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

2007－ARC－172 (10)
2007－HPC－109 (10)
2007／3／1

電力的制約を考慮した低消費電力指向最適化コンパイラ

塚本智博[†] 片桐孝洋[†]
吉瀬謙二[††] 弓場敏嗣[†]

本研究では，プロセッサの省電力化を目的として，DVFS 制御を利用したプロファイル情報に基づく最適化コンパイラを開発した．プログラムの稼動条件による時間的制約と電力的制約のそれぞれに対応する最適化方として，(1) 実行時間の増加を許容範囲内に抑えて消費電力量を削減する，(2) 消費電力量を許容範囲内に抑えてできるだけ高い性能を達成する，という 2 つを設定した．また，x86 プロセッサが持つタイムスタンプカウンタをプロファイル情報として利用することで，オーバヘッドを低く抑えてプロファイルできることを示した．本最適化コンパイラにより，ユーザが指定する実行時間の許容範囲に対して，平均で 5%程度の誤差に抑えて，プロセッサの消費電力量を削減できることを確認した．また，消費電力量の許容範囲に対しては，平均で 5%程度の誤差に抑えられることを確認した．

An Optimization Compiler Considering Energy Limitation for CPU Energy Reduction

Tomohiro Tsukamoto,[†] Takahiro Katagiri,[†] Kenji Kise[††]
and Toshitsugu Yuba[†]

We developed an optimization compiler based on the profile information that uses DVFS control for power-saving of CPU. We set two optimization policies, to this optimization compiler :(1)optimization based on the threshold of execution time; (2)optimization based on the threshold of energy consumed. In addition, we implemented a profile option with a low overhead using the x86 processor's Time Stamp Counter into this optimization compiler. We conclude that our optimization compiler is acceptable, since it has only 5% error on average to a specified parameter threshold from users.

1. はじめに

プロセッサの高性能化に伴い，その消費電力は増加の傾向にある．このような状況のもとで，組込みシステムや基幹サーバの開発では，計算機資源の電力消費を抑えることが求められている．例えば，モバイル端末においては，電力消費を抑えて，バッテリ寿命を長時間保つことが重要となる．基幹サーバでは，電力消費を抑えることで発熱量を削減し，熱によるシステムダウンを防ぐことが高信頼性に繋がる．

このような要請に対して，DVFS(Dynamic Voltage and Frequency Scaling) と呼ばれるプロセッサ技術がある．これらは，動作周波数を下げることにより，プロセッサの動作電圧を動的に下げ，低消費電力を実現している．また，プログラムの稼働環境によっては，時間的制約はあるが電力的制約はない，またはその逆も考えられる．つまり，処理時間と消費電力はトレードオフの関係にあり，時間的制約と電力的制約のどちらの観点に基づいて最適化するかは状況により異なる．

本研究では，時間的制約と電力的制約のそれぞれの状況に対応するために，最適化戦略として，(1) 実行時間の増加を許容範囲内に抑えて消費電力量を削減する，(2) 消費電力量を許容範囲内に抑えてできるだけ性能低下を抑える，という 2 つを設定する．そして，DVFS 制御を利用したプロファイル情報に基づく最適化機能をコンパイラに実装し，各最適化戦略をそれぞれ達成することを目的とする．また，プロファイル情報として，タイムスタンプカウンタを利用することで，高い精度で最適化できることを検証する．

本稿の構成を示す．2 章では，関連研究を述べる．3 章では，本最適化の手法を述べる．4 章では，本コンパイラの実装について述べる．5 章では，本コンパイラにおける最適化の評価をおこなう．6 章で本稿をまとめる．

† 電気通信大学 大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications
†† 東京工業大学 大学院情報理工学研究科
Graduate School of Information Science and Engineering, Tokyo Institute of Technology

