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

2008－ARC－177 (16)
2008－HPC－114 (16)
2008／3／5

自動プログラム領域分割を用いた実行時DVFS制御

木村英明† 佐藤三久†
今田貴之† 堀田義彦†

近年，PC クラスタにおいて DVFS を適切に制御することにより省電力化を実現する手法が数多く提案されている．プログラムを複数領域に分割しプロファイルを取得することで最適実行系列を決定する静的 DVFS 制御ではその煩雑さが問題であった．また，プログラム実行時に割り込みによって各種カウンタを取得し周波数を決定するランタイム DVFS 制御ではプログラムの特性が一定でない場合に適切な電力削減を実現できなかった．本稿では，プログラムを複数領域に分割しプログラム実行時に動作周波数を決定するコード埋め込み型動的 DVFS 制御を提案する．特徴として，同じ特性を持つ区間に分けることによりプログラムの情報を反映することができる．動作周波数の決定には β アダプテーションを用いた．評価の結果，コードを埋め込むことにより β アダプテーションで指定する性能低下条件を満足する適切な実行系列を決定できることが分かった．しかしながら，CPU 以外の消費電力が大きく，残念ながら PC クラスタ全体電力の削減までには到らなかった．

Dynamic DVFS control with defining program region using low-impact instrumentation

Hideaki Kimura,† Mitsuhisa Sato,† Takayuki Imada†
and Yoshihiko Hotta †

Recently, several energy reduction techniques using DVFS have been presented for PC clusters. Static DVFS control needs to divide program and to get profile. It is troublesome for users. Run-time DVFS control acquires program counter information by periodic interrupt to decide frequency. Run-time DVFS control can't reduce power consumption in certain applications. In this paper, we propose a dynamic DVFS control with defining program regions. We can use information on the program by defining program regions with same characteristics. Our technique decides the frequency dynamically while the program is executed. We use the β-adaptation to make a decision of the frequency.

We have designed and implemented our DVFS technique and evaluated it. The result shows that we have improved the β-adaptation by instrumented codes. Our dynamic DVFS control can select lower voltage and frequency than existing run-time DVFS control while meeting a given deadline. However, we lost the much power consumption of PC cluster because the execution time increased.

1. 序　　論

近年，PC クラスタを構成する各ノードの消費電力が問題となっている．特にプロセッサの消費電力増加により，信頼性の低下や実装密度の低下，電気代の増加など様々な問題が発生している[1]．このような問題を解決する方法として，動作周波数と動作電圧を動的に制御する DVFS（Dynanmic Voltage and Frequency Scaling）機構を適切に利用する方法が考えられる．DVFS 機構を適切に制御することで，高い性能と省電力を両立することができる．現在，PC クラスタ等の並列システムにおいて DVFS 機構を適切に制御することで省電力化を実現する手法が数多く提案されている[2)3)4)]．

PC クラスタ等の高性能計算向け並列システムにおいて省電力化を実現する方法として，プロファイルを用いた静的電力削減手法が盛んに提案されている．これは，プログラム中にコードをインスツルメントすることでプログラムを複数領域に分割し，プロセッサが動作可能な様々な周波数でプログラムを実行することでプロファイルを取得，このプロファイルを静的に解析しプログラム全体の最適実行系列を決定する，というものである．PC クラスタ等で頻繁に扱う大規模科学技術アプリケーションの多くは，計算と通信といった特徴的な動作を繰り返すことが多く，次に実行する処理を予測しやすい．このため，プロファイルを用いた静的電力削減手法が効果的に動作する．我々も，動

† 筑波大学大学院 システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba

作周波数切り替えオーバヘッドを考慮した静的電力削減手法を提案している[5]．プロファイル取得をともなう静的電力削減手法では，最適実行系列を決定できる反面，様々な動作周波数によるプロファイル取得を必要とし，多くの手間を要する．

ランタイムに動作周波数を決定するといった方法も考えられる．Linuxでは過去一定時間のプロセッサ状態を監視し次の動作周波数を決定するCPUfreq[6]が標準搭載されている．プログラム実行時に動作周波数を決定するため，プロファイル取得する必要はない．さらに，プログラムを複数領域に分割するといった作業も必要とせず，非常に簡単に利用することができる．しかしながら，通信と計算を短い時間間隔で繰り返すようなアプリケーションでは性能低下の一因となることが知られている．大規模科学技術アプリケーションでは，ソースコードの情報を活用することで効果的な電力削減を期待できるが，ランタイム電力削減手法では利用しない．

