケット，および，256GB の主記憶を搭載するノードが InfiniBand で相互結合されている．インテルコンパイラを使用し，一部ベンチマークで利用する数値演算ライブラリとして，Intel Math Kernel Library (MKL) を用いた．

2.4 ベンチマーク

2.4.1 *DGEMM, *STREAM(Scale, Triad), および*Random Access

*DGEMM と *STREAM(Scale, Triad)，および，*Random Access は HPC challenge [12] に含まれているベンチマークプログラムである．*DGEMM は High Performance Linpack (HPL) [15] のカーネルとしても知られている行列–行列積（DGEMM）を MPI で起動された全プロセスで実行す る計算律速の Embarassingly Parallel(EP) タイプのアプリである．本研究ではインテル社が提供している数値演算ライブラリ MKL に実装されている最適化されたスレッド並列化済みの DGEMM 関数を利用した．*STREAM(Scale) は 2 つのベクトル $\mathbf{a}, \mathbf{b}$ と 1 つの定数 α に対して，$\mathbf{b} = \alpha\mathbf{a}$ を計算するという処理を， また，*STREAM(Triad) は 3 つのベクトル $\mathbf{a}, \mathbf{b}, \mathbf{c}$ および 1 つの定数 α に対して $\mathbf{c} = \alpha\mathbf{a} + \mathbf{b}$ を計算するという処理を，起動された全 MPI プロセスで実行するメモリ律速の EP タイプアプリである．本実験では AVX 命令を利用するように変更したコードを作成し利用した．各モジュールに搭載された DRAM 容量を超えないように，各ベクトルサイズは 24GB とした．

*Random Access は巨大 64 ビット整数配列の要素に対するメモリアクセス（読み出し，更新，書き込み）をラン

ダムに行う，DRAM へのランダムアクセス性能を測定するベンチマークアプリである．*STREAM の場合と同様に，各モジュールの DRAM 容量を超えないように配列サイズを 72GB とした．以降，上記のアプリの略称として DGEMM（または dgemm），scale，triad，ra と表記する．

2.4.2 MHD

MHD（Magneto Hydro Dynamics）シミュレーション [8]（略称 MHD）は，太陽風と呼ばれる太陽から放出される磁場を伴ったプラズマと惑星の磁場との相互作用を解明するために用いられる電磁流体シミュレーションの一種である [14]．本研究で用いた MHD シミュレーションコードは，MPI と OpenMP によるハイブリッド並列化が行われている．シミュレーション空間を 3 次元領域にメッシュ分割し，各領域に 1 つの MPI プロセスを割り当て，さらに内部に含まれるループをスレッドに分割して計算を行う．MHD シミュレーションでは，MHD 方程式と呼ばれる偏微分方程式を解くための差分計算が主な処理であり，計算と隣接通信を繰り返し実行する典型的なステンシル型アプリである．

2.4.3 NAS Parallel Benchmark (NPB)–BT, SP

NAS parallel benchmark [1] 中の各種ベンチマークアプリのうち，MPI/OpenMP のハイブリッド並列化バージョン [20] に含まれるブロック 3 重対角行列ソルバ（BT）と 5 重対角行列ソルバ（SP）を利用した（以降，それぞれ NPB(BT)，NPB(SP) と記す）．実行時の問題クラスは，小規模（64）並列時には class C を，また，大規模（1920）並列時には class E をそれぞれ用いた．

2.4.4 mVMC-mini

mVMC-mini は強い電子相関を持つ分子系の電子状態計算を行う多変数変分モンテカルロシミュレーションの典型的な処理の性能評価を容易に行うために開発された小規模アプリケーションプログラム [18] である．Fiber benchmark suite [13] に含まれている（以降，mVMC と記す）．

2.4.5 Rodinia benchmark suites

Rodinia benchmark suite [6] は各種演算アクセラレータ向けに開発されたベンチマークアプリ群であり，医療画像処理や計算物理，パターン認識，データマイニングなど様々な分野のアプリで利用されている 20 種類以上のカーネルコードで構成されている．各コードは，CUDA や OpenCL および，OpenMP により並列化が施されているが，本研究では汎用プロセッサである Xeon プロセッサで動作させることを目的としているため OpenMP で並列化された 16 種類のコードを利用した．利用したコードの名称は 表 **2** に示す通りである．

表 **2** 利用した Rodinia benchmark suite 内のアプリ（カッコ内は略称）

Leukocyte (leukocyte)	Heart Wall (heartwall)
CFD Solver (cfd)	LU Decomposition (lud)
HotSpot (hotspot)	Back Propagation (backprop)
Needleman–Wunsch (nw)	Kmeans (kmeans)
Breadth-First Search (bfs)	SRAD[1] (srad1, srad2)
Streamcluster (sc)	Particle Filter (particle)
PathFinder (path)	k–Nearest Neighbors (nn)
LavaMD (lavaMD)	