2. 関連研究

既存のDVFS制御手法として，OSレベルの手法やハードウェアを用いた手法などがある[4)~6)]．OSレベルの手法では，各プロセスの性質から優先度を決定し，その優先度に基づき，プロセス単位でDVFS制御をおこなっている．この方法では，ソースコードレベルでのプログラムの構造を考慮することは難しくなる．ハードウェアを用いた手法では，サンプリングした性能指標に基づき，動作周波数変更後の性能を予測し，適切な周波数を設定し，DVFS制御をおこなっている．また，性能指標には，特別な外部測定装置で取得した電力データや特別なソフトウェアツールを利用することで取得できる性能カウンタなどが用いられてる．この様な方法では，ソースコードレベルでのプログラムの構造を考慮することが難しい．また，特別なツールが必要である．

一方，コンパイラレベルでの静的な手法では，対象とするプログラムの構造を考慮することができ，プログラムに対してより効果的に最適化ができる可能性がある．さらに，プログラム実行の最初の一部分の処理についてプロファイルをおこない，このプロファイル情報を基にコンパイル時に最適化することで，残り部分をプロファイルする必要がなくなる．これにより，実行時のオーバヘッドを抑えることができる．例えば，長時間に渡るシミュレーションにおいて，最初の一部分の処理と残りの部分の処理が同じ挙動を示すプログラムに対しては，特に有効な手段と言える．

3. 低消費電力指向最適化手法

3.1 設計方針

本研究では，シングルプロセス・シングルプロセッサ環境で実行されるプログラムを対象として，DVFS制御による静的な最適化をおこなう．マルチプロセス・シングルプロセッサ環境で実行されるアプリケーションプログラムを対象とするならば，DVFS制御はコンパイラレベルで静的におこなうべきではない．しかし，シングルプロセスとして実行されるプログラムに対しては，実行時のオーバヘッドを抑えられるため，コンパイラレベルでのDVFS制御は有効な手段と言える．また，対象プログラムへの入力データは固定として，本研究を進める．

プロファイル情報は，最適化の基になるデータであり，その精度が重要となる．高い精度のプロファイル情報を取得するためには，取得に伴うオーバヘッドを抑える必要がある．そのため，プロファイル情報として，x86プロセッサが持つタイムスタンプカウンタ(TSC: Time Stamp Counter)[2)]を利用することを考える．TSCは，クロックサイクル毎にインクリメントされ，ユーザプログラムからプロセッサレベルの1命令で読み取ることができる．TSCを用いる利点として，(1) 特別なソフトウェアツールが不要，(2) ライブラリコールを含む複雑な処理が不要，(3) 特別な外部測定装置が不要，という3つが挙げられる．

3.2 最適化戦略

3.2.1 閾値に基づく最適化

本研究では，プログラムの稼動条件による時間的制約と電力的制約のそれぞれの状況に対応するために，最適化戦略として次の2つを設定した．

(1) 実行時間比閾値に基づく最適化
(2) 消費電力量比閾値に基づく最適化

実行時間比閾値とは，ユーザが設定する実行時間の許容範囲を表す値である．実行時間比閾値$T_{threshold}$は，最高周波数での実行時間をT_{max}，最適化後の実行時間をT_{new}とおくと，

$$T_{threshold} = \frac{T_{max}}{T_{new}} \quad (\leq 1) \tag{1}$$

と表すことができる．実行時間比閾値に基づく最適化では，実行時間をユーザにより決められた範囲内に抑えて，消費電力量を削減することを目指す．

消費電力量比閾値とは，ユーザが設定する消費電力量の許容範囲を表す値である．消費電力量比閾値$E_{threshold}$は，最高周波数での消費電力量をE_{max}，最適化後の消費電力量をE_{new}とおくと，

$$E_{threshold} = \frac{E_{new}}{E_{max}} \quad (\leq 1) \tag{2}$$

と表すことができる．消費電力量比閾値に基づく最適化では，消費電力量をユーザにより決められた範囲内に抑えて，できるだけ性能低下を抑えることを目指す．

3.2.2 オーバヘッドを考慮した最適化

本研究では，プロファイリング時と最適化後実行時に生じる2つのオーバヘッドがあり，これらを考慮して最適化をおこなう．

1つ目のプロファイリング時に生じるオーバヘッドとは，プロファイル情報に含まれるプロファイル取得コードの処理時間である．本研究では，取得コードの処理時間をキャッシュミスを考慮した最悪時間で事前に計測する．そして，最適化の前処理として，プロファイル情報から取得コードの処理時間を差し引くことで，その精度を上げる．