本稿では，プログラム中にコードを埋め込むことでプログラムを複数領域に分割し，プログラム実行時に動作周波数を決定する“コード埋め込み型動的DVFS制御”を提案する．領域区分のためのコードをインスツルメントし，それぞれの領域の動作周波数を実行時に決定する．インスツルメントすることでソースコード上の特性を反映できるという利点がある．インスツルメントの方法としては，我々がこれまでに提案した“プログラムの実行に与える影響の少ないインスツルメント手法[7]”を利用する．これは，ソースコードの情報と予備実行によって取得する実行情報からコード挿入箇所を決定する手法である．この手法を用いることで，挿入したコードが頻発することによるプログラムの特性変化を防ぎつつ，各領域の特性が一定となるインスツルメントを実現できる．このインスツルメントによってプログラムを複数領域に分割し，プログラム実行中にプロセッサのカウンタ情報から動作周波数を決定，変更する．動作周波数の決定にはChungら[3]のβアダプテーションを利用する．これは，周波数と性能の相関から動作周波数を決定する方法である．

本稿の構成は以下のようになっている．第2章では，関連研究について述べる．第3章では，提案する動的電力削減手法の概要について述べる．第4章で評価結果を示し，第5章で考察を行う．最後にまとめと今後の課題について述べる．

2. 関連研究

近年，PCクラスタにおいて省電力化を実現する方式が数多く提案されている．プロファイルを取得し最適化を行う手法として，Geら[2]の方法やFreehら[8]の方法がある．Geらは並列計算プログラムの通信に着目し，通信部において低い周波数を選択することで省電力化を実現している．プログラムを通信部と計算部に分け，プロセッサが動作可能な様々な周波数でプログラムを実行しプロファイルを取得する．このプロファイル情報から各領域の最適動作周波数を決定している．FreehらはOPM ($OperationsPerMiss$)の変動からプログラムを複数領域に分割し，プロファイルを取得するとともに動作周波数を決定する手法を提案している．

Chungら[3]はランタイムにプログラムの動作周波数を決定する方法を提案している．プログラム実行中にMIPS値の変動を観測し，許容する性能低下の範囲内に処理を終了するよう動作周波数を調整する．この手法では，プログラムの分割やプロファイル等を取得する必要が無い反面，ソースコード情報を活用できない．

佐々木ら[9]は，プロセッサが持つカウンタ値と性能寄与率から許容する性能低下の範囲内で電力を削減する手法を提案している．学習用プログラムを実行するとともにプロセッサが持つ様々なカウンタ値を取得し，統計的な学習により性能寄与率の高いカウンタを決定する．その後，評価プログラム実行時にカウンタ値と性能寄与率の関係を用い，許容する性能低下の範囲内で電力を削減するよう動作周波数を変更する．学習用プログラムを異なる周波数で実行し，取得可能なすべてのカウンタ値を取得する必要がある．しかしながら，一度学習を行えば他のプログラムに対しても適用可能である．本手法は，プログラム実行前に学習を行わず，プログラム実行時に指定した各領域の動作周波数を適応させていく．

3. 実行時DVFS制御

本章では，プログラム本実行時に動作周波数を決定する“実行時DVFS制御”について述べる．実行時DVFS制御として，“コード埋め込み型動的DVFS制御”と“ランタイム電力削減手法”が考えられる．これらの違いはコード埋め込みの有無であり，実行時動作周波数決定手法については同一のものを利用することができる．

3.1 コード埋め込み型動的DVFS制御

プログラム中にコードを埋め込むことでプログラムを複数領域に分割し，領域ごとに異なるデータを用いて実行時に動作周波数を決定することを“コード埋め込み型動的DVFS制御”と呼ぶ．コードの埋め込みには，影響の少ないインスツルメント手法[7]を利用することができる．これは，以下の条件を同時に満たすインスツルメント手法である．

