社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－ARC－167 (24)
2006－HPC－105 (24)
2006／2／28

PCクラスタにおける電力実行プロファイル情報を用いたDVS制御による電力性能の最適化

堀田義彦† 佐藤三久† 木村英明†
松岡聡†† 朴泰祐† 高橋大介†

本論文では，高性能PCクラスタにおけるプロファイル情報を用いたDVSスケジューリングによる電力性能最適化手法を提案する．近年，従来低消費電力向けプロセッサに実装されていた，消費電力を削減するためにプロセッサの動作周波数・電圧を動的に変更するDVS(Dynamic Voltage Scaling)が高性能プロセッサにも実装されている．性能低下を最小限にし，消費電力を削減するために通信やメモリアクセスの際に適切な周波数スケジューリングを行う必要がある．電力性能を最適化するために，プログラムをいくつかの領域に分割し，領域ごとに適切な周波数を選択する．DVSによる周波数変更は，オーバーヘッドを発生するため，これを加味した周波数選択アルゴリズムを提案する．システムの詳細な電力消費特性を測定するため，電力測定環境を構築した．このシステムにより，異なるプロセッサを使用する2つのクラスタで提案するアルゴリズムの有効性能の評価を行った．その結果，標準の周波数で動作するときと比べ，最大40%のをEDP(ED積)を5%以下の性能低下でできることがわかった．

Profile-based Optimization of Power Performance by using Dynamic Voltage Scaling on a PC cluster

Yoshihiko Hotta,† Mitsuhisa Sato,† Hideaki Kimura,†
Satoshi Matsuoka,†† Taisuke Boku† and Daisuke Takahashi †

Currently, several of the high performance processors used in a PC cluster have a DVS (Dynamic Voltage Scaling) architecture that can dynamically scale processor voltage and frequency. Adaptive scheduling of the voltage and frequency enables us to reduce power dissipation without a performance slowdown during communication and memory access. In this paper, we propose a method of profiled-based power-performance optimization by DVFS scheduling in a high-performance PC cluster. We divide the program execution into several regions and select the best gear for power efficiency. We propose an optimization algorithm to select a gear using the execution and power profile by taking the transition overhead into account. We have built and designed a power-profiling system, PowerWatch. With this system we examined the effectiveness of our optimization algorithm on two types of power-scalable clusters (Crusoe and Turion). According to the results of benchmark tests, we achieved almost 40% reduction in terms of EDP (energy-delay product) without performance impact (less than 5%) compared to results using the standard clock frequency.

1. はじめに

近年の計算機システムにおける消費電力の爆発的な上昇は地球シミュレータやASCIシリーズのような巨大なシステムの設計において非常に大きな問題として認識された．この問題のひとつの解決方法は，低消費電力プロセッサを用いることである．BlueGene/L[1]では低消費電力なコンポーネントを使用することでシステム全体の消費電力を削減し，高密度実装による高性能化を実現している．また，Fengらによる Green Destiny[6]は低消費電力プロセッサを高密度に実装することで高性能を実現し，低消費電力化の必要性を提言した最初のクラスタである．我々はTransmetaのEfficeonプロセッサを使用し数千以上のプロセッサで構成されるmega scale computingを実現するためのプラットホームとしてMegaProtoを開発している[5]．

一方で，消費電力を削減する仕組みとしてプロセッサの電圧・周波数を動的に変化させることで消費電力を下げるDVS(Dynamic Voltage Scaling)があり，現在様々なプロセッサに実装されている．DVSによって周波数・電圧を変更することができるようになり，アプリケーション実行において，高い周波数での動作や定格周波数よりも低い周波数や低電圧で動作する方が性能低下が少なく，電力効率が高い場合があることがわかってきた[7]．例えば通信の場合は，周波数を下げて実行することで，電力の削減をすることが考えられる．通信に関してはプロセッサの速度との差が，近年顕著になってきており，並列アプリケーションの通信

† 筑波大学大学院システム情報工学研究科
Graduate School of Information and Sciences Engeneering,University of Tsukuba
†† 東京工業大学
Tokyo Institute of Technology

時間の比率も高くなっている．この待ち時間に高い周波数でプロセッサが動作することで無駄に電力を消費していると考えられる．また，通信時に周波数を下げることは，計算時に周波数を下げるよりもアプリケーションの性能低下を抑えることができる．