2つ目の最適化後実行時に生じるオーバヘッドとは，DVFS制御コードの処理時間である．また，その消費電力量もオーバヘッドとなる．本研究では，OSの機能を利用し，擬似ファイルを介して，DVFS制御を実現する．そのため，コンパイラで挿入するDVFS制御コードはファイルアクセスのための幾つかのシステムコールを含んでいる．本研究では，プログラム全体の処理時間または消費電力量を閾値以上または以下に保つ必要があるため，システムコールを含むDVFS制御コードの処理時間は，無視することはできない．そこで，DVFS制御コードの処理時間をキャッシュミ

スを考慮した最悪時間で事前に計測し，このオーバーヘッドを考慮して最適化をおこなう．これにより，閾値を超えない最悪ケースでの最適化が可能となる．

3.2.3 動作周波数の変更領域の粒度

動作周波数の変更領域の粒度は，先に述べた最適化後実行時のオーバヘッドを考慮して，関数に設定する．プロセッサレベルの命令やソースコードレベルのステートメントのように小さい粒度に設定すると，DVFS 制御コードが頻繁に実行され，オーバヘッドが大きくなる恐れがある．プロファイル取得コードについても頻繁に実行されるとプロファイル情報に与える影響が大きくなる．このため，変更領域の粒度は，ある程度の大きさを持たせる必要がある．

3.3 最適化の流れ

本最適化は，次の 3 つのフェーズで構成される．

最初のフェーズでは，入力ソースコードのコンパイルをおこなう．生成されるオブジェクトコードには，プロファイリングのための処理コードが挿入される．

次のフェーズでは，プロセッサで設定可能な全ての周波数でプロファイリングをおこなう．

最後のフェーズでは，入力ソースコードの再コンパイルをおこなう．パラメタとして最適化の種類と閾値，プロファイル情報の 3 つを指定する．DVFS 制御コードが挿入されたオブジェクトコードが生成される．

以上の 3 つのフェーズにより，最適化処理が施されたオブジェクトコードが生成される．

3.4 プロファイル情報

本研究では関数単位に DVFS 制御をおこなうため，プロファイル情報として，関数の処理コストと関数の処理回数を取得する．

関数の処理コストとして，タイムスタンプカウンタの値を取得する．また，関数の処理コストには，呼び出し先の関数の処理コストを含むものとする．これにより，呼び出し先を含まない場合の関数の処理コストと比較して，プロファイル取得コードの実行回数が少なくなる．プログラム全体の実行時間への影響を抑えることができ，その結果として，高精度な関数の処理コストを取得することができるようになる．

関数の処理回数として，関数間における呼び出し回数を取得する．関数の処理回数を明確にすることで，最適化後の DVFS 制御によるオーバヘッドを予測することが可能となる．これにより，オーバヘッドを考慮して，設定周波数を決めることができ，より高い精度の最適化をおこなえる．また，関数呼び出しの親子関係が分かることにより，関数の処理コストに含まれる呼出先の関数におけるプロファイル取得コードのオーバヘッドを予測することが可能となる．そして，これを関数の処理コストから差し引くことで，最適化に用いる関数の処理コストの精度を上げる．

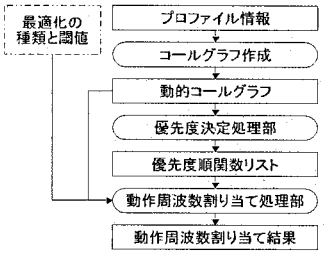

図 1 動作周波数最適化処理の構成

3.5 動作周波数最適化処理

3.5.1 処理の構成

図 1 に，動作周波数の変更対象と設定周波数を決定する動作周波数最適化処理の構成を示す．

プロファイル情報を入力として，コールグラフ作成部が動的コールグラフを作成する．各ノードには関数の処理コスト，各エッジには関数の処理回数を設定する．そして，優先度決定処理部により，検査の優先度の高い順になるように関数リストを作成する．動作周波数割り当て処理部は，ユーザにより指定されるパラメタ（最適化の種類と閾値）に従い，関数リストを先頭から順に検査し，動作周波数の変更領域と設定周波数をアドホックに決定し，結果をファイルに保存する．

