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，電力測定環境である PowerWatch 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 30%以上の EDP（電力遅延積）を 5%以下の性能低下でできることが分かった．

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 DVS scheduling in a high-performance PC cluster. We divide the program execution into several regions and select the best *gear* (combinations of clock frequency and voltage) for power efficiency. Selecting the best *gear* is not straightforward since the overhead of DVFS transition is not free. 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 30% 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，PC クラスタなどの高性能計算機システムにおいてプロセッサの消費電力の上昇による消費電力の増加が問題となっている．プロセッサの消費電力は100 W を超え，冷却のために大きな装置が必要となりシステムのコンパクト化の大きな障害となっている．また，高い消費電力のためにプロセッサの高密度実装も困難になっている．

この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 1 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

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

減し，高密度実装による高性能化を実現している．また，Warren らによる Green Destiny [9] 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 [6]．

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分かってきた[11]．たとえば通信の場合は，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．

本論文の貢献は以下のとおりである．

- プログラムを領域に分け，各領域のプロファイルを取得することにより DVS の周波数変更のオーバヘッドを考慮した周波数選択アルゴリズムの提案．
- 消費電力測定環境を構築し，実際のクラスタシステムでアルゴリズムの有効性を評価した．

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．この手法では，プロセッサが使用可能なすべての *gear* 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各領域を実行するの最適な *gear* を決定する．異なる動作周波数へ変更するにはオーバヘッドが発生するため，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．そこで，我々はこのオーバヘッドを考慮した周波数選択アルゴリズムを提案する．領域の分割方法に関しては，今回は通信とそれ以外に着目して領域の分割を行う．領域の分割方法は，重要な問題であるが，本論文では，領域に分けた際の最適な周波数選択アルゴリズムの提案であるため，ここでは議論の範囲としない．また，我々が提案するアルゴリズムでは，領域の分割がどのように行われても，その分割方法に最適な周波数を選択する．

DVS をプロセッサが状況に応じて自動的に制御する仕組みとして Transmeta の LongRun がある[4]．LongRun は，元々ノート PC のような環境で使用されることを目的に動作しており，高性能計算にとって必ずしも最適な周波数を選択しておらず，性能低下や電力量が増えることがある[6]．そのために，つねに性能・電力を最適にするためには，アプリケーションの状況に応じた周波数のスケジューリングが必要となる．そこで，本論文ではアプリケーションをいくつかの領域に分け，領域ごとに最適な周波数で動作させることで電力効率を向上させる．

本論文では，評価指標として PDP と EDP（電力遅延積）の 2 つを用いて，評価を行う．PDP 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，PDP が削減されても，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，PDP とは異なる EDP での評価を行った．EDP は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．

評価にあたり，我々は消費電力などを測定する環境である PowerWatch を構築した．このシステムはホール素子を使用した消費電力測定環境，実行プロファイルを取得するライブラリ，DVS をコントロールするライブラリを備える環境である．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をあげる．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 章では開発した電力測定環境について述べる．5 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．6 章では考察と今後の検討課題について述べる．最後にまとめを述べる．

2. 関 連 研 究

静的な解析による DVS の最適化

DVS を用いて実行中に周波数を制御する研究には様々なものがある．Hsu [12] らはコンパイラによってアプリケーションプログラムを静的に解析し，逐次プ

ログラム内で動作周波数を変更することで大幅なエネルギーの削減を実現している．我々は並列アプリケーションにおいて周波数の最適化を行うためにアプリケーションの動的な振舞いを考慮して，実際の消費電力を基に最適化を行うために，プロファイル情報を利用した最適化を行う．

プロファイル情報を用いた DVS 最適化

HPC クラスタにおける DVS の最適化として様々な研究が行われており，特に近年ではプロファイル情報を基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．

Ge[5] らは，プロファイラに MPICH に付属している MPE ツールを使用しており，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 に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．この手法は本研究のアプローチと同様であるが，この手法では，通信ライブラリのプロファイルを用いており任意の領域分割を考慮していない．また，彼らは通信のみに着目しているが，通信に必要な時間は必ずしも DVS のオーバヘッドと比較して大きいものではなく，DVS による消費電力削減の効果を打ち消してしまう可能性がある．そこで，本研究では DVS のオーバヘッドを考慮した周波数選択アルゴリズムを提案する．

Freeh ら[10] は，プログラムが 1 回のキャッシュミスあたりの命令数（OPM: *operations per miss*）に注目した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．これにより，周波数を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．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，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．この手法では，アプリケーションの詳細な解析を静的に行うことなく，最適な周波数選択を行う．しかし，この手法では，OPM の値を基に領域を分割し，周波数を決定するため，実際の消費電力がどのようになるかは考慮されていない．本研究では，実際の消費電力の測定を基にすることでより確かな周波数選択を行うことができる．

DVS の動的最適化

slack time に注目した DVS の最適化として，Kappiah らは DVS を最適化する仕組みとして Jitter（Just in time）を開発しエネルギーの削減をわずかな性能低下で実現している[8]．Jitter は同じイタレーションを繰り返すアプリケーションに注目し，イタレーションごとでの特性の違いがないことを利用し，特性を取得しながら状況に応じて周波数を決め，消費電力の削減を行っている．この研究では，対象をイタレイティブなアプリケーションとしており，その適用範囲は限られる．本研究では，プロファイルを用いることでどのようなアプリケーションでも実行時の振舞いを取得し，最適化を行うことができる．