1; SRAD には 2 つのカーネルアプリが含まれる

3. 電力制約を考慮した性能推定法

3.1 基本方針

電力制約を適用せずに大規模実行した場合の実行時間 T_0 は何らかの推定手法を用いて得られていると仮定し，電力制約を適用して大規模実行した場合の実行時間を推定する．アプリ実行時に電力制約を適用した場合，非制約時に比べてどの程度実行時間が長くなるかを表す値，すなわち，非制約時実行時間に対する電力制約時実行時間の比（＝正規化実行時間）r を知ることができると仮定すると，非制約時の実行時間との積として電力制約適用時のアプリ実行時間 T を推定することが可能である．

$$T = T_0 \times r$$

本手法では，正規化実行時間と電力制約値との関係は，アプリ特性ならびにモジュール特性の両方に依存すると考え，電力制約時の正規化実行時間をモジュールごとに推定して，電力制約時の並列アプリ実行時間を求める．また，モジュール消費電力を制約した場合の正規化実行時間を推定する際に，(1) モジュール電力制約値からモジュール動作周波数を推定し，(2) 得られたモジュール動作周波数を用いて正規化実行時間を推定する，といった 2 段階を経て電力制約値から正規化実行時間を推定する．この電力制約値からの正規化実行時間推定はシミュレーションのような高コストの手法を用いず，消費電力と動作周波数，および，動作周波数と正規化実行時間との関係をモデル化し，得られたモデル式を用いて正規化実行時間を推定する．すなわち，動作周波数 f を消費電力（電力制約値）P の関数（$f = f(P)$）として，また，正規化実行時間 r を動作周波数 f の関数（$r = r(f)$）としてそれぞれ表し，これら 2 つのモデル関数を使って電力制約値と正規化実行時間の関係を求める．

$$r = r(P) = r\left(f(P)\right)$$

本提案手法では以下の仮定を設ける．

(1) 同一動作周波数での実行時間はモジュールに依存せず一律である．
(2) 正規化実行時間の周波数依存性はアプリ特性に依存するが，モジュールならびに入力サイズに対しては非依存である．

並列アプリ実行時には，各モジュールで動作周波数が異なり，結果として正規化実行時間がモジュール間で異なる場合

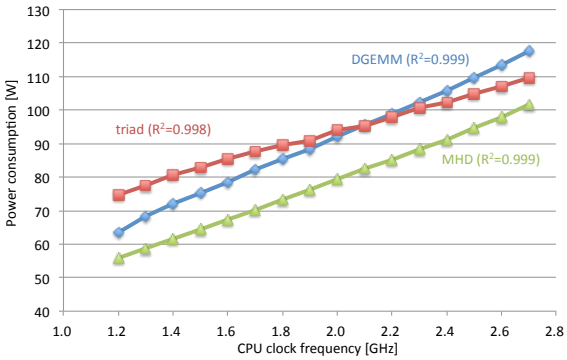

図 2　各アプリの動作周波数と消費電力の関係

も想定される．このように正規化実行時間 $\{r_i\}$ がモジュールごとに異なる場合には，その最大値 $r_{\max} = \max\{r_i\}$ と非制約時実行時間との積を電力制約時の並列アプリ実行時間 T とする．

$$T = T_0 \times r_{\max} \tag{1}$$

3.2　モジュール消費電力–動作周波数相関モデリング

マイクロプロセッサなどの半導体の動的消費電力が動作周波数と電源電圧の 2 乗に比例関係があることが一般的に知られている．しかしながら，電源電圧が変化しない場合には，実際に HPC システムに搭載されているモジュールの消費電力と動作周波数との間に，アプリの性質（計算律速かメモリバンド幅律速か）に依らず線形関係があることが示されている [10]．DGEMM, triad，および，MHD を 64 プロセス（12 スレッド/プロセス）並列で動作周波数制約を適用して実行した場合の消費電力を測定し，各動作周波数制約時のプロセス間平均モジュール消費電力をプロットした結果を 図 2 に示す．図中のベンチマーク名の後の ( ) 内に記されている R^2 は消費電力を動作周波数で線形近似した場合の近似値と実測値との相関係数である．本図より，演算律速の DGEMM，メモリバンド幅律速の triad，および，通信を伴うステンシル型アプリである MHD の全てにおいて，動作周波数の一次関数で近似した推定消費電力と平均消費電力実測値との相関係数が 0.99 を超えていることが分かる．これは，アプリの性質に関わりなくモジュール消費電力と動作周波数との間に線形関係を想定することが妥当であると示唆している．また，文献 [10] などで示されているように，製造ばらつきを原因とした電力消費特性のばらつきにより，全く同じマイクロアーキテクチャを持つ CPU にて同一アプリを実行した場合でもモジュール間で消費電力が異なる．そこで，文献 [10] と同様に，モジュール i の消費電力 P_i と動作周波数 f_i の間の線形関係を，定数 $\alpha\,(0 \le \alpha \le 1)$ を使って式 (2) および式 (3) のように表す．

$$P_i = \alpha_i \left(P_i^{\max} - P_i^{\min}\right) + P_i^{\min} \tag{2}$$

$$f_i = \alpha_i \left(f^{\max} - f^{\min}\right) + f^{\min} \tag{3}$$

ここで，$\{P_i^{\max}\}$, $\{P_i^{\min}\}$ は，それぞれ，モジュール i の非電力制約（最高動作周波数）時，最低動作周波数時でのアプリ実行時のモジュール消費電力を示す．また，$f^{\max}$, $f^{\min}$ は，それぞれ最高動作周波数と最低動作周波数である．