DVS をサポートするプロセッサには周波数と電圧の組合せが定められており，本論文ではこれを *gear* と呼ぶ．本論文では，我々はプログラムの実行をいくつかの領域に分け，実行時間と消費電力のプロファイル情報を元にそれぞれの領域を実行する際的な *gear* を選択する方法を提案する．この手法では，様々な周波数でアプリケーションを実行し，そのときのデータを評価することでフェーズごとの最適な周波数を決定する．異なる動作周波数へ変更するにはオーバーヘッドが発生するため，各領域の周波数は単純に決めることができない．別な *gear* へ移行するオーバーヘッドは数十 μ 秒であり，DVS を使用して消費電力を削減する効果を打ち消してしまう可能性がある．我々が提案するアルゴリズムで動作周波数の変更に必要なオーバーヘッドを考慮した電力性能最適化を行う．領域にどのように分けるかは重要な問題である．本論文では，通信部分に着目して領域を分けた．通信以外の部分に関してはメインループの関数レベルで領域を分けた．これらの領域のプロファイルを取得し我々の提案するアルゴリズムに適用し，評価を行った．

本論文では，評価指標として電力量と EDP(電力遅延積) の 2 つを用いて，評価を行う．電力量は消費するエネルギー量であり，この値を削減することで省電力化が実現できる．しかし，電力量が削減されても，上記のように性能が大幅に低下することを避けなければならない．そこで，電力量とは異なる EDP での評価を行った．EDP は電力量と異なり，性能を考慮した指標であり，この値を最適化することによって性能と電力量の両方を考慮した評価を行った．

本論文では，まず 2 種類のクラスタにおける各 *gear* で並列アプリケーションを実行したときの電力と性能の特性について述べる．評価にあたり，我々は消費電力などを測定する環境を構築した．検証のためのプラットフォームとして低消費電力プロセッサである Turion と Crusoe を使用する 2 つのクラスタを構築し，評価を行った．これにより，HPC 向け power-scalable cluster におけるプロファイル情報を利用した動作周波数最適化の有効性について検証する．

本論文の構成は以下のようになっている．2 章では関連する研究について述べる．3 章ではプロファイル情報を用いた電力性能を最適にするための周波数選択アルゴリズムについて述べる．4 章では各クラスタの特性評価ならびに周波数スケジューリングの効果について述べる．最後にまとめを述べる．

2. 関連研究

プロファイル情報を元に動作周波数を最適化することでエネルギーや EDP を削減をする研究が行われている．V.Freeh[3] は，プログラムが 1 回のキャッシュミス当たりの命令数 (OPM:*operations per miss*) に注目した DVS の最適化によりオフチップアクセスの特性を評価することで周波数をプログラム中で変更する。周波数を選択するために，アプリケーションを 1 度実行し，プロファイルとして OPM の値を取得し，その特性により領域を分割し最適な周波数を選択する．これにより、固定する場合と比べて約 9%のエネルギー削減を実現している．

Rong[2] らは，プロファイルに MPICH に付属している MPE ツールを使用し，通信に着目したプロファイル情報を取得し，通信時に動作周波数を下げることでエネルギーの削減を行っている．また，エネルギーだけでなく EDP やさらに性能を重みとして付加した ED2P おいても削減を実現しており，プロファイル情報を基にした最適化の有効性を示している．

3. 最適周波数選択アルゴリズム

3.1 電力性能の指標

HPC における性能評価には FLOPS のような様々な指標がある．D.Brooks ら[4] は，今後の HPC における評価指標として，性能のみに注目した FLOPS に代わる性能だけでなく消費電力を加味した PDP(Power Delay Product) や EDP(Energy Delay Product) を電力性能として使用することを提案している．

本論文では PDP，EDP の両方の指標で評価を行う．PDP は単位時間の消費電力と実行時間の積分によって求められ，アプリケーションを実行するのに必要なエネルギーの量を表す．PDP はノート PC や携帯機器などバッテリ駆動において最も重要である電力効率を評価するのに適している．

しかし，高性能システムにおいて，PDP を削減しても性能が極端に低下することは避けなければならない．我々の対象は高性能かつ低消費電力なクラスタであり，このことをふまえて評価するには PDP は十分ではなく，性能を加味した指標である EDP を使用することにした．これは PDP だけでなく，実行時間で重みをつけた指標であり，以下のように示される．

$$EDP = T_{exectime} \times Energy \qquad (1)$$

この指標において，EDP の値が低いことは優れた電力性能でプログラムを実行することを意味する．

3.2 電力性能最適化アルゴリズム

ここでは，提案する電力性能最適化アルゴリズムについて説明する．まずプログラム P をいくつかの領域にわけ，この領域を R_i とする．それぞれの領域はプログラム中に適切な部分にプロファイル取得コードを挿入することで定義する．図 2 に電力性能最適化の流れを示す．以下にこのシステムの各処理が行う作業を示す．