検査の優先順位決定処理と動作周波数割り当て処理について，次のそれぞれの項で説明する．

3.5.2 検査の優先順位決定処理

次項で述べる動作周波数割り当て処理のために，割り当て処理における検査の優先度を決定する．呼出先を含まない処理コストを計算し，そして，処理コストが大きい関数には高い優先度，処理コストが小さい関数には低い優先度を設定する．

プロファイル情報の関数の処理コストには，呼出先の処理コストが含まれている．そこで，まず最初に，動的コールグラフを利用して，呼出先を含む関数の処理コストから呼出先を含まない関数の処理コストを計算する．これと同時に，関数の処理コストからプロファイル取得のオーバヘッド (最悪ケースでの予測値) を差し引く．次に，呼出先を含まない処理コストが降順になるように，関数リストを作成する．

3.5.3 動作周波数割り当て処理

動作周波数割り当て処理は，2 段階の処理に別れている．最初の段階では，プログラム全体に対して周波数を割り当てる．次の段階では，閾値により近づけるように，各関数に周波数を割り当てていく．

次に，消費電力量比閾値に基づく最適化を例にして，動作周波数割り当て処理を示す．実行時間比閾値に基づく最適化と消費電力量比閾値に基づく最適化では，処理内容が異なるため，実行時間比閾値に基づく最適化については括弧で示している．

Step 1 プログラム全体の消費電力量（実行時間）が

閾値以下（以上）になるように，できるだけ低い周波数を設定する．消費電力量を削減するためには，多くの時間を低い周波数で処理することが必要になる．このため，プログラム全体に対して低い周波数を設定する．

Step 2 閾値により近づけるために，プログラム全体の消費電力量（実行時間）を閾値以下（以上）に保ちつつ，さらに高い周波数（低い周波数）を各関数に設定する．また，DVFS 制御コードの挿入箇所の増加を抑えるために，呼出元の関数が変更対象である場合は，その関数は変更対象にならないものとする．ただし，呼出元の関数がプログラムのエントリポイントの場合は除く．

以上の処理により，動作周波数の変更領域と設定周波数を決定し，閾値を超えない最適化をおこなう．

4. COINS を利用したコンパイラの構築

本コンパイラの構築には，COINS(COmpiler INfraStructure)[1] を用いた．COINS とは，新しいコンパイラ方式を容易に構築，評価できる共通インフラストラクチャである．

図 2 に，本コンパイラの構成を示す．本研究で開発した新しい処理部を灰色で示している．本研究では，入力として C のソースコード，ターゲットプロセッサとして x86 プロセッサを想定している．C の解析器 (C Analyzer) が C のソースコードを高水準中間表現 (HIR) に変換し，その後，HIR 変換器 (HIR to LIR) が低水準中間表現 (LIR) に変換する．そして，コード生成部 (x86 Code Generator) が LIR からオブジェクトコード (Optimized Code) を生成する．

本研究で開発したプロファイル取得コード挿入処理部 (PProf) と動作周波数最適化処理部 (ReBuild) は，それぞれに対応するオプションがユーザから指定された場合に HIR 処理部として動作する．PProf では，プロファイル取得コードを HIR 上の各関数に挿入する．ReBuild では，最初に，3.5 節で述べた動作周波数最適化処理をおこなう．そして，動作周波数最適化処理の結果（動作周波数の変更領域と設定周波数）に基づいて，DVFS 制御コードを HIR 上の各関数に挿入する．

5. 評　　価

5.1 実験方法

評価実験は，PentiumM 733 プロセッサ（32+32KB L1 キャッシュ，2MB L2 キャッシュ，400MHz システムバスクロック）を搭載する計算機上でおこなう．オペレーティングシステムは Linux kernel 2.6.17 である．PentiumM 733 プロセッサでは，5 段階の周波数を設定することができ，周波数と電源電圧の関係は **表 1** のようになっている．