2 つの消費電力パラメタ $\{P_i^{\max}\}$, $\{P_i^{\min}\}$ は，アプリおよびモジュールに依存するが，プロセッサ動作周波数パラメタ $f^{\max}$, $f^{\min}$ はマイクロアーキテクチャ固有の値であり基本的にアプリならびにモジュールとは非依存である．式 (2) および式 (3) を利用することで，アプリ全体（利用モジュール数 n）での消費電力バジェット P^{budget} が与えられた場合の各モジュールの電力制約値や動作周波数を求めることができる．まず，全モジュールで一律の電力制約を適用する場合を考える．この場合は各モジュールの電力制約値 $\bar{P}$ は電力バジェットを利用モジュール数で等分したものになり，各モジュールの動作周波数 f_i は電力消費特性のばらつきによりモジュールごとに異なる値を持つ．

$$\begin{aligned}
\bar{P} &= \frac{P^{\text{budget}}}{n} \\
\alpha_i &= \frac{\bar{P} - P_i^{\min}}{P_i^{\max} - P_i^{\min}} \\
f_i &= \alpha_i \left(f^{\max} - f^{\min}\right) + f^{\min}
\end{aligned}$$

一方，文献 [10] で報告されている電力特性のばらつきを考慮した電力配分を行う場合の各モジュール消費電力と動作周波数は，次式のように表される．

$$\begin{aligned}
\alpha &= \frac{P^{\text{budget}} - \sum_i^n P_i^{\min}}{\sum_i^n P_i^{\max} - \sum_i^n P_i^{\min}} \\
P_i &= \alpha \left(P_i^{\max} - P_i^{\min}\right) + P_i^{\min}
\end{aligned}$$

この電力配分法では動作周波数はモジュール間で同じ（定数 α が全モジュールで同じ）だが，モジュールの電力消費特性のばらつきによってモジュールごとに電力配分（電力制約値）が異なる．このようにして，式 (2) ならびに (3) で表される消費電力–動作周波数相関モデルを用いることで，電力制約時の各モジュールの（平均）動作周波数を決定できる．

ここで，式 (2) を用いる消費電力–動作周波数相関モデルでは非制約時，最低動作周波数制約時のアプリ実行時の各モジュールの消費電力データが必要になるが，システムで動かす予定のあるすべてのアプリに対して，かつ，システム内のすべてのモジュールに対してこの消費電力情報を取得することは困難である．この問題は，各モジュールの消費電力ばらつきがアプリに依存しないと仮定し，1) 小規模なベンチマークプログラム（μ ベンチマーク）をシステム内の全モジュールを使ってシステム導入時に実行し，各モジュールで取得した消費電力情報に基づき生成する消費電力ばらつきテーブル（Power Variation Table, PVT）と，2) アプリの小規模実行で得られた消費電力情報を使って，システム内の全てのモジュールでのアプリ実行時消費電力

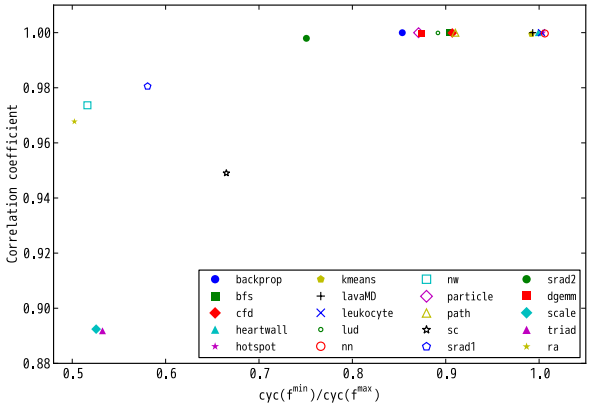

図 3　最低・最高動作周波数時でのサイクル数比と反比例式での正規化実行時間近似値と実測結果との相関係数

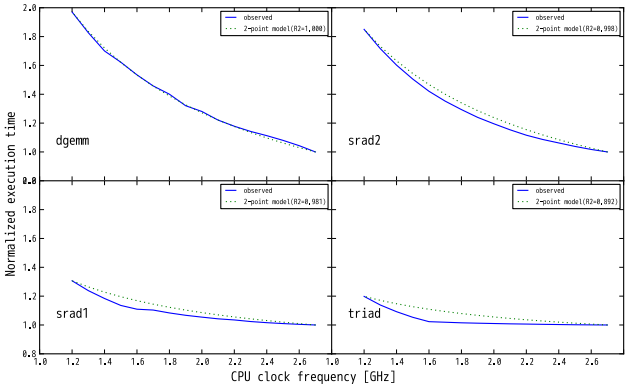

図 4　動作周波数に対する正規化実行時間の変化と 2 点反比例モデルでの近似曲線

を推定する手法を用いることで解決できる [10]．そこで，第 4 節での性能評価では，PVT を利用した推定消費電力を用いた消費電力–動作周波数相関モデルと，消費電力の実測値を用いて構築した消費電力–動作周波数相関モデルの両方での評価を行った．