Hsu[7] らはランタイムによる DVS の制御を行い，様々なアプリケーションで大幅なエネルギーの削減を実現している．この研究では，次のある一定の期間の周波数を決めるために，過去の様々な情報を統計的に処理することで適切な周波数を決める．このため，動作周波数が大きく変動することはなく，粗粒度で滑らかに変動する．これにより，アプリケーションに依存することなく周波数の最適化を行うことができる．ランタイムによる実装により，ユーザに意識させることなく最適化が可能であるが，粗粒度でスケジューリングを行うため，HPC アプリケーションにおいて必ずしも最適な周波数を選択しているとは限らない．そこで本研究では，プロファイル情報を用い，細粒度に周波数スケジューリングを行うことにより，より大きな効果が期待できる．

これらの研究においては，並列アプリケーションにおいて，通信時に周波数を下げることが行われているが，DVS のオーバヘッドが考慮されておらず，状況によっては不必要な周波数変更を行う場合がある．本研究では，これを解決するためにオーバヘッドを考慮したスケジューリングアルゴリズムを提案する．

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．Brooks ら[3] 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し，その単位を Ws とする．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 だけでは十分ではなく，性能を加味した指標での評価が必要である．そこで，電力遅延積である EDP を使用することにした．EDP は PDP より，実行時間で重みを付けた指標であり，HPC クラスタにおけるエネルギー効率を評価するために使用されている．本論文において，EDP を以下のように定義する．

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

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．本研究では，まず EDP を最適化することに焦点をあてる．さらに，EDP が同一な場合は，PDP が少なくなるほうが，エネルギーの削減になるため PDP についても評価を行う．

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の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．プロファイル取得コードを加えたプログラムを各 *gear* で実行し，各 *gear* におけるデータを得る．あるプログラム P の EDP である E(P) を最適化するために，プログラム中の n 個の各領域を実行する *gear* の組合せを決める．**図 1** 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．

(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* の組合せを決定するために評価関数を

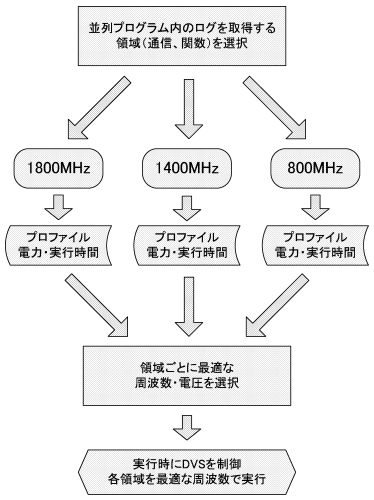


図 1 電力性能最適化の流れ
Fig. 1 Flow of profile-based power-performance optimization.

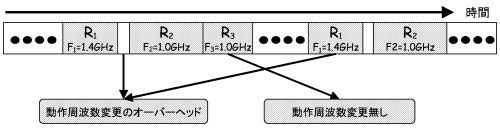


図 2 プログラム内の領域と DVS のオーバヘッド
Fig. 2 Regions and overhead of a DVS transition.

以下の式のように定義する：(R_i (i = 1…))

$$E(P) = \sum_{i=1}^{n} (E(R_i, f_i) + E_{trans}(R_i, f_i))$$

評価関数は EDP でも PDP でもよいが，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ヘッドの際の EDP の総和を表している．このオーバヘッドの有無は領域 R_i と隣接する領域の動作周波数に依存する．**図 2** 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ヘッドは 0 になる．異なる周波数を選択した場合，オーバ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 の組合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

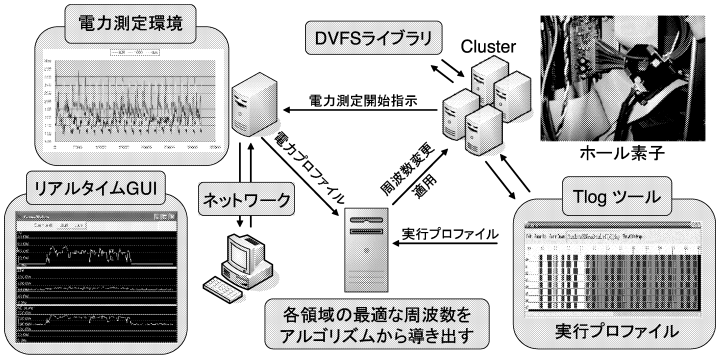


図 3 PowerWatch：電力測定システム
Fig. 3 PowerWatch: Overview of the measurement system.