(1) プロファイル取得のための試行
プロファイル情報を付加したアプリケーションを様々な周波数で実行することで，各フェーズの実行時間のプロファイルを取得する．また，同時に消費電力の測定を行い，電力プロファイルを取得する．

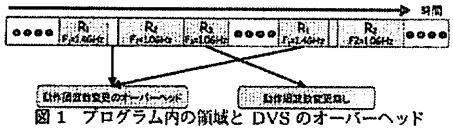


図 1 プログラム内の領域と DVS のオーバーヘッド

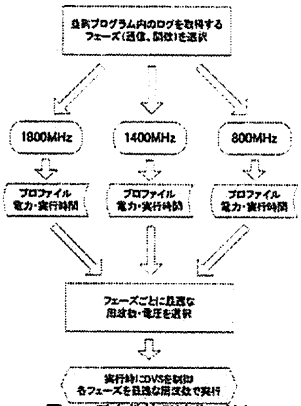


図 2 電力性能最適化の流れ

(2) PDP・EDP の評価
各 *gear* でプログラムを実行することで，それぞれの *gear* での各領域の EDP 値を求める．

(3) 周波数スケジューリングの選択
各領域を実行するのに最適な動作周波数である F_i を後述のアルゴリズムを適用することで求める．

(4) 周波数スケジューリングの適用
求められた最適な動作周波数を実際の実行時に適用する．プログラム内の各領域の直前に DVS のシステムコールを挿入し，求めた周波数で実行する．

我々は，*gear* の組み合わせを決定するために評価関数を以下の式のように定義する：(R_i (i = 1 ···))

$$E(P) = \sum_{i=0}^{n} (E(R_i, f_i) + E_{trans}(R_i, f_i))$$

評価関数は EDP でも PDP でも良いが，EDP として説明する．この式において，$E(R_i, f_i)$ は領域 R_i を動作周波数 f_i で実行するときの EDP の総和である．f_i はプロセッサが変更することができる *gear* の周波数である．$E_{trans}(R_i, f_i)$ は *gear* を動作周波数 f_i で実行するときの周波数変更に必要なオーバーヘッドの際の EDP の総和を表している．このオーバーヘッドは R_i と隣接する領域の動作周波数に依存する．図 1 にそれぞれの領域のプログラムにおける実行の流れを示す．隣接する領域が同じ周波数を選択した場合，周波数を変更する必要はないためオーバーヘッドは 0 になる．異なる周波数を選択した場合，オーバーヘッドを加えて評価式を計算しなくてはならない．E_{trans} は周波数変更に必要な時間とそのときの消費電力の積から求める．目的は $E(P)$ を最小にする f_i の組み合わせを求めることである。決定した f_i をここでは F_i と表すことにする。プログラム上の領域の先頭で、周波数を制御するため同一の領域は同じ周波数で実行されるものとする。$E(P)$ を最適化するために，我々は EDP が最も大きくなる領域から周波数を決定する．これは，EDP が最大となる領域は，全体の EDP を削減する際に最も寄与の大きい領域であるためである．以下にアルゴリズムを示す．

- Step 1: 各領域 R_i の動作周波数 F_i を**未定義**とする．F_i は領域 R_i を実行する動作周波数である．
- Step 2: 全ての $E(R_i, f_i)$ を予備実行から得られたプロファイル情報を元に求める．
- Step 3: F_i が**未定義**である領域から $E(R_i, f_{max})$ が最大となる領域 R_i 選択する．全ての F_i が決まっている場合，アルゴリズムを停止し結果を出力する
- Step 4: 領域 R_i を動作周波数 f_i で実行する時の $E(P)$ を算出する $E(P)$ を算出する際に，領域の動作周波数が決まっていない場合は，EDP とオーバーヘッドは 0 として計算する．隣接する領域の動作周波数がすでに決まっており，現在の領域 R_i と動作周波数が異なる場合，動作周波数の変更に必要なオーバーヘッドを加えて計算する．$E(P)$ を最小にする最適な f_i を F_i とし，領域を**定義済み**にセットする F_i は領域 R_i を実行する動作周波数である．
- 全ての F_i が**定義済み**になるまで step3，4 を繰り返す．