3.3　動作周波数–正規化実行時間モデル式

実行サイクル数が動作周波数の変化に関わらず一定の値を持つ場合には，正規化実行時間と動作周波数には反比例関係がある．ただし，メモリアクセス遅延によるプロセッサ・ストールが多く発生するようなアプリの場合には，プロセッサの動作周波数を低下した場合に（ストール数が減少することにより）実行クロック・サイクル数が減少するため，正規化実行時間が動作周波数の反比例式からずれることが予想される．そこで，HPC challenge や Rodinia benchmark suite に含まれるベンチマークアプリ（20 種類）に対し，動作周波数制約を変えながら実行して，動作周波数と正規化実行時間との関係を調べた．

図 **3** は，非制約実行時のクロックサイクル数に対する最低動作周波数制約実行時のクロックサイクル数の比（＝クロックサイクル数比）と，正規化実行時間を動作周波数を変数とした 1 つの反比例式（非制約時と最低動作周波数時の実行結果を用いた反比例モデル＝ **2** 点反比例モデル）で近似した場合の推定値と実測値の相関係数との関係を表したグラフである．横軸はクロックサイクル数の比であり，この値が 1.0 から離れて小さくなるほど，動作周波数低下に伴うクロックサイクル数減少の程度が大きい，すなわち，DRAM アクセスに伴うストールが大きいアプリであるという指標である．縦軸は，正規化実行時間を動作周波数に対する 2 点反比例モデルによる近似値と実測値の相関係数であり，1.0 に近ければ 1 つの反比例式で精度よく近似できており，1.0 から離れるほど 2 点反比例モデルによる近似精度が低下することを表す．図 3 には Rodinia benchmark に含まれる 16 種類，および，HPCC に含まれる 4 種類の

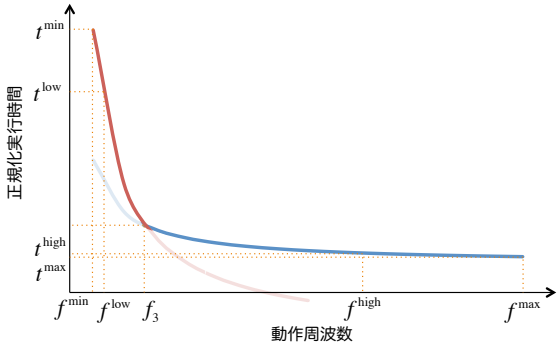

図 5　4 つの動作周波数制約下での実測値を基にした近似 3 点モデルの求め方

計 20 種類のベンチマークアプリに対する結果をプロットしている．図 3 より，クロックサイクル数比が 0.8 を超える，すなわち DRAM アクセス遅延が小さいアプリでは正規化実行時間が 2 点反比例モデルでうまく近似できている（相関係数が 1.0 に非常に近い）が，それよりクロックサイクル数比が小さく DRAM アクセス遅延が大きいアプリでは，2 点反比例モデルと実測値の乖離が大きくなっている（相関係数が 1.0 から大きく離れている）ことが分かる．

図 3 に示されている 20 種類のアプリのうち，クロックサイクル数比が異なる 4 つのアプリ（`dgemm`, `srad2`, `srad1`, `triad`）に対して，正規化実行時間と動作周波数変化の関係をプロットしたグラフを図 **4** に示す．図 4 には 4 つのアプリの正規化実行時間の実測値（青実線）と 2 点反比例モデルでの近似曲線（緑破線）が記されており， 凡例の `2-point model` の横に記載されている数値（`R2`）は実測値と近似値との相関係数である．この結果より，DRAM アクセス遅延が小さい `dgemm` では 2 点反比例モデルで正規化実行時間がうまく近似されているが，クロックサイクル数比が小さくなるに従って（`srad2`→`srad1`→`triad` の順で）2 点反比例モデルでの近似精度が悪化している（相関係数が小さくなる）ことが分かる．このように，DRAM アクセス遅延が大きいアプリでは，単純な 2 点反比例モデルで正規化実行時間をモデル化することが困難であることが分

