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

2006－OS－103 (15)
2006／8／1

電力制御スケジューラのプロトタイプ実装

宮川大輔† 石川 裕†

計算機の消費電力低減が重要な課題となっている現在，OS レベルの電力制御手法が必要である．本論文では OS レベルの電力制御手法として，電力制御スケジューラ PCCS のプロトタイプを実装する．PCCS では，OS は各プロセスの実行状況を監視し，そのプロセスに最適な動作速度を動的に決定する．CPU の動作周波数は実行プロセスの動作速度に基づいて変更される．単純な Web サーバにおける実験の結果，本プロトタイプが応答性能を下げずに消費電力の増大を既存実装の 36%に抑え，手動設定による理想的なケースとも遜色がないことが示された．

Prototype of Process Consumption Controling Scheduler

Daisuke Miyakawa† and Yutaka Ishikawa†

Reducing power consumption of computers is one of the most important issues. OS level power consumption control is necessary. As one of those methods, we develop the prototype of the PCCS (Power Consumption Controlling Scheduler). In PCCS, OS monitors the execution state of each process and automatically decides it's operating speed. The frequency of CPU is changed in accordance with that information. Experiments using a simple Web server showed that the prototype reduces the power consumption to 36% without lowering interactiveness. Those experiments also showed that the prototype is equivalent to the ideal case in which the frequency of each process isset by hand.

1. はじめに

組み込みシステムから大規模並列計算機に至るまで，消費電力を下げつつ効率良く処理を実行する機構の実現が急務となっている．

プロセッサの低消費電力化のために DVFS (Dynamic Voltage and Frequency Scaling) と呼ばれる技術が広く研究，利用されている．これはプロセッサの動作電圧を稼働時に動的に下げることによってプロセッサの消費電力を下げる技術である．CMOS 半導体の制約により，動作電圧を下げた際にはプロセッサの動作周波数も並行して下げなければならないため，動作速度と消費電力はトレードオフの関係となる．この技術は Intel 社の SpeedStep[4] や AMD 社の PowerNow![3]，Transmeta 社の LongRun[5] 等，すでに一般のプロセッサに実装されている．

プロセッサの消費電力を下げることは，計算機の消費電力を下げるうえで重要である．Thinkpad X31 (Pentium M 1.6GHz 搭載) における消費電力傾向を表 1 に示す．液晶ディスプレイを比較対象に取り上げたのは，それが近年の PC において消費電力が大きいと考えられるデバイスだからである．実験では負荷として図 1 に示すプログラムを実行し，20 秒間の平均消費電力を測定した．

```
int main(){
  int i;
  while( 1 ) i++;
  return 0;
}
```

図 1 疑似負荷プログラム

表 1 周波数と液晶ディスプレイによる消費電力の違い (W)

周波数 (液晶)	idle	負荷
600MHz(なし)	38.5	40.2
600MHz(あり)	44.4	46.7
1600MHz(なし)	42.9	58.0
1600MHz(あり)	48.6	64.3

結果から以下の二点を読みとることが出来る．

- 液晶ディスプレイの有無による消費電力よりもプロセッサの動作周波数の影響が大きい
- 負荷の有無に関わらず CPU の動作周波数によって消費電力が変化する

本研究では DVFS を利用し，低消費電力かつ応答

† 東京大学
The University of Tokyo

性能の高いシステムを構築することを目標とする．汎用 OS においては，応答性能を要求されるプロセスと応答性能を要求されないプロセスが混在する．例えば Web サーバでは，サーバプロセスが散発的なアクセスを高速に処理をする必要がある一方，検索インデクスのアップデートといった応答性能の必要がないバッチプロセスも実行されている．このような環境では必要なときにのみ速度を上げ，それ以外では速度を下げて消費電力を減らすことが期待される．

この目標を実現するための制御手法には以下の 3 つが考えられる．

- ハードウェアレベル制御
- ソフトウェアレベル制御
- OS レベル制御

ハードウェアレベル，ソフトウェアレベル制御の研究例については第 6 章で紹介する．

本研究で著者らは OS レベル制御を提案する．OS レベル制御では，ハードウェアの支援を受けつつ OS 固有の情報を消費電力制御に用いる．ハードウェアが適切な API を提供していれば，他の 2 つの手法よりはるかに多くの情報を電力制御に用いることが出来る．本研究の場合，DVFS がこの API に相当する．

本研究で目指すのは，OS レベル制御手法の中でも OS 内部のプロセスの情報を電力制御に用いる手法である．従来手法ではプロセッサのコア単位で電力を制御しているが，プロセッサ上で動作する各プロセスの速度要求は，本来プロセッサの物理的性質とは独立に判定されるべきものである．速度を要求するプロセスを高速に動作させ，速度を要しないプロセスを低速化することで，不要な電力消費を抑止することが出来る．