3.3 領域の分割方法について

どのように領域を分割するかが本手法において最も重要な問題である．本研究で使用するプロファイラでは任意の部分 (関数やループ) にプロファイル取得コードを挿入することができる．しかし，分割の仕方によってアプリケーションの特性を正しく反映できな

表 1 クラスタの仕様

システム名	Turion クラスタ	Crusoe クラスタ
CPU	Turion MT-32	TM-5800
Clock	1.80GHz	933MHz
Cache L1/L2	64KB/1MB	64KB/512KB
Memory	1GB(DDR)	256MB(SDR)
kernel	2.6.11	2.6.11
Compiler	gcc3.4.11	gcc3.4.11
MPI	LAM 7.1.1	LAM 7.1.1
num of nodes	8	4
Network	Gigabit Ethernet	Fast Ethernet
TDP	24W	9W

表 2 Crusoe の周波数と電圧

gear	周波数	電圧	FSB	TDP
1	933MHz	1.35V	133MHz	9W
2	800MHz	1.25V	133MHz	7W
3	667MHz	1.20V	133MHZ	5W
4	533MHz	1.10V	133MHz	4W
5	300MHz	0.90V	100MHz	3W

表 3 Turion の周波数と電圧

gear	周波数	電圧	TDP
1	800MHz	0.90V	9W
2	1000MHz	1.00V	-
3	1200MHz	1.05V	-
4	1400MHz	1.10V	-
5	1600MHz	1.15V	-
6	1800MHz	1.20	25W

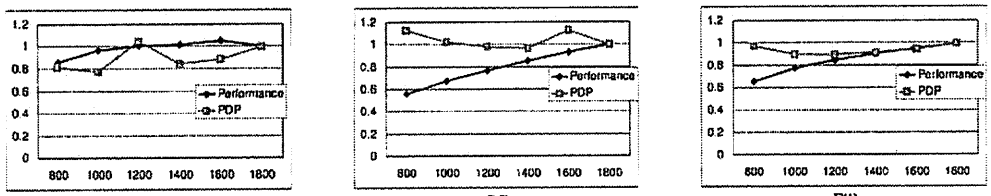


図 3 Turion クラスタの各 *gear* における性能・電力特性 (一部推測)

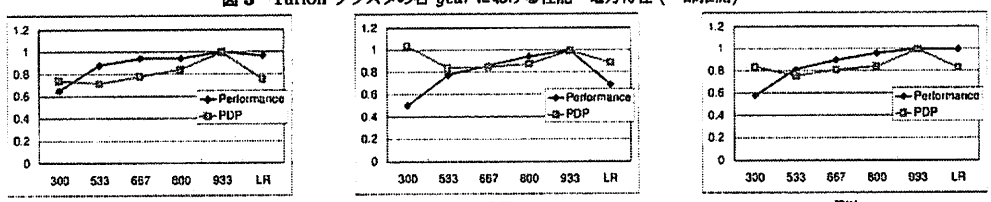


図 4 Crusoe クラスタの各 *gear* 時の性能・電力特性

表 4 各ベンチマークの領域

	領域 1	領域 2	領域 3	領域 4
IS	rank	MPI_Alltoall	MPI_Alltoallv	MPI_Allreduce
FT	fft	evolve	MPI_Allreduce	checksum
CG	conj_grad	MPI_Send	MPI_Wait	MPI_Irecv

い場合がある．例として，最粒度に分割しすぎることで，プロファイル取得のオーバーヘッドが大きくなり，アプリケーションの性能が低下する．また，適切な分割を行わなかった場合，本来周波数を変更すべき状況においても間違った周波数を選択する可能性がある．

3.4 実行時間プロファイルツール Tlog

tlog とは *time log* の略称であり，イベントごとの log を取るライブラリと可視化を行うツールである tlogview から構成されている．プロファイルを取得するために，プロファイル取得コードをプログラム内の適当な場所 (例えば MPI_Send など) を挟みこむことで，tlog 初期化からの開始時点の経過時間と終了時点の経過時間を取得することができる．プロファイルから得られた時間と電力プロファイルと対応づけることでフェーズの PDP・EDP を取得することができる．