- ソースコード上にインスツルメントすることで，プログラムを複数領域に分割する．
- 適切な最適化を実現するために，領域内の特性が同一になるようインスツルメントを行う．
- インスツルメントによってプログラムの挙動や特性が大幅に変化しない．

これにより，プログラム本来の実行に影響を与えることなく適切なインスツルメントを実現できる．

表 1　電力削減手法の比較

	ランタイム DVFS 制御	コード埋め込み型動的 DVFS 制御	静的 DVFS 制御
コード埋め込みの有無	割り込み	コード埋め込み	コード埋め込み
動作周波数	本実行中に決定	本実行中に決定	本実行前に決定
電力削減	△ 通信と計算を繰り返す場合に不利	○ 最適ではないが，ランタイムの問題を解決	◎ 最適実行系列を決定することが可能
ユーザへの負担	◎ ユーザが意識せず利用可能	○ 領域の指定が必要	△ 領域の指定，プロファイル取得が必要

動作周波数の決定には既存の実行時動作周波数決定手法を用いる．多くの実行時動作周波数決定手法は，プログラム実行時に各種プロセッサカウンタや電力データを取得することで実現する．コード埋め込みによって分割した各領域は最初から決まった周波数でプログラムを実行せず，プログラム実行時に周波数を適応させていくことによって省電力化を実現する．この点がプロファイル取得をともなう静的 DVFS 制御と異なる．

プログラムを複数領域に分割しない場合，特性の異なる処理によって実行時 DVFS 制御が適切に動作しない可能性がある．例えば計算時に取得したデータを用いて通信部の動作周波数を決定する可能性がある．プログラムを複数領域に分割することでこのような問題を回避し，適切な実行時 DVFS 制御を実現できると考えられる．

3.2　ランタイム DVFS 制御

割り込みによってプログラムの実行情報を取得し，プログラム実行時に動作周波数を決定する方法を “ランタイム DVFS 制御” と呼ぶ．

ランタイム DVFS 制御はユーザ側にとって非常に扱いやすく，プログラムの特性を熟知している必要は無い．Linux で広く利用されている CPUfreq は，ランタイム手法の一種であるといえる．しかしながら，割り込みによる実装は細粒度通信やメモリアクセスと計算処理を短い時間間隔で繰り返し実行するようなアプリケーションにおいて効果を得られないことが知られている．大規模科学技術計算アプリケーションの多くは，ソースコードの情報によって次の処理を予測しやすいが，ランタイム実行ではこのソースコードの情報を使用しない．

3.3　DVFS 制御に関する比較

表 1 に各手法の比較を示す．ランタイム DVFS 制御は，割り込みによって実現する．ユーザへの負担は小さいが対象プログラムによっては適切に動作しない可能性がある．静的 DVFS 制御は，対象プログラムのプロファイルを取得することで実現する．このためユーザへの負担が非常に大きい．静的 DVFS 制御ではプログラム実行前に動作周波数を決定するが，．コード埋め込み型動的 DVFS 制御はこの中間に位置する．

コード埋め込み型動的 DVFS 制御は，プロファイルを取得する必要が無いため静的 DVFS 制御と比較してユーザに対する負担が小さい．プログラムを複数領域に分割する必要があるが，これは我々が提案した影響の少ないインスツルメント手法[7] を利用できる．ランタイム DVFS 制御では通信と計算を繰り返すアプリケーションで性能が低下するが，コード埋め込み型動的 DVFS 制御ではこれを回避できると考えられる．

3.4　β アダプテーション

本稿では，実行時動作周波数決定手法として Chung ら[3] が提案した β アダプテーションを用いた．これは，プログラム実行時に動作周波数と性能の相関を求め，指定した性能低下条件を満たす範囲でもっとも低い周波数を選択する手法である．周波数決定のために電力データを使用しておらず，“低い電圧・周波数を選択することでエネルギーを削減できる” ことが前提となっている．