ベンチマークには，組込みシステム用ベンチマーク MiBench[3] より，FFT, JPEG, LAME（MP3 エンコーダ）の 3 つを用いた．各ベンチマークの入力データには，それぞれ 4.1M 個の単精度の実数データ，5.7MB の PPM 形式の画像ファイル，49.3MB の WAV 形式の音声ファイルを用いる．

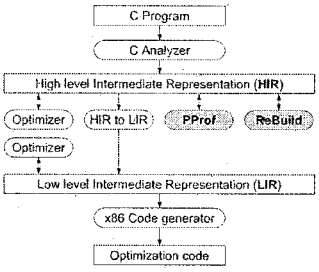

図 2 低消費電力指向最適化コンパイラの構成 (COINS ホームページの図を基に作成)

表 1 PentiumM 733 における周波数 [GHz] と電圧 [V] の関係

周波数	1.1	1.0	0.9	0.8	0.6
電源電圧	1.004	0.988	0.972	0.956	0.844

表 2 実行時間とプロファイリング時間 (秒) の比較

Benchmark	Before	After	Number
FFT	37.10	37.10	4,194,311
JPEG	0.65	1.32	4,145,838
LAME	54.60	57.99	89,011,480

これらのベンチマークのコンパイルには，本最適化を実装した COINS version 1.3.2.2 を用いる．また，ベンチマークのソースコードと共に提供されている Makefile を利用する．実行時間の測定は，time コマンドを用いる．FFT と LAME は 10 回ずつ，JPEG は 100 回実行し，その実行時間の算術平均で求めた．消費電力量の測定は，各動作周波数での処理時間を計測し，表 1 を用いて，理論値で求めた．

5.2 実行時間とプロファイリング時間の比較

プロファイル取得コードがプログラム全体の実行時間に及ぼす影響を評価する．**表 2** に，プロファイル取得コードの挿入前と挿入後における最高周波数での実行時間を示す．左から 1 列目はベンチマーク名，2 列目と 3 列目には，それぞれプロファイル取得コードの挿入前 (Before) と挿入後 (After) の実行時間を示している．4 列目には，プロファイル取得コードの実行回数 (Number) を示す．

表 2 の結果より，FFT と LAME のプロファイル取得コードによるオーバヘッドは，挿入前の実行時間に対して，それぞれ 0.0%と 6.3%であり，小さく抑えられていることがわかる．一方，JPEG では，103.1%となり，プロファイリング時間の約半分を取得コードの処理時間が占める結果となった．

JPEG のオーバヘッドが大きくなった原因は，ベン

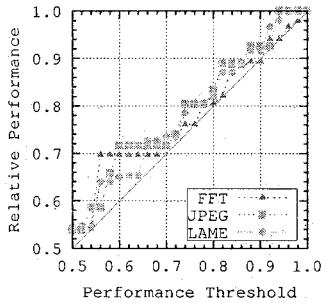

図 3　実行時間比閾値に基づく最適化における実行結果

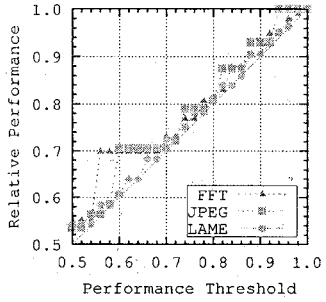

図 4　実行時間比閾値に基づく最適化における予測結果

表 3　最高周波数に対する実行時間比閾値毎の消費電力量比

	FFT	JPEG	LAME
0.90	0.960	0.989	0.959
0.80	0.925	0.917	0.935
0.50	0.701	0.716	0.709

チマークの実行時間が 1 秒未満と小さかったためである．しかしながら，関数が 4,145,838 回処理されているにも関わらず，プロファイル取得コードによるオーバヘッドは 0.67 秒と非常に小さい．

以上のことから，本研究におけるプロファイリング方法は，システムコールを利用せず，タイムスタンプカウンタを用いたことによって，オーバヘッド時間を非常に小さく抑えることができたと考えられる．