4. 評　　価

4.1 クラスタ環境

手法の有効性を検証するために 2 つの powerscalabe なクラスタにおいて評価を行った．我々はプロセッサとして Turion と Crusoe を選択した．Turion は Pentium-M と同様にノート PC 向けとして開発された低消費電力かつ高性能なプロセッサである．Crusoe は DVFS をサポートした最初の IA-32 互換のプロセッサであり LongRun と呼ばれる DVFS の自動制御機構を備えている．LongRun により周波数と電圧を CPU の状況に応じて変化させることができる．

表 1 にクラスタの仕様を示す．OS は Linux を使用しどちらの環境においても同じバージョンの kernel，コンパイラ，ライブラリなどを使用した．表 2 に Crusoe の表 3 に Turion の周波数と電圧の組み合わせを示す

提案するアルゴリズムを使用するには，DVS のオーバーヘッドを定める必要がある．Turion では、停止時間を 50μ 秒以上にすることが可能である．

予備評価として Turion で周波数変更に必要なオーバーヘッド (プログラムが動作可能になるまでの時間) を DVS のランタイムの前後の時間を測定することで求めた．その結果，平均して 30μ 秒の時間を必要としたため，最適化においてオーバーヘッドを 50μ 秒とした．Crusoe に関しては，このような停止時間をセットすることができないため，同じ値をオーバーヘッドとして使用した．

4.2 使用したベンチマークとクラスタの特性

評価には NPB(NAS Parallel Benchmarks) のバージョン 3.1 を使用した．実行するベンチマークは IS, FT, CG である．CLASS は Turion クラスタは C, Crusoe クラスタは A を使用した．まず，特性を知るために各ベンチマークにおける *gear* 毎の性能と PDP を計測した．全ての評価は複数回実行し，その中の最大値を使用している．図 3 に Turion クラスタの図 4 に Crusoe クラスタの標準の周波数を基準とした性能と PDP 消費の特性を示す．これ以降 Crusoe に関する図において LongRun を LR と表記する．それぞれのクラスタについて以下の特性がわかった．

Turion クラスタ：低い *gear* のときに，いくつかのベンチマークで PDP 消費を削減することができる．CG では低い *gear* のときに性能が大きく低下するが，IS や FT では性能低下が少ない．

Crusoe クラスタ：LongRun 使用時よりも低い周波数において PDP 消費を削減することができることがわかる．最も低い周波数である 300MHz において，大幅に性能が低下している．これは，動作周波数が下がるだけでなく FSB の周波数も 133MHz から 100MHz へと低下していることが原因である．CG において LongRun 使用時の性能が著しく低下している．我々の過去の研究において，LongRun を使用する場

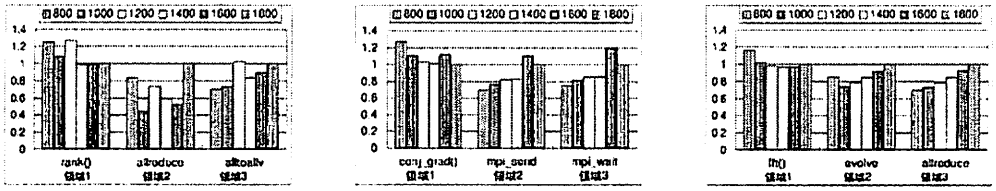


図 5 Turion クラスタの各 *gear* による領域ごとの PDP 比較

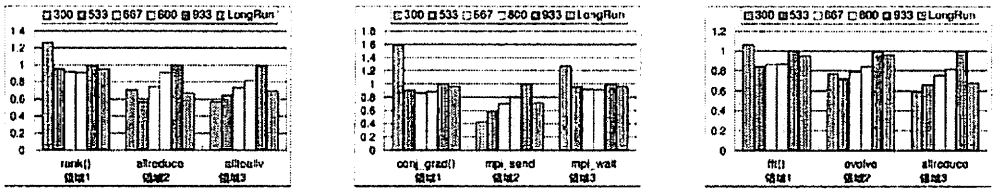


図 6 Crusoe クラスタの各 *gear* による領域ごとの PDP の比較

表 5 Turion クラスタのアルゴリズムが決定した周波数 [MHz]

		領域 1	領域 2	領域 3
IS	PDP	1800	1000	800
	EDP	1800	1000	1000
FT	PDP	1400	1400	1000
	EDP	1800	1800	1000
CG	PDP	1000	1600	1400
	EDP	1800	1800	1800