周波数の決定には式 (1)，式 (2) を用いる．β は周波数と性能の相関，δ はユーザが指定する実行時間の低下率を意味する．

$$\beta = \frac{\sum_{i=1}^{n} \left(\frac{f_{max}}{f_i} - 1\right)\left(\frac{\mathrm{FLOPS}(f_{max})}{\mathrm{FLOPS}(f_i)} - 1\right)}{\sum_{i=1}^{n} \left(\frac{f_{max}}{f_i} - 1\right)^2} \quad (1)$$

$$f^* = \max\left(f_{min}, \frac{f_{max}}{1 + \delta/\beta}\right) \quad (2)$$

Algorithm 1 にアルゴリズムを示す．Chung らは，β アダプテーションをランタイムシステムとして実現した．本稿では，β アダプテーションの適用をランタイムシステムに限定せず，コード埋め込み型動的 DVFS 制御にも適用する．

Algorithm 1 β-adaptation

Initialize phase - executing program at f_i
repeat
1. Compute β
$$\beta = \frac{\sum_{i=1}^{n} \left(\frac{f_{max}}{f_i} - 1\right)\left(\frac{\mathrm{FLOPS}(f_{max})}{\mathrm{FLOPS}(f_i)} - 1\right)}{\sum_{i=1}^{n} \left(\frac{f_{max}}{f_i} - 1\right)^2}$$
2. Compute f^*
$$f^* = \max\left(f_{min}, \frac{f_{max}}{1+\delta/\beta}\right)$$
3. Execute the program at f^*
4. Update FLOPS(f^*)

until the program is finished

4. 実クラスタ環境における評価

4.1 評価環境

評価対象として，Opteron148 (2.2GHz, L2: 1MB) を搭載した 8 ノードクラスタを用いる．動作可能な周波数は 2200MHz, 2000MHz, 1800MHz, 1600MHz, 1400MHz, 1200MHz, 1000MHz の 7 種類である．メモリは DDR-SDRAM 1GB であり，カーネルは Linux kernel 2.6.19 with perfctr，コンパイラは gcc 4.1.1, MPI ライブラリとして LAM MPI 7.1.3 を用いた．FLOPS 値を取得するために perfctr パッチを当て，PAPI 3.2.1 ライブラリを用いた．

評価ベンチマークとして NPB (NAS Parallel Benchmarks) 3.2.1-MPI を用いた．カーネルベンチ 5 つの中から，主に浮動小数点演算を行う FT, MG, CG, EP について評価を行う．整数ソートを行う IS ベンチマークでは，FLOPS 値の変動が少なく β アダプテーションによる効果が得られないと考えられるため，今回の評価から除外した．各ベンチマークは CLASS=C で実行する．

電力評価環境として，ホール素子を用いた電力測定システム PowerWatch[7] を用いる．各ノードは電力測定装置を介してコンセントに接続している．したがって，評価に用いる電力値はプロセッサの電力のみならずメモリや HDD などのその他構成要素，電源における AC-DC 変換損失等を含む．

4.2 ランタイム DVFS 制御に関する評価

ベンチマークプログラムを実行すると同時に割り込みによって FLOPS 値の取得，動作周波数の決定を行う．割り込みの頻度は 1 秒とした．許容する性能低下率 δ の値は 0.0 (性能低下を許さない)，0.1, 0.2, 0.5 の 4 種類とした．

図 **1** に CG，図 **2** に FT を実行した時の消費電力遷移を示す．プログラム開始直後に異なる周波数で実行し，初期値を取得する．初期値取得後，CG では δ に応じ低い周波数を選択することで消費電力を削減している．しかしながら，FT では δ を変更しても実行時間，電力特性に変化が見られない．これは，δ に応じて適切に動作周波数を変更していないためである．

ランタイム DVFS 制御では，プログラムの特性を意識することなく動作周波数を決定する．したがって，通信と計算といった特性の異なるコードが混在していたとしても式 (1) を用いて同一の処理を行う．これにより，β アダプテーションが適切に動作しなくなる．

例えば，計算時に各周波数 f_i の性能 FLOPS(f_i) を取得し，その後通信など特性の異なるコードを実行時に極端に小さい FLOPS 値を取得したとする．このような状況で式 (1) から動作周波数と性能の相関 β を求めると，極端に大きな β を得る．β が 1.0 を大幅に超える値となる可能性もあり，式 (2) によって常に最高周波数を選択するようになる．以上の理由から，β アダプテーションが期待される動作をしていない．

表 **2** ランタイム DVFS 制御時の実行時間

benchmark	実行時間変化率 [%]		
	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.5$
CG	+4.89	+15.8	+33.0
FT	+0.48	+1.97	+2.09
MG	+3.50	+11.0	+42.1
EP	+6.82	+18.4	+50.5