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：すべての $E(R_i, f_i)$ を予備実行から得られたプロファイル情報を基に求める．
- 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．すべての 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 で実行するときの $E(P)$ を算出する．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．
- すべての 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重要な問題である．本論文で使用するプロファイラである tlog 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，領域を自由に定義することができる．一方で，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い場合がある．例として，細かく領域を分割しすぎることで，プロファイル取得のオーバヘッドが大きくなり，アプリケーションの性能が低下し，正確な予測を行うことができない．しかし，本論文は任意の領域に分割が行われたときに，それを最適に実行するための周波数選択を行うことを目的としている．たとえば，細かい通信が発生する場合などは，周波数変更のオーバヘッドを考慮し，変更を行わない場合もある．したがって，本論文では，領域の分割方法については議論の対象とはしない．

4. 電力測定環境 PowerWatch

消費電力と実行時間のプロファイルを取得するためのシステム PowerWatch を開発した．このシステムによりクラスタの各ノードの電力プロファイルを測定し収集することができる．**図 3** にこのシステムの概念図を示す．このシステムは以下に示す 3 つの特徴がある

4.1 ホール素子を使用した電力モニタリング

測定対象を破壊することなく数十 μ 秒の解像度で電力の測定を可能にする．現在のシステムは，48 個の電力線の消費電力を同時に測定でき，これにより 8 ノードすべての詳細な消費電力の測定を同時に行うことができる．アプリケーションプログラムに組み込むための API を作成した．この API をプログラムに挿入することで任意の部分における各ノードの電力プロファイルを取得することができる．また，ネットワーク上でリアルタイムに消費電力をモニタリングする GUI を提供している．

4.2 実行時間プロファイルツール Tlog

アプリケーションの実行プロファイルを取得するためにイベントの実行時間を取得するライブラリである tlog を使用した．tlog とは *time log* の略称であり，イベントごとの log をと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（たとえば MPI_Send など）に挟みこむことで，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を対応づけることで領域の PDP・EDP を取得することができる．

4.3 DVFS コントロールランタイムライブラリ

DVS をプログラム内で制御するためにライブラリを実装した．周波数と電圧の組合せである *gear* を指定することで動作周波数の変更を行うことができる．

5. 評　　価

5.1 クラスタ環境

提案手法の有効性を検証するために 2 つの power-scalabe クラスタ（Crusoe と Turion を使用）において評価を行った．現在，DVFS を実装しているプロセッサは Pentium-M，ARM，Intel の XScale など様々なものがある．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．Turion は，Pentium-M と異なり，dual-issue の浮動小数点演算や SSE2，SSE3，さらに 64 bit 命令をサポートしており科学技術計算のアプリケーションを実行するのに適している．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．Turion クラスタは 8 ノード，Crusoe クラスタは 4 ノードで，それぞれネットワークは Gigabit Ethernet，Fast Ethernet である．**表 2** に Turion の**表 3** に Crusoe の周波数と電圧の組合せを示す☆．

提案するアルゴリズムを使用するには，DVS のオーバヘッドを定める必要がある．予備評価として Turion で周波数変更に必要なオーバヘッド（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．様々な周波数変更の組合せを行った結果，平均して 30 μ 秒の時間を必要とした．動作周波数を上げるときと下げるときでは，実際にはオーバヘッドが異なる可能性があるため，本論文ではオーバヘッドを 50 μ 秒とし，時間的余裕を持たせた．また，Crusoe においても，同じ値をオーバヘッドとして使用した．

☆ Turion の一部の周波数と電圧の組合せは公式にアナウンスされていないため推測で値を決めた．

表 1 クラスタの仕様
Table 1 The specification of clusters.

システム名	Turion クラスタ	Crusoe クラスタ
CPU	Turion MT-32	TM-5800
Clock	1.80 GHz	933 MHz
Cache L1/L2	64 KB/1 MB	64 KB/512 KB
Memory	1 GB(DDR)	256 MB(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	25 W	7.5 W

表 2 Turion の周波数と電圧
Table 2 Votage and frequency of Turion.

gear	周波数	電圧	TDP
1	800 MHz	0.90 V	9 W
2	1,000 MHz	1.00 V	–
3	1,200 MHz	1.05 V	–
4	1,400 MHz	1.10 V	–
5	1,600 MHz	1.15 V	–
6	1,800 MHz	1.20 V	25 W

表 3 Crusoe の周波数と電圧
Table 3 Voltage and frequency of Crusoe.

gear	周波数	電圧	FSB	TDP
1	300 MHz	0.90 V	100 MHz	–
2	533 MHz	1.10 V	133 MHz	–
3	667 MHz	1.20 V	133 MHZ	–
4	800 MHz	1.25 V	133 MHz	–
5	933 MHz	1.35 V	133 MHz	7.5 W

5.2 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，MG 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，各クラスタの特性を知るために各ベンチマークにおける *gear* ごとの性能と PDP を計測した．すべての評価は複数回実行し，実行性能が最大であったときのデータを最適化に使用する．**図 4** に Turion クラスタの**図 5** に Crusoe クラスタの標準の周波数を基準とした性能と PDP，EDP 消費の特性を示す．図 5 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分かった．

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．

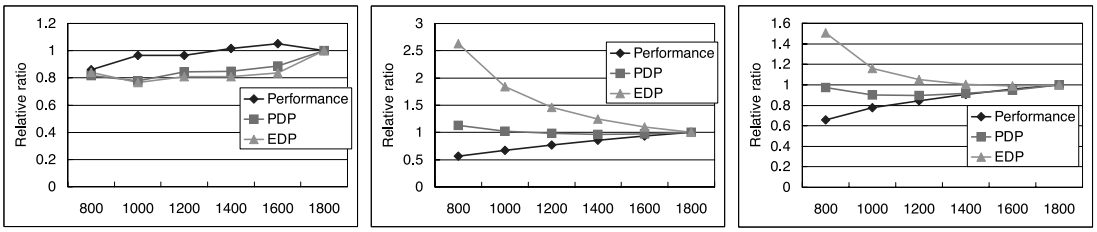


図 4 Turion クラスタの各 *gear* における性能・電力特性
Fig. 4 Characteristics of the Turion cluster on each gear with NPB.