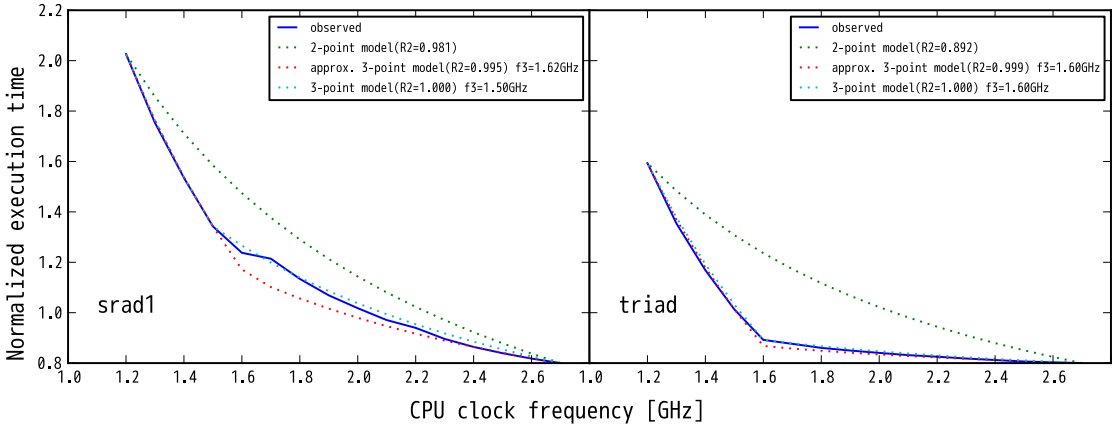

図 6　近似 3 点反比例モデルで近似した正規化実行時間の動作周波数変化近似曲線と実測値

かった．

ここで，図 4 のうち 2 点反比例モデルでの近似精度が悪い 2 つのアプリ（srad1, triad）において，動作周波数に対する正規化実行時間の変化の様子を見ると，高周波数側と低周波数側で変化の仕方が異なっていることが分かる．そこで，高周波数側と低周波数側を異なる反比例式で近似する方法（**3** 点反比例モデル）を導入する．また，モデル式構築で必要となる実行時間情報取得のための動作周波数制約下でのアプリ実行回数を少なくするために，高周波数側，低周波数側でそれぞれ 2 つの周波数，合計 4 つの周波数制約下での実行時間実測値だけを用いて，高周波数側，低周波数側の反比例モデル，および，両者を切り替える動作周波数を決めることにする（このモデルを近似 **3** 点反比例モデルと呼ぶ）．対象アプリで近似 3 点反比例モデルを求めるための手順を図 **5** を用いて説明する．このモデル式を作成するためには，(1) 高周波数側では最高動作周波数 $f^{\max}$ とそれより若干低い動作周波数 f^{high}，また，低周波数側では最低動作周波数 $f^{\min}$ とそれより若干高い動作周波数 f^{low}，の計 4 種類の動作周波数制約下でアプリを実行し，各周波数制約下での実行時間（それぞれ，$t^{\max}$, t^{high}, $t^{\min}$, t^{low}）を取得する．(2) 次に，高周波数側の $(f^{\max}, t^{\max})$, $(f^{\mathrm{high}}, t^{\mathrm{high}})$ の 2 点のデータを使って高周波数側の反比例モデル式を，また，低周波数側の $(f^{\mathrm{low}}, t^{\mathrm{low}})$, $(f^{\min}, t^{\min})$ の 2 点のデータを使って低周波数側の反比例モデル式をそれぞれ求める．(3) そして，得られた 2 つの反比例式の交点の動作周波数を高周波数側と低周波数側を分ける動作周波数 f_3 とする．今回の実験では，4 種類の動作周波数の値として $f^{\max} = 2.7$, $f^{\mathrm{high}} = 2.4$, $f^{\mathrm{low}} = 1.5$, $f^{\min} = 1.2$（単位 GHz）を用い，これらの周波数制約下での実測値を利用してモデル構築を行った．

上述した近似 3 点反比例モデルを用いて，正規化実行時間の動作周波数変化を近似した結果を図 **6** に示す．図 6 には 2 点反比例モデルでの近似精度が悪いことが図 4 で示されていた 2 つのアプリ（srad1, triad）に対して，近似 3 点反比例モデルを用いた近似曲線（図中凡例の approx. 3-point model で示されている赤破線曲線）を 2 点反比例モデルでの近似曲線とともに記載している．また，本図には，高周波数側と低周波数側を分ける動作周波数 f_3 を実験結果との一致が最もよくなるように選んで構築した 3 点反比例モデルの結果（図中凡例の 3-point model で示されている水色破線曲線）も合わせて記している．凡例部分に示してある R2，および，f3 の値は，それぞれ，モデルでの推定値と実測値の相関係数，および，周波数領域を分割する動作周波数 f_3 である．この結果より，2 点反比例モデルでは正規化実行時間の動作周波数依存性をうまく近似できていなかったアプリであっても，3 点反比例モデルを用いることで精度よくモデル化できていることが分かる．また，3 点反比例モデルに比べるとわずかに精度が低下するものの，4 種類の動作周波数制約下でのアプリ実行結果のみを用いた近似 4 点反比例モデルで精度よく正規化実行時間と動作周波数の関係を表現できている（つまり，実測結果を再現できている）ことが分かる．実際，この結果は相関係数の大きさ（srad1, triad に対してそれぞれ 0.995, 0.999）からも示唆される．