表 **3** ランタイム DVFS 制御時の消費エネルギー

benchmark	消費エネルギー変化率 [%]		
	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.5$
CG	+1.04	+7.14	+14.3
FT	+0.39	+1.58	+1.77
MG	-0.99	+2.03	+16.0
EP	+3.13	+8.50	+26.5

CG においても通信と計算といった特性の異なる動作が繰り返し実行されるが，FT と比べて通信の粒度が細かいためこの問題を回避できた．今回の評価では，割り込み時間間隔を 1 秒としている．FT では MPI_Alltoall など比較的規模の大きな通信と割り込み間隔と合致し，β アダプテーションが適切に動作しなかった．割り込み時間間隔によっては CG など他のアプリケーションにおいても同様の問題が発生する可能性がある．

表 **2** に許容する実行時間増加値 δ を変化させた時の実行時間変化率，表 **3** にその時の消費エネルギーの変化率を示す．δ に応じて低い動作周波数を選択し，実行時間が増加している．また，実行時間が増加するにしたがって消費エネルギーも増加している．

4.3 コード埋め込み型動的 DVFS 制御に関する評価

本節では，影響の少ないインスツルメント手法によってプログラムを複数領域に分割し β アダプテーションを適用した結果を述べる．

分割した領域ごとに FLOPS 値を取得し，β アダプテーションを実行する．それぞれの領域について最初の数回は，初期化のために異なる動作周波数で実行する．それ以降は，式 (1)，式 (2) を用いて動作周波数を変更しながら処理を進める．

図 **3** に CG，図 **4** に FT を実行した時の消費電力遷移を示す．CG，FT ともに許容する実行時間増加率 δ に応じて低い動作周波数で処理を行っており，消費電力を削減している．CG を $\delta = 0.5$ として実行した場合，一部の領域ではランタイム DVFS 制御時より低い動作周波数で実行している．ランタイム DVFS 制御時と異なり，FT においても指定した実行時間の増加率 δ に応じて低い周波数を選択している．これらは，プログラムを複数領域に分割し領域別に FLOPS 値の蓄積・更新したためであると考えられる．

表 **4** に許容する実行時間増加値 δ を変化させた時の実行時間変化率，表 **5** にその時の消費エネルギー変化率を示す．

EP に対して影響の少ないインスツルメント手法を適用したところ，主要ループの外側にコードをインス

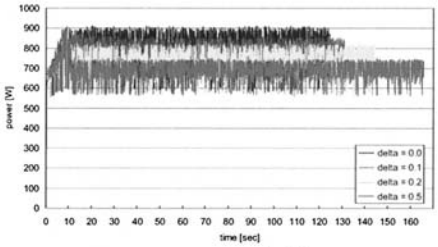

図 1　ランタイム DVFS 制御 CG

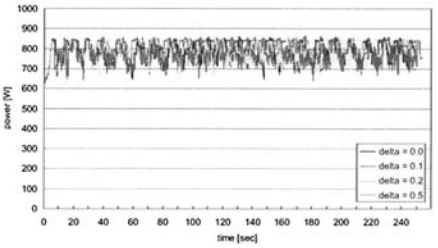

図 2　ランタイム DVFS 制御 FT

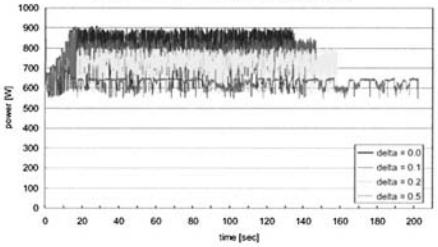

図 3　コード埋め込み型動的 DVFS 制御 CG

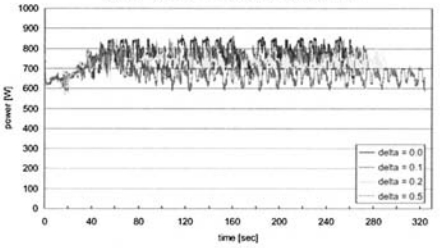

図 4　コード埋め込み型動的 DVFS 制御 FT

ツルメントした．今回の実装では，領域を実行する最初の数回で初期値を取得するなど領域内を複数実行することを前提としている．EP では領域実行回数が少なく，β アダプテーションによる周波数選択処理が行われる前にプログラムを終了する．一方，主要ループの内側にコードをインスツルメントした場合，このオーバヘッドによりプログラムの特性を変化させてしまう．そのため，評価対象から外している．