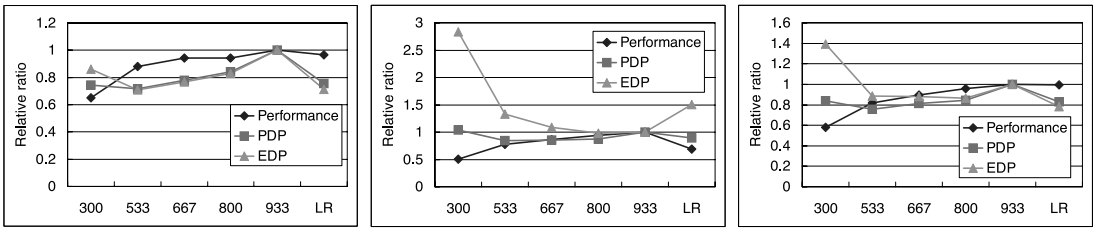


図 5 Crusoe クラスタの各 *gear* ごとの性能・電力特性
Fig. 5 Characteristics of the Crusoe cluster on each gear with NPB.

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．EDP に関して，IS では，周波数を下げることで EDP を減らすことができるが，他のベンチマークでは低い周波数では，急激に増加してしまう．

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分かる．最も低い周波数である 300 MHz 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 133 MHz から 100 MHz へと低下していることが原因である．また，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[6]．EDP に関して，IS では Turion クラスタと同様に削減が可能であるが，他のベンチマークでは，低い周波数を使用すると急激に増加する．

5.3 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 tlog のライブラリを付加し，領域とした．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．全体全通信はすべ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．

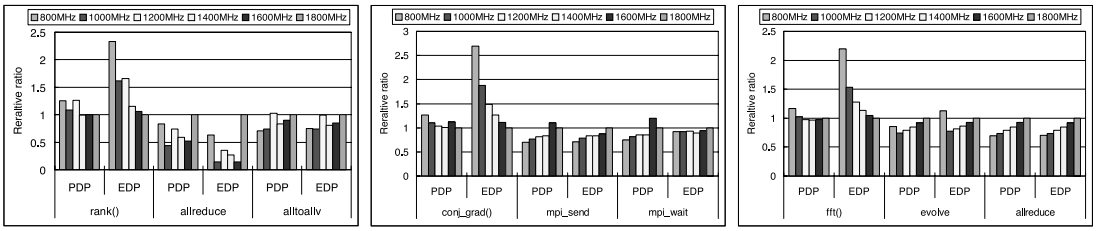
通信以外の戦略としては，メモリアクセスや計算量が多い領域に着目することである．オフチップのアクセスが必要な場合，待ち時間が生じるため低い周波数での動作によって性能低下を最小限に抑え，PDP の削減が期待できる．実際にオフチップアクセスを考慮して DVS を最適化することで PDP の削減ができることが報告されている[12]．しかし，オフチップアクセスするかどうかを判断するのには，詳細なプログラムの解析が必要であるため，本論文で関数レベルで計算領域を定めることで低い周波数で PDP の削減ができるかどうかを検討した．

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．IS と FT は全体全通信を含んでおり，長時間を通信に費やす可能性がある．このときに動作周波数を下げることで性能低下をわずかに抑え PDP を抑えることが期待できる．CG や MG にはアイドル状態となる MPI_Wait があり，このときにも PDP の削減が期待できる．

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．

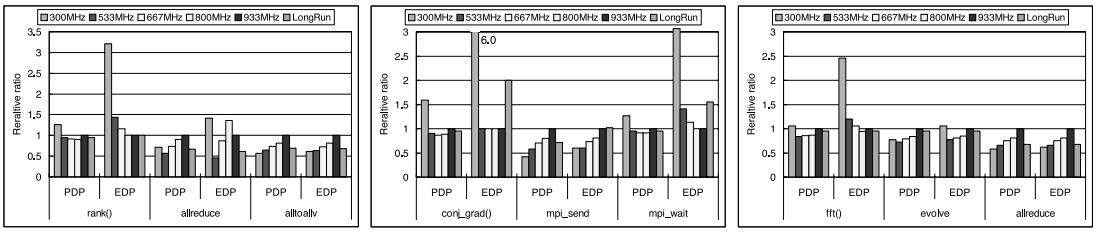
表 4 各ベンチマークの領域
Table 4 Regions of the benchmark.

prog	領域 1	領域 2	領域 3	領域 4	領域 5
IS	rank()	MPI_Alltoall	MPI_Alltoallv	MPI_Allreduce	
FT	fft()	evolve()	MPI_Alltoall	checksum()	
CG	conj_grad()	MPI_Send	MPI_Wait	MPI_Irecv	
MG	resid()	mg3P()	MPI_Send	MPI_Wait	MPI_Irecv



IS CG FT

図 6 Turion クラスタの各 *gear* による領域ごとの PDP，EDP の比較
Fig. 6 Details of the PDP and EDP on each gear in the Turion cluster with NPB.