以上，本節では正規化実行時間と動作周波数との関係を調べるために Rodinia benchmark などの電力制約下における実行時間実測データを取得し，その実測結果をうまく近似できる 3 点反比例モデル，および，より低コストな近似 3 点反比例モデルを提案し，これらの精度を検証した．その結果，近似 3 点モデルを用いることで，正規化実行時間と動作周波数との関係を精度よく再現できることが分かった．シングルノード，あるいは，少数ノードでのアプリ実行結果を使ってアプリの正規化実行時間–動作周波数相関モデルを構築できれば，その結果と非制約時の大規模実行時間情報，および，消費電力–動作周波数相関モデルを用いて，電力制約下でアプリの大規模並列実行を行った場合の実行時間が推定できる．次節では，この実行時間推定手法の精度を幾つかの並列アプリを使って検証する．

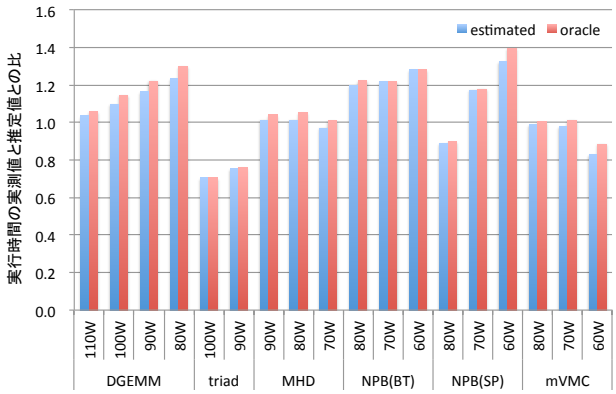

図 7　CPU 一律電力制約適用時の実行時間の実測値と推定値との比

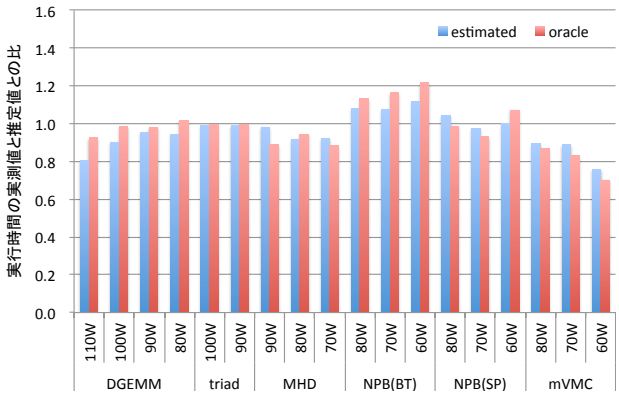

図 8　ばらつきを考慮して CPU 個別の電力制約を適用した場合の実行時間の実測値と推定値との比

4. 大規模並列アプリを用いた推定精度の評価

4.1 評価手順

本節では，電力制約時の大規模（960 ノード，1960 モジュール）スーパーコンピュータを用いた並列アプリ実行時間の推定精度を検証する．以下，本評価の手順を説明する．まず，消費電力–動作周波数の線形モデル，および動作周波数–正規化実行時間の近似 3 点モデルを構築するために，非制約（2.7 GHz），最低動作周波数（1.2 GHz），および，2.4 GHz，1.5 GHz に動作周波数を制約してアプリを小規模並列（32 ノード = 64 モジュール）実行し，消費電力と実行時間のデータを取得する．次に，作成した消費電力–動作周波数モデル（線形モデル），動作周波数–正規化実行時間モデル（近似 3 点反比例モデル），ならびに，非電力制約時の大規模（960 ノード= 1,920 モジュール）実行時の実行時間を用いて，前節で説明した性能推定法により電力制約時の大規模実行時間を推定した．

推定対象は，CPU 電力を全モジュールで一律に制約した場合（CPU 一律制約時）と，文献 [10] で提案された製造ばらつきを考慮して各 CPU に異なる電力バジェットを配分をする場合（CPU 個別制約時）の 2 つのケースにおける大規模実行の実行時間である．評価対象アプリは，DGEMM，triad，MHD，NPB(BT)，NPBSP，および，mVMC の 6 つを対象とした．

4.2 評価結果

図 7 に，全 CPU に一律の電力制約を施した場合における，大規模実行実行時間の実測値と推定値との比を示す．横軸は，アプリとモジュール当たり電力制約値，縦軸は実測値に対する推定値の比であり，1.0 に近いほど推定精度が良いことを表す．ここでは，消費電力–動作周波数の線形モデルを PVT を使用して作成した場合の結果（estimated）と，実測値を基に作成した場合の結果（oracle）の 2 種類の結果を記している．実験結果より，DGEMM，NPB(BT)，NPB(SP) では推定値が実測値よりも数%~40%大きくなっており，triad では逆に 2 割以上小さな値となっている．