表 6 Crusoe クラスタのアルゴリズムが決定した周波数 [MHz]

		領域 1	領域 2	領域 3
IS	PDP	933	533	300
	EDP	933	533	300
FT	PDP	533	800	533
	EDP	800	533	300
CG	PDP	533	533	800
	EDP	800	800	800

合に同様の症状が見られており，細かい通信が頻繁に発生する場合，LongRun が適切に動作していないことが明らかになっている[5]．

4.3 領域の分割方法の方針

領域の決定指針としていくつかの戦略がある．領域の決定指針の中で最も重要なのは，通信に着目することである．並列プログラムでは，実行時間の多くを通信に費やしており，このときに動作周波数を低くすることで，無駄な PDP の消費を抑えることができる．領域を決めるにあたって，まず各ベンチマークの主要な通信にプロファイラーを付加した．さらに，通信の中でも全体全通信 (MPI_Alltoall など) に特に着目した．全体全通信は全てのノードが通信を行うために，動作周波数を下げることで大幅な電力の削減が期待できる．また，今回用いるベンチマークはループを繰り返すアプリケーションであり，メインループにある通信も領域とした．

通信以外の戦略としては，メモリアクセスや計算量が多い領域に着目することである．オフチップのアクセスが必要な場合，待ち時間が生じるため低い周波数での動作によって性能低下を最小限に抑え，消費電力の削減が期待できる．実際にオフチップアクセスを考慮して DVS を最適化することで消費電力の削減ができることが報告されている．しかし，オフチップアクセスするかどうかを判断するのには，詳細なプログラムの解析が必要であるため，本研究では計算部分を関数レベルで領域を定めることで低い周波数で電力消費の削減ができるかどうかを検討した．

表 4 に各アプリケーションにおいて選択した領域を示す．これ以降の表・図において領域を示すものはこの表と対応する．

各クラスタにおける領域の特性を以下に示す．ここでは，実行時間の長かった上位 3 つの領域の特性を示す．

Turion クラスタ:

図 5 に標準の動作周波数の場合を基準とした，動作周波数別の各領域の PDP を示す．

- 通信の領域で低い周波数を用いることで PDP を削減できる
- IS や FT のいくつかの領域で PDP の削減が可能
- fft() では周波数を変更しても PDP の削減はない

Crusoe クラスタ:

図 6 に標準の動作周波数の場合を基準とした，動作周波数別の各領域の PDP を示す．

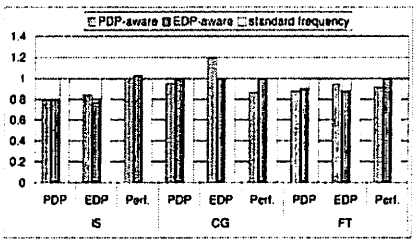
- 通信の領域では，低い動作周波数で大幅に PDP の削減が可能
- 全体的に Turion クラスタよりも PDP の削減の幅が大きい
- LongRun は概ね PDP 削減ができている

4.4 評価結果

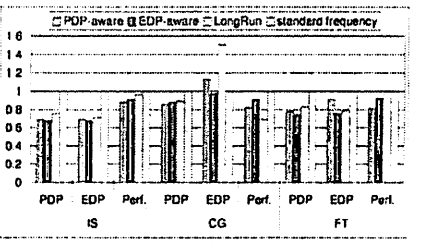
Turion クラスタでの評価結果

表 5 にアルゴリズムで求めた主要な領域の周波数を示す．評価にあたり EDP だけでなく PDP の 2 つの値をそれぞれ最適化を行った．PDP 最適化の場合は，いくつかの領域で周波数を下げている．特に通信の領域では周波数を下げる．EDP 最適化の場合は，CG や MG では，周波数を変更しないという結果になった．

図 7(左) に Turion クラスタにおける標準の周波数である 1800MHz を基準とした場合の PDP ならびに EDP 最適化における結果を示す全てのベンチマークにおいて，PDP や EDP の削減が実現しており，特に IS と FT においては約 20%の削減を実現している．



Turion クラスタ



Crusoe クラスタ

図 7 最適化適用時と標準の周波数における性能，PDP，EDP の比較 (基準は標準周波数)

PDP 最適化が性能低下を引き起こしているのに対して，EDP 最適化では性能低下が非常に少ない．結果として EDP 最適化が PDP と EDP の両方を効果的に削減している．