IS CG FT

図 7 Crusoe クラスタの各 *gear* による領域ごとの PDP，EDP の比較
Fig. 7 Details of the PDP and EDP on each gear in the Crusoe cluster with NPB.

Turion 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，EDP 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 のいくつかの計算領域で PDP の削減が可能．
- 計算の領域では低い周波数で EDP が増加する．

Crusoe クラスタ：

図 7 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，EDP 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 はおおむね PDP，EDP の削減ができている．
- 計算領域では，300 MHz 使用時には EDP が急激に上昇．

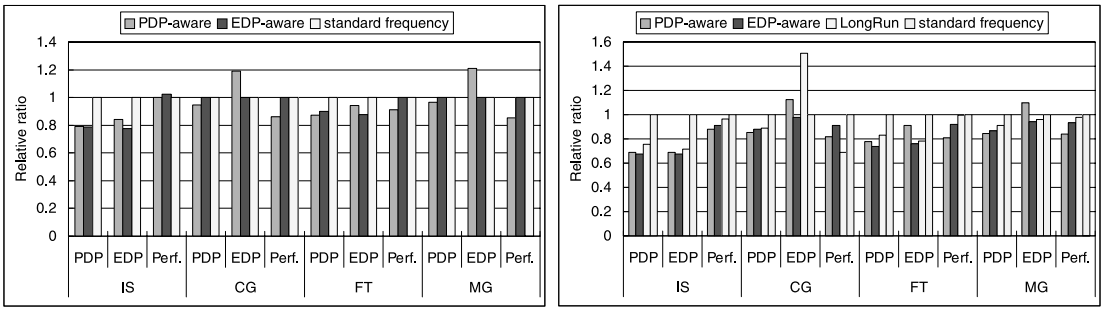
各アプリケーションにおける，これら 3 領域は実行時間の 9 割以上を占めており，各領域において周波数選択によって PDP や EDP の大幅な削減の可能性があることが分かったことから，今回の領域の選択は十分であったといえる．

5.4 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 つの指標で最適化を行った．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方が良いという結果になった．

図 8 (左) に Turion クラスタにおける標準の周波数である 1,800 MHz 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．すべ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



Turion クラスタ　　Crusoe クラスタ

図 8 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）
Fig. 8 PDP, EDP and the performance of our method normalized with a standard frequency on Relative ratio of PDP and EDP.

表 5 Turion クラスタのアルゴリズムが決定した周波数 [MHz]
Table 5 Selected Frequency of Truion in each benchmark.

		rank()	allreduce	alltoallv
IS	PDP	1800	1000	800
	EDP	1800	1000	1000
		fft()	evolve()	MPI_Allreduce
FT	PDP	1400	1400	1000
	EDP	1800	1800	1000
		conj_grad()	MPI_Send	MPI_Wait
CG	PDP	1600	1600	1400
	EDP	1800	1800	1800
		resid()	mg3P()	MPI_Send
MG	PDP	1400	1400	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]
Table 6 Selected Frequency of Crusoe in each benchmark.

		rank()	allreduce	alltoallv
IS	PDP	933	533	300
	EDP	933	533	300
		fft()	evolve()	MPI_Allreduce
FT	PDP	533	800	533
	EDP	800	533	300
		conj_grad()	MPI_Send	MPI_Wait
CG	PDP	533	533	800
	EDP	800	800	800
		resid()	mg3P()	MPI_Send
MG	PDP	667	533	533
	EDP	800	800	800

ている．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．また，周波数を固定した場合と比較しても約 10%の EDP の削減を実現している．このことから，領域ごとに最適な周波数を選択する効果が大きいことが分かる．

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．Turion クラスタと同様に 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 8 (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．これにはいくつかの理由が考えられる．まず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．また，システム全体におけるプロセッサの消費電力の占める割合が Crusoe クラスタ（90%）の場合 Turion クラスタ（60%）よりも大きく，DVFS によって消費電力を削減することの効果が Turion クラスタよりも大きくなる．結果として，IS では PDP を 30%以上，他のベンチマークでも約 20%程度の削減を実現した．また，周波数を固定した場合と比較しても約 12%の EDP の削減を実現している．このことから，領域ごとに最適な周波数を選択する効果が大きいことが分かる．一方で，PDP 最適化は性能低下を引き起こしている．CG において LongRun を使用することは大きな性能低下を引き起こす．この現象は我々の過去の研究においても生じており，細粒度通信と細かい計算を繰り返すことで，通信時に LongRun が動作周波数を下げた場合，その後の計算が短時間で終了してしまうため，LongRun は周波数を上げる必要がないと判断し，低い動作周波数のままアプリケーションを実行してしまうためである[6)]