一方，MHD では実測値と推定値との一致が非常によいことが分かり，全体としては比較的精度よく推定できている．なお，mVMC に関しては精度良く実行時間を推定した結果となったが，実行毎での測定結果にばらつきが大きいため，本結果のみから精度を評価できない．この問題の解決は今後の課題である．

次に，製造ばらつきを考慮した CPU 個別制約時の実行時間の実測値と推定値の比を図 8 に示す．縦軸は図 7 と同様に実行時間の実測値と推定値との比であり，横軸はアプリの種類と平均モジュール電力制約値（実際の電力制約値はモジュールごとに異なる）となっている．この結果より，すべてのアプリ，電力制約条件下において実行時間の推定値と実測値との比が 1.0 に近く，提案手法による電力制約時のアプリ実行時間推定の精度が高いことが分かる．

推定値ベースの消費電力–動作周波数線形モデルを用いた場合で CPU 一律制約時と CPU 個別制約時の実行時間推定誤差を比較すると，CPU 一律制約時では平均約 15 % の推定誤差があるのに対して，CPU 個別制約時では推定誤差が平均 10 % 程度であり，CPU 個別制約時の推定精度が CPU 一律制約時に比べて高いことが分かった．これは，CPU 一律制約時には消費電力–動作周波数線形モデルを用いて各モジュールの動作周波数を個別に求めるため，線形モデルを用いた個々のモジュールに対する動作周波数推定精度が直接実行時間の推定精度に影響するのに対して，CPU 個別制約時では求めるべきモジュール一律平均動作周波数がモジュール平均消費電力に依存し，その結果として，個々のモジュール消費電力の推定精度が直接的には影響にしにくい，という違いがあることが原因であると考えられる．

基にした消費電力情報の違い（実測値か PVT を用いた推定値か）による実行時間推定精度を見ると，その両者で推定精度に大きな差が見られないことが分かる．したがって，電力制約時の実行時間推定を PVT を用いた推定消費電力を基にした消費電力–動作周波数線形モデルと動作周波数–正規化実行時間の近似 3 点モデルという低コストで

作成可能なモデルを用いることで，電力制約時並列アプリ実行時間を精度良く推定できることが示された．

5. まとめと今後の課題

本研究では，電力制約型スーパーコンピュータを対象とした並列アプリ実行時間推定法を提案した．本手法では，小規模クラスタを用いてアプリ実行を行うことで消費電力–動作周波数および動作周波数–正規化実行時間に関する相関モデルを構築し，これらと非制約時の実行時間の実測値とを組合わせることで電力制約時の実行時間を推定する．1,920 モジュールを有する大規模並列実行を対象とした評価を行った結果，モデルに基づく簡便な推定手法であるにもかかわらず，平均誤差 10∼15% の精度で推定できることが明らかとなった．

今後の課題としては，まず，非電力制約時の大規模実行の実行時間を大規模実行を実際に行うことなく推定する手法の確立である．電力制約適応型スパコンで非電力制約下での大規模並列実行を行うことは，限られた供給電力制約下では困難であるため，これは重要な課題である．次に，動作周波数–正規化実行時間の精度向上である．現在は 4 つの動作周波数制約条件下での実測値を基に 2 つの反比例モデル（近似 3 点モデル）を構築しているが，図 6 で示したように依然として実測値に基づいた 3 点モデルとの差がある．ここで，例えば非制約時，最低動作周波数時の 2 つの動作周波数制約下での実行結果を用いて周波数領域を分割する動作周波数 f_3 を知ることができれば，動作周波数を f_3 に制約してアプリを実行するだけで，より精度の高い動作周波数–正規化実行時間に対する 3 点モデルを構築することができる．また，本研究では動作周波数–正規化実行時間相関モデルが計算サイズに依存しないと仮定していたが，計算サイズ依存性を考慮することで，さらに推定精度がよくなることが期待される．さらには，消費電力–動作周波数相関モデルの改良，あるいは，アプリを構成する関数（区間）ごとの実行時間を推定するなどの細粒度での推定法によっても，電力制約時の実行時間推定が改善されることが期待されるので，これらも今後検討すべき課題である．

謝辞　本研究は，一部，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」，ならびに，九州大学情報基盤研究開発センターの「先端的計算科学研究プロジェクト」の支援を受けている．

参考文献

[1] NASA Advanced Supercomputing Division, NAS Parallel Benchmark Suite v3.3. http://www.nas.nasa.gov/Resources/Software/npb.html.

[2] Ashby, S., Beckman, P., Chen, J., Colella, P., Collins, B., Crawford, D., Dongarra, J., Kothe, D., Lusk, R., Messina, P., Mezzacappa, T., Moin, P., Norman, M., Rosner, R., Sarkar, V., Siegel, A., Streitz, F., White, A. and Wright, M.: The Opportunities and Challenges of Exascale Computing, *Summary Report of the Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee* (2010).