ランタイム DVFS 制御の結果と比較して，実行時間の増加率が大きくなっていることが分かる．これは，プログラムを複数領域に分割したことで領域別に適切な周波数を決定したためである．例えば，通信時など動作周波数と性能の相関 β の値が小さな場面でランタイム DVFS 制御時より低い周波数を選択している．この結果，許容性能低下率 δ を満たす範囲で実行時間が増加している．

ランタイム DVFS 制御の結果と比較して消費エネルギーが増加している．これは，ランタイム DVFS 制御と比較して実行時間増加率が大きくなっているためである．今回の評価環境ではプロセッサ以外の構成要素の消費電力が非常に大きく，DVFS 制御によるプロセッサ電力削減が実行時間増加によるエネルギー増加によって打ち消されてしまう．

5. 考　　察

5.1 実行時間に関する比較

実行時 DVFS 制御手法としてランタイム DVFS 制御手法とコード埋め込み型動的 DVFS 制御手法の実行時間について比較する．コード埋め込み型動的 DVFS 制御では，プログラムを複数領域に分割し領域ごとに相関データを取得する．このため，プログラムの特性変化による FLOPS 値の変動を回避できる．その結果，

表 4　コード埋め込み型動的 DVFS 制御時の実行時間

benchmark	実行時間変化率		
	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.5$
CG	+8.67	+17.4	+43.9
FT	+4.63	+10.7	+23.1
MG	+8.70	+16.9	+34.1

表 5　コード埋め込み型動的 DVFS 制御時の消費エネルギー

benchmark	消費エネルギー変化率 [%]		
	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.5$
CG	+3.95	+7.71	+22.0
FT	+2.16	+4.67	+8.78
MG	+1.12	+4.26	+12.6

ランタイム DVFS 制御手法と比較して低い動作周波数を選択し，実行時間の増加が許容性能低下率 δ に近くなっている．以上のことから，コード埋め込み型動的 DVFS 制御のようにコード中にインスツルメントすることで実行時 DVFS 制御を効果的に利用していると言える．

5.2 システム全体の消費エネルギー

消費エネルギーについて比較すると，実行時間増加率が大きくなるにしたがって消費エネルギーの増加を確認できる．これは，プログラムの実行時間が増加しプロセッサ以外の構成要素による定常的な消費エネルギーが増加するためである．本評価結果は，PC クラスタ全体の消費電力を考えた場合に低い動作周波数を選択することが必ずしも省電力化に繋がるとは限らないことを示唆している．プロセッサのみに着目すれば，低い動作周波数を選択することで省電力化を実現することができる．しかしながら PC クラスタの消費エネルギーについて考えると，β アダプテーションの前提となっている “低い周波数を選択することで省電力化を実現できる” が成り立つとは限らない．特にプロ

セッサ以外の構成要素による消費電力が大きい環境ではこの傾向が強い．

5.3 実行時周波数決定手法の検討

DVFS 制御によって省電力化を実現するために，β アダプテーションに変わる実行時動作周波数決定手法を検討しなければならない．β アダプテーションは，ユーザが与えた許容性能低下率を満たす範囲内で最低の動作周波数を選択することで省電力化を目指す手法である．動作周波数を決定するために電力データを用いておらず，性能のみを指標としている．性能のみを指標とする DVFS 制御手法は β アダプテーション以外にも数多く存在する．このような手法の多くはプロセッ以外の構成要素による消費エネルギーを考慮していない．DVFS 制御によるプロセッサのエネルギー削減効果と実行時間増加によるプロセッサ以外のエネルギー増加はトレードオフの関係にあり，PC クラスタの省電力化を実現するためには定常的なエネルギーを無視することはできない．