5.3　実行時間比閾値に基づく最適化

図 3 に，最高周波数での実行時間に対する閾値毎の相対比を示す．横軸は実行時間比閾値，対角線（実線）は最適値を示している．同様に，コンパイラによる予測結果を**図 4** に示す．このときの消費電力量の相対比を**表 3** に示す．

図 3 の結果より，FFT の一部を除いて，最適値より高い性能を達成できていることがわかる．また，最適値に対して平均で，FFT では 3.5%，JPEG では 4.4%，LAME では 3.6%の誤差に抑えることができている（最適値より低い場合を除く）．

FFT の結果について考察する．FFT は 5 個の関数から構成される．閾値 0.98, 0.90 のとき，それぞれ 0.1%, 0.6%の性能低下となった．共に，1%未満であり，測定誤差と考えられる．最適値に対する最大誤差は 13.7%（閾値 0.56）となった．誤差が大きくなった原因として，周波数変更対象となる関数が 4 つのみで少ないことが考えられる．例えば，ある関数の周波数を 1 段階下げると閾値より低い性能となる場合，高い周波数が設定される．ただし，このとき，他に対象となる関数があれば，閾値に対する誤差は小さくなる可能性がある．

JPEG の結果について考察する．JPEG は 120 個の関数から構成される．全ての閾値において最適値より高い性能を達成できている．最適値に対して，最大で 11.6%（閾値 0.60）の誤差となった．図 4 の予測結果より，閾値 0.7 から 0.6 までの動作周波数割り当て結果は，全て同様になっていることがわかる．このことから，誤差が大きくなった原因として，動作周波数割り当て対象の関数がなかったことが考えられる．プログラム全体の実行時間が 1 秒未満と小さいために，DVFS 制御コードを挿入することで，最適値を超えてしまう予測結果になり，対象となる関数がなかったと思われる．

LAME の結果について考察する．LAME は 123 個の関数から構成される．全ての閾値において最適値より高い性能を達成できている．最適値に対する最大誤差は 8.0%（閾値 0.56）となった．この最大誤差の原因として，図 4 の予測結果では最適値に近い結果を得ていることから，DVFS 制御コードの処理時間を最悪時間で計算していることが考えられる．ただし，ユーザが指定する閾値を下回ることなく，ユーザの要望を満たすことができている．

表 3 の結果より，実行時間比閾値に基づく最適化では，閾値 0.90, 0.80, 0.50 のとき，1.1%から 29.9%の消費電力量を削減することができた．

5.4　消費電力量比閾値に基づく最適化

図 5 に，最高周波数での消費電力量に対する閾値毎の相対比を示す．横軸は消費電力量比閾値，対角線（実線）は最適値を示している．同様に，コンパイル時の予測結果を**図 6** に示す．このときの実行時間の相対比を**表 4** に示す．ただし，PentiumM 733 プロセッサにおいて，最高周波数での消費電力量に対する最低周波数での相対比が 0.707 となり，理論上はこれが限界値である．そのため，後述する誤差の計算では，閾値 0.7 以下の結果については考慮しない．

図 5 の結果より，FFT の一部を除いて，最適値より低い消費電力量に抑えられていることがわかる．また，最適値に対して平均で，FFT では 4.5%，JPEG では 4.5%，LAME では 3.1%の誤差に抑えることができている（最適値より高い場合を除く）．

FFT の結果について考察する．4 通りの閾値 (0.74, 0.76, 0.92, 0.94) で，消費電力量が最適値を超える結果となった．共に 1%未満の増加であり，測定誤差と