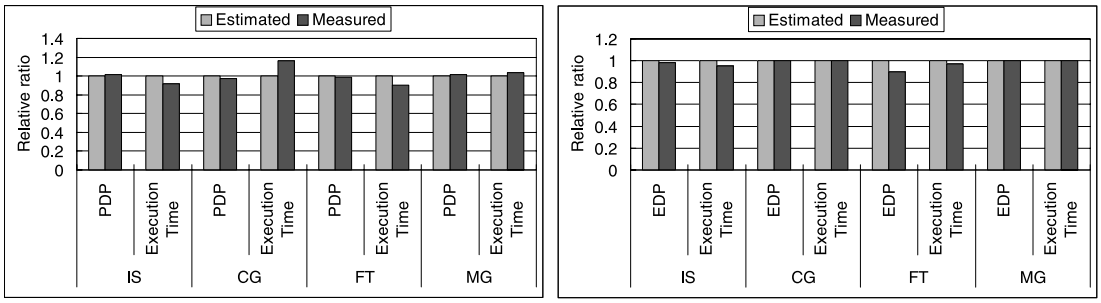


図 9 Turion クラスタにおける最適化の予測と実測の比較
Fig. 9 The difference between the estimated and measured result on the Turion cluster.

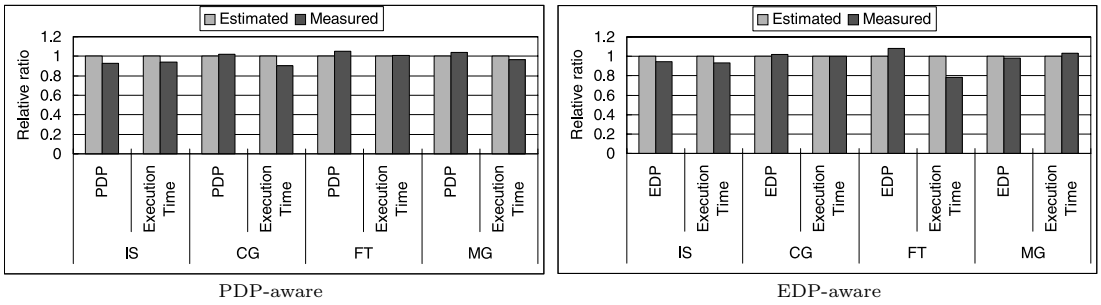


図 10 Crusoe クラスタにおける最適化の予測と実測の比較
Fig. 10 The differnece between the estimated and measured result on the Crusoe cluster.

5.5 アルゴリズムの予測精度

アルゴリズムを適用することで PDP，EDP 両方の削減を実現することができた．ここでは，我々のアルゴリズムの予測精度を評価する．**図 9** に Turion クラスタの**図 10** に Crusoe クラスタにおけるアルゴリズムが予想した周波数適用時の結果を実際に適用した場合を 1 とした場合との比較を示す（左が PDP 最適化，右が EDP 最適化）．Turion クラスタでは，おおむね，アルゴリズムが予想した結果と同じになり，本アルゴリズムの有効性が確認できる．PDP 最適化において CG の性能が予測と異なっている．これは DVS のオーバヘッドが我々の予測よりも大きかったのが原因ではないかと考えられる．EDP 最適化においてはほぼ予測と実測が一致している（CG と MG は周波数変更をしないため同じ）．

Crusoe クラスタにおいては，おおむね実行時間，PDP，EDP とも正確に予測することができている．EDP 最適化の場合，FT の実行時間が予測と大きく異なっている．しかし，実行時間は異なっているが，EDP はそれほど相違が見られない．

6. 考　　察

6.1 電力性能最適化について

本研究では，2 つの低消費電力クラスタを用いて電力性能を最適化するために，プロファイル情報を用いた動作周波数スケジューリングアルゴリズムの提案をし，評価を行った．

すべての周波数において予備実行を行い，それを基に最適な周波数を予測し，実行することで最大で 30%以上の PDP の削減を実現した．また，PDP だけでなく性能に比重を置いた EDP においても評価を行い，性能低下をわずかにおさえ，約 20%程度の EDP を削減し，電力性能を向上させることができた．

DVS のオーバヘッドのコストは無視することができない．そのため，我々は評価に先立ち DVFS のオーバヘッドを測定した．その結果，オーバヘッドは稀に 300 μ 秒要することもあったが，平均して 30 μ 程度となった．この値から 50 μ をオーバヘッドの値として用いてアルゴリズムを適用することでより正確な予測を行うことができ，実行時間，PDP，EDP の予測もおおむね正確にできた．

2 つのクラスタで評価を行うことで，クラスタの持つ消費電力の特性によって本手法の効果の現れ方が違

うことが分かった．Turion は TDP が 24 W であり，電圧の振れ幅も他のプロセッサと比べて小さい（1.2–0.9 V）ハイエンドなプロセッサ，たとえば Opteron のようなプロセッサを使用したクラスタにおいても本手法は有効であると考えている．Opteron の TDP は 83 W であり，動作周波数を変更する機構である PowerNow!を実装している．このようなプロセッサを搭載したクラスタにおいて，DVF を使用することで消費電力の削減を行うことは非常に効果が高いと考えられる[2]．