PC クラスタシステムの省電力化を実現するために“性能を評価するための情報”と“消費電力に関する情報”を与える必要があると考えられる．これまでに提案されてきた性能を指標とする DVFS 制御は，定常的な消費電力の大きい PC クラスタにおいて必ずしもエネルギーを削減できるとは限らない．各ノードの消費電力，あるいはプロセッサとその他構成要素による消費電力の比率などを与える必要がある．性能・消費電力情報を用いた実行時周波数決定手法とコード埋め込み型動的 DVFS 制御を組み合わせることにより，PC クラスタの省電力化を実現できると考えられる．

比較的小規模な PC クラスタであれば各ノードの電力測定結果をそのまま反映させることも可能であるが，超大規模な並列システムでは各ノードで消費電力を取得することはコストの問題から難しい．したがって，プログラム実行時に消費電力を予測するシステムについても検討する必要がある．

6. 結　　論

本稿では，影響の少ないインスツルメント手法を利用したコード埋め込み型動的 DVFS 制御を提案した．これはインスツルメントによってプログラムを複数領域に分割し，プログラム実行時に動作周波数を決定する手法である．これにより，静的 DVFS 制御で問題となったプロファイル等取得の煩雑さや，ランタイム DVFS 制御で問題となった特性の異なる処理の繰り返し時の性能低下を回避することができる．

プログラム実行時に動作周波数を決定する手法として Chung らの β アダプテーションを用い，PC クラスタによる評価を行った．その結果，コード埋め込み型動的 DVFS 制御はランタイム DVFS 制御手法と比較して優位な結果を得た．ランタイム DVFS 制御と実行時動作周波数決定手法を組み合わせて実行した場合，プログラムによっては適切に動作しないことがあった．しかしながら，コード埋め込み型 DVFS 制御ではこれを回避することができた．

低い動作周波数を選択することによって消費エネルギーが増加するという結果を得た．これは，プロセッサ以外の構成要素による消費エネルギーが増加したことが原因である．プロセッサの動作周波数を下げることによってプロセッサの消費エネルギーを削減することが可能であるが，実行時間の増加によってプロセッサ以外の構成要素による消費エネルギーが増加し効果を打ち消してしまう．DVFS 制御によって低い周波数を選択することが PC クラスタの消費エネルギーを削減するとは限らない．今後は，プロセッサ以外の構成要素の省電力化やシステム全体の消費エネルギーを考慮した DVFS 制御について検討する必要がある．

参 考 文 献

1) Chung-Hsing Hsu and Ulrich Kremer. The Design, Implementation, and Evaluation of a Compiler Algorithm for CPU Energy Reduction. In *PLDI*, pp. 38–48, 2003.
2) Rong Ge, Xizhou Feng, and Kirk W. Cameron. Performance-constrained Distributed DVS Scheduling for Scientific Applications on Power-aware Clusters. In *SC05*, 2005.
3) Chung-Hsing Hsu and Wu chun Feng. A Power-Aware Run-Time System for High-Performance Computing. In *Supercomputing-Conference*, 2005.
4) Nandini Kappiah, Vincent W. Freeh, and David K. Lowenthal. Just In Time Dynamic Voltage Scaling: Exploiting Inter-Node Slack to Save Energy in MPI Programs. In *SupercomputingConference*, 2005.
5) Y. Hotta, M. Sato, H. Kimura, S. Matsuoka, T. Boku, and D. Takahashi. Profile-based Optimization of Power-Performance by using Dynamic Voltage Scaling on a PC cluster. In *HP-PAC in IPDPS*, 2006.
6) Linux kernel CPUfreq subsystem. http://www.kernel.org/pub/linux/utils/kernel/cpufreq/cpufreq.html.
7) 木村英明, 佐藤三久, 堀田義彦, 今田貴之. 影響の少ないインスツルメント手法と電力最適化のためのプログラム領域分割. 情報処理学会論文誌 Vol.48 No.SIG13 (ACS19), pp. 247–259, 2007.
8) Vincent W. Freeh and David K. Lowenthal. Using multiple energy gears in MPI programs on a Power-scalable Cluster. In *PPoPP*, pp. 164–173, 2005.
9) 佐々木広, 浅井雅司, 池田佳路, 近藤正章, 中村宏. 統計情報に基づく動的電源電圧制御手法. 情報処理学会論文誌コンピューティングシステム (ACS16), pp. 80–91, 2006.