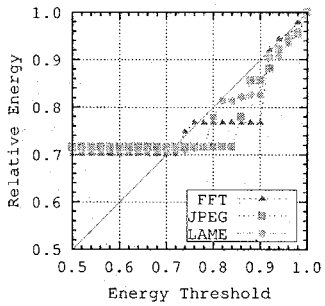

図 5　消費電力量比閾値に基づく最適化における実行結果

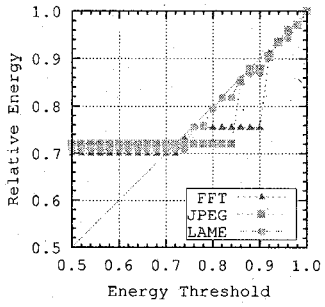

図 6　消費電力量比閾値に基づく最適化における予測結果

表 4　最高周波数に対する消費電力量比閾値毎の実行時間比

	FFT	JPEG	LAME
0.92	0.753	0.697	0.729
0.82	0.592	0.538	0.602
0.72	0.550	0.538	0.544

考えられる．最適値に対して，最大で 13.2%（閾値 0.90）の誤差となった．

JPEG と LAME の結果について考察する．全ての閾値において，最適値より低い消費電力量に抑えられている．最適値に対して，最大で，JPEG では 12.4%（閾値 0.84），LAME では 7.4%（閾値 0.90）の誤差となった．全体的な傾向として，必要以上に消費電力量を抑えていると言える．特に，LAME の 6 通りの閾値 (0.76, 0.78, 0.86, 0.88, 0.90, 0.94) においては，最適値に対して，平均 3%程度低い．一方，図 6 の予測結果では，最適値に近い結果を得ていることがわかる．例えば，LAME において，閾値 0.78 のときの DVFS 制御コードにおける消費電力量は，最高周波数でのプログラム全体の消費電力量に対して，予測では 2.2%，実行時では 0.8%であった．このことから，最悪ケースで予測していることが，最適値との差を大きくした要因として考えることができる．ただし，ユーザにより指定された閾値は，超えておらず，ユーザの要望は満たしている．

表 4 の結果より，消費電力量比閾値に基づく最適化では，閾値 0.92, 0.82, 0.72 のとき，24.7%から 46.2%の速度低下に抑えることができた．

6. おわりに

本研究では，プロセッサの省電力化を目的として，(1) 実行時間比閾値に基づく最適化，(2) 消費電力量比閾値に基づく最適化，という 2 つの最適化方針を設定した．各最適化方針は，プログラムの稼動条件による時間的制約と電力的制約のそれぞれの状況に対応するものである．

DVFS 制御を利用したプロファイル情報に基づく最適化をコンパイラに実装し，評価した．プロファイル情報には，タイムスタンプカウンタを利用した．コンパイラでおこなった最適化処理では，まず最初に，プログラム全体に対して，閾値を超えないように周波数を設定する．そして，さらに閾値に近づくように，プログラムの部分的（関数単位）に周波数を設定していく．

最適化方針 (1) については，ユーザが指定する実行時間比閾値に対して，平均で，FFT では 3.5%，JPEG では 4.4%，LAME では 3.6% の誤差に抑えられることを示した．また，その結果として，消費電力量を削減できることを示した．

同様に，最適化方針 (2) については，ユーザが指定する消費電力量比閾値に対して，平均で，FFT では 4.5%，JPEG では 4.5%，LAME では 3.1% の誤差に抑えられることを示した．また，その結果として，ある程度の性能低下に抑えることができた．

今後の課題として，最適化精度向上のために，対象関数を限定したプロファイル機能を実装しプロファイル情報の精度を上げる，プロファイル取得コードと DVFS 制御コードの処理コストの予測方法の再検討などが挙げられる．

参考文献

1) COINS Project: www.coins-project.org/.
2) Intel Corp.: *Intel 64 and IA-32 Intel Architecture Software Developer's Manual* (2006).
3) MiBench: www.eecs.umich.edu/mibench/.
4) 堀田義彦，佐藤三久，木村英明，松岡　聡，朴　泰祐，高橋大介：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，情報処理学会論文誌コンピューティングシステム，Vol.47, No.SIG12(ACS 15), pp.272–284 (2006).
5) 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，Vol.47, No.SIG 18, pp. 80–91 (2006).
6) 宮川大輔，石川　裕：電力制御スケジューラのプロトタイプ実装，情報処理学会研究報告 2006-OS-103，Vol.2006, No.86, pp.109–115 (2006).