6.2 ネットワークの消費電力

近年，ネットワークインタフェース，とりわけネットワークスイッチの消費電力が計算機システムにおいて非常に重要となっている．Gigabit Ethernet や専用ネットワークを用いるような場合，スイッチの中にプロセッサが備わっており，消費電力が非常に大きくなっている．本論文では，ネットワークスイッチの消費電力は評価に含めていない．今回の評価に用いたスイッチは 24 ポートのスイッチであり，8 ノードしかないクラスタの消費電力の評価に加えることは適当でない．ネットワークスイッチに関して検討を行ったところ，1 ポートあたり 1 W 程度のスイッチを用いることで全体の消費電力への影響を少なくすることができる．一方で近年，ネットワークスイッチも消費電力と性能のトレードオフの関係になっており，今後スイッチを含めた消費電力の関係の評価が必要である．

6.3 アルゴリズムの自動化

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が，電力性能を大幅に向上させることができた．今回は，計算部分に関しては，細かく領域を分割しなかったが，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可能性がある．Hsu ら[12] 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により，自動的に領域を分割することでさらなる効果が期待できる．

6.4 通信ライブラリレベルでの領域設定についての考察

今回は，アプリケーションのソースコードにおいて，MPI 関数と主要な関数を領域として設定し，本手法の適用，評価を行った．MPI 関数を領域に設定した理由は，計算の領域に比べて通信処理についてはその振舞い大きく違いがあると考えられるからである．実際，MPI_Recv や MPI_Wait などにおいては，通信待ちが生じる．その場合，周波数を低減することにより，大幅な電力削減が見込まれる．しかしながら，MPI の実装によっては MPI_Recv において他のデータの送信が行われる可能性もあるし，受信側で周波数を下げたときには，プロトコル処理が遅くなる可能性もある．また，本実験においては collective 通信の MPI 関数を 1 つの領域としているが，その中では様々な送信と受信が混在している．

本手法は，基本的にはユーザのソースコードレベルでのプロファイルを前提としており，上に述べたような通信の詳細については考慮していない．本論文の手法は任意の領域に分割した場合に，粗の分割での最適周波数の組合せを求めるもので，前説の評価結果から，おおむね MPI レベルの通信を領域とすることで，最適化の効果が得られることが分かった．MPI ライブラリあるいはカーネルの通信部分のソース中にプロファイルコードを設定し，それを領域と設定することで，さらなる最適化の可能性はあるが，逆に周波数の切替えオーバヘッドが大きくなり，周波数が変更されず，全体として最適化がされないことも考えられる．

6.5 大規模システムへの適用

本論文では，小規模のクラスタシステムにおいて評価を行った．より大規模なシステムにおいて電力性能を最適にするために本手法を用いることができないか検討する必要がある．近年，HPC のシステムは数千以上のプロセッサを搭載するなど巨大化している．そのような環境では，本研究のように各ノードの消費電力を測定することによる最適化が困難である．そこで，小規模なクラスタでの結果から大規模なシステムの消費電力特性を予測することが求められる．

7. 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 による電力性能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，様々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．また，評価のために電力測定環境である PowerWatch を開発した．

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 30%以上の EDP の削減をすることができることが分かった．

今回の評価には科学技術計算の並列アプリケーションベンチマークを使用したが，他の分野のアプリケーションへの適用による評価を行いたい．たとえば現在最も消費電力の問題に悩まされているサーバシステムやデータセンタなどが対象とするアプリケーションにおいて適用し，PDP の消費を削減することによって管理コストや故障に対するコストなどの削減をすることが期待できる．

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) Adiga, N., Almasi, G., Almasi, G., Aridor, Y., Barik, R., Beece, D., Bellofatto, R., Bhanot, G., Bickford, R., Blumrich, M., Bright, A., et al.: An Overview of the BlueGene/L Supercomputer, *Supercomputing Conference 05* (*SC'05*) (Nov. 2005).
2) AMD Corp.: AMD Athlon^{TM} 64 Processor Power and Thermal Data Sheet (2005).
3) Brooks, D., Bose, P., Schuster, S., Jacobson, H., Kudva, P. and Cook, P.: Power-Aware Microarchitecture: Design and Modeling Challenges for Next-Generation Microprocessors, *IEEE Micro*, Vol.20, pp.26–44 (2000).
4) Transmeta Corp.: Longrun Thermal Management (2001). htto://www.transmeta.com/
5) Ge, R., Feng, X. and Cameron, K.W.: Performance-constrained Distributed DVS Scheduling for Scientific Applications on Power-Aware Clusters, *Supercomputing Conference 05* (Nov. 2005).
6) Nakashima, H., Nakamura, H., Sato, M., Boku, T., Matsuoka, S., Takahashi, D. and Hotta, Y.: Megaproto: 1 TFlops/10 kw rack is feasible even with only commodity technology, *Supercomputing Conference 05* (Nov. 2005).
7) Hsu, C.H. and Feng, W.C.: A Power-Aware Run-Time System for High-Performance Computing, *Supercomputing Conference 05* (Nov. 2005).
8) Kappiah, N., Freeh, V.W. and Lowenthal, D.K.: Just in Time Dynamic Voltage Scaling: Exploiting Inter-Node Slack to Save Energy in MPI Programs, *Supercomputing Conference 05* (Nov. 2005).
9) Warren, M., Weigle, E. and Feng, W.: High-Density Computing: A 240-processor Beowulf in One Cubic Meter, *Supercomputing Conference 02* (Nov. 2002).
10) Freeh, V., Pan, F., Kappiah, N. and Lowenthal, D.K.: Using Multiple Energy Gears in MPI Programs on a Power-Scalable Cluster, *Symposium on Principles and Practice of Parallel Programming* (PPoPP'05) (July 2005).
11) Freeh, V., Pan, F., Kappiah, N., Lowenthal, D.K. and Springer, R.: Exploring the Energy-Time Tradeoff in MPI Programs on a Power-Scalable Cluster, *16th International Parallel and Distributed Processing Symposium* (*IPDPS'05*) (Apr. 2005).
12) Hsu, C.H. and Kremer, U.: The Design, Implementation and Evaluation of a Compiler Algorithm for CPU Energy Reduction, *ACM SIGPLAN Conference on Programming Languages, Design and Implementation* (*PLDI'03*) (June 2003).