[3] Barker, K. J., Davis, K., Hoisie, A., Kerbyson, D. J., Lang, M., Pakin, S. and Sancho, J. C.: Using Performance Modeling to Design Large-Scale Systems, *IEEE Computer, Vol.42, No.11* (2009).

[4] Bergman, K., Borkar, S., Campbell, D., Carlson, W., Dally, W., Denneau, M., Franzon, P., Harrod, W., Hiller, J., Karp, S., Keckler, S., Klein, D., Lucas, R., Richards, M., Scarpelli, A., Scott, S., Snavely, A., Sterling, T., Williams, R. S., Yelick, K., Bergman, K., Borkar, S., Campbell, D., Carlson, W., Dally, W., Denneau, M., Franzon, P., Harrod, W., Hiller, J., Keckler, S., Klein, D., Kogge, P., Williams, R. S. and Yelick, K.: ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems (2008).

[5] Borkar, S.: Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation, *Micro, IEEE*, Vol. 25, No. 6, pp. 10–16 (2005).

[6] Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, J. W., Lee, S.-H. and Skadron, K.: Rodinia: A Benchmark Suite for Heterogeneous Computing, *Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC)*, IISWC '09, pp. 44–54 (2009).

[7] Dighe, S., Vangal, S. R., Aseron, P., Kumar, S., Jacob, T., Bowman, K. A., Howard, J., Tschanz, J., Erraguntla, V., Borkar, N., De, V. K. and Borkar, S.: Within-Die Variation-Aware Dynamic-Voltage-Frequency-Scaling With Optimal Core Allocation and Thread Hopping for the 80-Core TeraFLOPS Processor, *IEEE Journal of Solid-State Circuits*, Vol. 46, No. 1, pp. 184–193 (2011).

[8] Fakazawa, K., Ogino, T. and Walker, R. J.: Configuration and dynamics of the Jovian magnetosphere, *Journal of Geophysical Research*, Vol. 111, p. A10207 (2006).

[9] Harriott, L. R.: Limits of lithography, *Proceedings of the IEEE*, Vol. 89, No. 3, pp. 366–374 (2001).

[10] Inadomi, Y., Patki, T., Inoue, K., Aoyagi, M., Rountree, B., Schulz, M., Lowenthal, D., Wada, Y., Fukazawa, K., Ueda, M., Kondo, M. and Miyoshi, I.: Analyzing and Mitigating the Impact of Manufacturing Variability in Power-Constrained Supercomputing, *Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis, Nov. 2015., Austin* (2015).

[11] Intel Corporation: *Intel 64 and IA–32 Architectures Software Developerś Manual Volume 3(3A, 3B & 3C): System Programming Guide* (2012).

[12] Luszczek, P., Bailey, D., Dongarra, J., Kepner, J., Lucas, R., Rabenseifner, R. and Takahash, D.: HPC Challenge Benchmark Suite. http://icl.cs.utk.edu/hpcc/index.html.

[13] Maruyama, N., Suzuki, S., Mikami, K., Komuro, Y., Takizawa, S. and Matsuda, M.: Fiber Miniapp Suite, fiber-miniapp.github.io.

[14] Ogino, T., Walker, R. J. and Ashour-Abdalla, M.: A

Global Magnetohydrodynamic Simulation of the Magnetopause when the Interplanetary Magnetic Field is Northward, *IEEE Transaction on Plasma Science*, Vol. 20, pp. 817–828 (1992).

[15] Petitet, A., Whaley, C., Dongarra, J. and Cleary, A.: High Performance Linpack. `http://www.netlib.org/benchmark/hpl/`.

[16] Sachs, S. R.: 2013 Exascale Operating and Runtime Systems, Technical report, Advanced Science Computing Research (ASCR) (2013). `http://science.doe.gov/grants/pdf/LAB13-02.pdf`.

[17] Susukita, R., Ando, H., Aoyagi, M., Honda, H., Inadomi, Y., Inoue, K., Ishizuki, S., Kimura, Y., Komatsu, H., Kurokawa, M., Murakami, K., Shibamura, H., Yamamura, S. and Yu, Y.: Performance Prediction of Large-scale Parallel System and Application using Macro-level Simulation, *International Conference for High Performance Computing, Networking, Storage and Analysis* (2008).

[18] Tahara, D. and Imada, M.: Variational Monte Carlo Method Combined with Quantum-number Projection and Multi-variable Optimization, *J. Phys. Soc. Jpn.*, Vol. 77, p. 114701 (2008).

[19] Tschanz, J., Kao, J., Narendra, S., Nair, R., Antoniadis, D., Chandrakasan, A. and De, V.: Adaptive Body Bias for Reducing Impacts of Die-to-die and Within-die Parameter Variations on Microprocessor Frequency and Leakage, *Solid-State Circuits, IEEE Journal of*, Vol. 37, No. 11, pp. 1396–1402 (2002).

[20] Wijngaart, R. F. V. D. and Jin, H.: NAS Parallel Benchmarks, Multi-Zone Versions, Technical report, NASA Advanced Supercomputing (NAS) Division, NASA Ames Research Center (2003).