Crusoe クラスタでの評価結果

表 6 にアルゴリズムが導き出した主要な領域の周波数を示す．各領域の対応は Turion クラスタと同じである。PDP 最適化において，いくつかの領域で周波数を下げている EDP 最適化の場合は，CG と MG で周波数を固定するという結果となったが，Turion と異なり最高動作周波数での固定ではなかった．IS では主要な領域における周波数は PDP と EDP 両方の最適化で同じ周波数を選択した．

図 7(右) に EDP，PDP 両方の最適化を適用したときと標準の周波数で実行したときとの PDP，EDP，性能の比を示す．全体的に Turion クラスタよりも PDP,EDP の削減の効果が高い．主な理由として，ネットワークインタフェースが Turion クラスタとは異なり Fast Ethernet であるため，通信時間がアプリケーションに占める割合が大きくなり，通信時間に動作周波数を下げ，消費電力を削減している時間が長くなるために全体として削減の効果が高くなる．結果として，IS では PDP を約 40%，他のベンチマークでも約 20%程度の削減を実現した．一方で，PDP 最適化は性能低下を引き起こしている．

5. おわりに

本論文では，HPC クラスタにおいてプロファイル情報を用いた DVS による電力のの最適化の検討を行った．プログラムをいくつかの領域にわけ，さまざまな周波数で予備実行を行うことで各領域の電力，実行プロファイルを取得することで，それぞれの領域を最適に実行する動作周波数を求めた．

我々が提案した周波数選択アルゴリズムは，高い電力性能を実現するために DVFS のオーバーヘッドも考慮する．また，評価においては PDP 削減を行う PDP 最適化だけでなく，性能に比重を置いた EDP 最適化も行った．その結果，両方のクラスタにおいて大幅な PDP，EDP の削減を実現した．特に Crusoe クラスタにおいて，一部のベンチマークで 40%の EDP の削減をすることができることがわかった．

現在，領域を分割するプロファイル取得コードを人手で挿入している．今後の課題として，領域をコンパイラなどで自動的に分割する仕組みの検討が必要である．本論文では，通信以外は関数レベルでの領域分割を行った．アプリケーションをより詳細に解析することで，電力性能をより高くすることが可能である．Hsu ら[8] は，静的にプログラムを解析し，DVS を制御することで PDP の大幅な削減をわずかな性能低下で実現している．本手法でも，このようなコンパイラによる静的な解析を行い領域を分割することで更なる効果が期待できる．

謝辞 様々な御助言をいただいた CREST チームの方々に感謝します．本研究の一部は，科学技術振興機構・戦略的創造研究「低消費電力化とモデリング技術によるメガスケールコンピューティング」による．

参考文献

1) N. Adiga, G. Almasi, G. Almasi et.al. An overview of the bluegene/l supercomputer. In *Supercomputing Conference 05 (SC'05)*, November 2005.
2) R. Ge, X. Feng, and K. W.Cameron. Performance-constrained distributed dvs scheduling for scientific applications on power-aware clusters. In *Supercomputing Conference 05*, November 2005.
3) V.Freeh, F.Pan, N.Kappiah, and D.K.Lowenthal. Using Multiple Energy Gears in MPI Programs on a Power-Scalable Cluster. In *Symposium on Principles and Practice of Parallel Programming PPoPP'05*, July 2005.
4) D. Brooks, P.Bose, S. Schuster et.al. Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. In *IEEE Micro*, volume 20, pages 26–44, 2000.
5) H.Nakashima, H.Nakamura, M.Sato, T.Boku, S.Matsuoka, D.Takahashi, Y.Hotta. Megaproto: 1 TFlops/10kw rack is feasible even with only commodity technology. In *Supercomputing Conference 05*, November 2005.
6) M.Warren, E.Weigle, and W.Feng. High-density computing: A 240-processor beowulf in one cubic meter. In *Supercomputing Conference 02*, November 2002.
7) V.Freeh, F.Pan, N.Kappiah, D.K.Lowenthal, and R.Springer. Exploring the Energy-Time Tradeoff in MPI Programs on a Power-Scalable Cluster. In *16th International Parallel and Distributed Processing Symposium (IPDPS'05)* , April 2005.
8) C-H. Hsu and U.Kremer. The design, implementation, and evaluation of a compiler algorithm for cpu energy reduction In *ACM SIGPLAN Conference on Programming Languages, Design, and Implementation (PLDI'03)*, June 2003.