(平成 18 年 1 月 27 日受付)
(平成 18 年 5 月 22 日採録)

堀田 義彦（学生会員）

昭和 54 年生まれ．平成 15 年筑波大学第三学群情報学類卒業．現在，同大学大学院システム情報工学研究科在学中．クラスタシステム等の低消費電力化に関する研究に従事．

佐藤 三久（正会員）

昭和 34 年生．昭和 57 年東京大学理学部情報科学科卒業．昭和 61 年同大学大学院理学系研究科博士課程中退．同年新技術事業団後藤磁束量子情報プロジェクトに参加．平成 3 年通産省電子技術総合研究所入所．平成 8 年新情報処理開発機構並列分散システムパフォーマンス研究室室長．平成 13 年より，筑波大学システム情報工学研究科教授．同大学計算科学研究センター勤務．理学博士．並列処理アーキテクチャ，言語およびコンパイラ，計算機性能評価技術，グリッドコンピューティング等の研究に従事．IEEE，日本応用数理学会各会員．

木村　英明（学生会員）

昭和 58 年生．平成 16 年小山工業高等専門学校電子制御工学科卒業．平成 18 年筑波大学第三学群情報学類卒業．現在，同大学大学院システム情報工学研究科在学中．低消費電力クラスタシステムに関する研究に従事．

松岡　　聡（正会員）

1986 年東京大学理学部情報科学科卒業，1989 年同大学大学院博士課程から，学情報科学科助手に採用，同大学情報工学専攻講師を経て，1996 年に東京工業大学情報理工学研究科数理・計算科学専攻助教授．2001 年 4 月に東京工業大学学術国際情報センター教授，2002 年より国立情報学研究所の客員教授を併任．博士（理学）（東京大学）．高性能システム，並列処理，グリッド計算，クラスタ計算機，高性能・並列オブジェクト指向言語処理系，等の研究に従事．わが国のナショナルグリッドプロジェクトである NAREGI プロジェクトのサブリーダであり，また 2006 年時点でわが国最高性能のスーパコンピュータ TSUBAME を構築．1996 年度情報処理学会論文賞，1999 年情報処理学会坂井記念賞 2006 年学術振興会賞（JSPS Award）等を受賞．国際学会 ISOTAS'96，ECOOP'97，ISCOPE'99，ACM OOPSLA'02，IEEE CCGrid'03，HPC Asia 05，Grid2006 等のプログラム（副）委員長，IEEE/ACM Supercomputing'04–Network Track Chair，Reflection'01，IEEE CCGrid'06 大会委員長．また，グリッド国際標準化団体の Global Grid Forum の Steering Group 委員等を務める．

朴　　泰祐（正会員）

昭和 36 年生．昭和 59 年慶應義塾大学工学部電気工学科卒業．平成 2 年同大学大学院理工学研究科電気工学専攻後期博士課程修了．工学博士．昭和 63 年慶應義塾大学理工学部物理学科助手．平成 4 年筑波大学電子・情報工学系講師，平成 7 年同助教授，平成 16 年同大学大学院システム情報工学系助教授，平成 17 年同教授，現在に至る．超並列計算機アーキテクチャ，ハイパフォーマンスコンピューティング，クラスタコンピューティング，グリッドに関する研究に従事．平成 14 年度および平成 15 年度情報処理学会論文賞受賞．日本応用数理学会，IEEECS 各会員．

高橋　大介（正会員）

昭和 45 年生．平成 3 年呉工業高等専門学校電気工学科卒業．平成 5 年豊橋技術科学大学工学部情報工学課程卒業．平成 7 年同大学大学院工学研究科情報工学専攻修士課程修了．平成 9 年東京大学大学院理学系研究科情報科学専攻博士課程中退．同年同大学大型計算機センター助手．平成 11 年同大学情報基盤センター助手．平成 12 年埼玉大学大学院理工学研究科助手．平成 13 年筑波大学電子・情報工学系講師．平成 16 年筑波大学大学院システム情報工学研究科講師．博士（理学）．並列数値計算アルゴリズムに関する研究に従事．平成 10 年度情報処理学会山下記念研究賞，平成 10 年度，平成 15 年度情報処理学会論文賞各受賞．日本応用数理学会，ACM，IEEE，SIAM 各会員．