著者らは論文 14)，16) で，プロセス単位電力制御機構を Linux 上に実装し，OS レベル制御の有用性についての評価を行なった．評価の結果，Pentium M であれば数%という比較的低オーバーヘッドでのプロセス単位電力制御が可能であることが示された．この機構を用いることで，低消費電力かつ応答性能が高いシステムを構築することが出来る．応答性能を要求されるプロセスの動作速度を上げ，その他のプロセスの動作速度を下げれば良い．

プロセス単位電力制御機構では，各プロセスの動作周波数をユーザが手動で設定する必要がある．しかし，全てのアプリケーションの周波数をユーザがその度に手動で指定することには限界がある．また，各プロセスに期待される応答性能は，そのプロセスの親によっても変化する．例えばユーザがコンソールから起動したプロセスと，cron のようなバッチ処理を行なうデーモンから起動したプロセスでは，同じプログラムを実行しているのであっても応答性能に対する要求は異なる．ユーザが手動で設定するのではなく，OS 自身が自動的に各プロセスの性質を判別し，動作速度を設定出来ることがより望ましい．

筆者らは論文 17) で，各プロセスに最適な動作周波数を自動的に設定する低消費電力スケジューリングアルゴリズムを提案した．このアルゴリズムは，親プロセスの優先度，I/O の性質，CPU の利用状況の三種類のパラメータをプロセスから抽出することで，応答性能に影響のあるプロセスと影響のないプロセスを動的に予測する．これにより，ユーザの手動設定なしに高応答性能かつ低消費電力なシステムを構築可能とする．

本研究では上記の提案を元にした PCCS (Power Consumption Controlling Scheduler) のプロトタイプを Linux 上に実装する．PCCS は，各プロセスの動作周波数を実行状況から動的に見積もることで，ユーザの手動設定に頼らずに高応答性能な低消費電力システムを実現する．本論文では，このプロトタイプを実装した PCCS プロトタイプを，既存実装および手動設定による理想的なケースと比較する．評価には応答性能を要求するプロセスとそうでないプロセスが混在する仮想的な Web サーバを用いる．実機による実験により，本プロトタイプが既存実装より優れ，手動設定による理想的なケースとも遜色のないことを示す．

2. PCCS プロトタイプの設計と実装

PCCS は，論文 14)，16) で提案したプロセス単位電力制御機構を元に，自動的に各プロセスの電力制御ポリシーを見積もる実装を加えたものである．実装環境には Linux Kernel 2.6.16.20 を用いた．以下では PCCS の設計と実装について概略を述べる．

DVFS を利用するため，本研究では Linux Kernel の cpufreq モジュールを用いた．cpufreq は DVFS を制御するための Kernel モジュールである．cpufreq には governor と呼ばれる機構が用意されており，ユーザは governor 実装のうち一つを選択することができる．DVFS の電力制御は選択された governor に従って行なわれる．図 2 に cpufreq と governor の関係を示す．ユーザは cpufreq に対して sys ファイルシステムに電力に関する要求を書き込む．governor はこの書き込みを解釈し，DVFS のドライバを経由してハードウェアに動作周波数変更の要求を送る．

既存の governor を表 2 に示す．ondemand はカーネル内部のタイマと CPU 負荷を保存する構造体を用いて定期的に CPU 負荷を調べ，その負荷に応じて CPU の

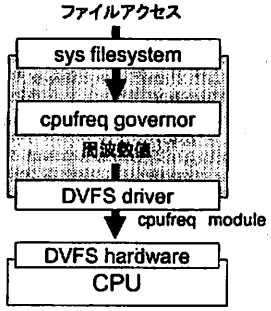

図 2 cpufreq モジュールの構造

表 2 既存の cpufreq governor

名称	効果
performance	周波数最大に固定
powersave	周波数最小に固定
ondemand	定期的に CPU 負荷を調べて周波数を変更
conservative	ondemand と同様，変更頻度が少ない

動作周波数を変更する．conservative は DVFS のオーバヘッドが大きい CPU のための実装で ondemand とアルゴリズムは類似している．

本論文では手動設定によるケースに加え，これら既存の governor 実装も PCCS の比較対象とした．本研究では PCCS に対応した governor を新たに追加し，スケジューラ等のソースコードも一部修正する．

DVFS のオーバーヘッドは十分小さいものとする．論文 16) で示した通り，Pentium M 上で標準の優先度の計算処理を同時実行した場合，オーバーヘッドは最大でも 3%程度である．

各プロセスの優先度を低く設定すると，システム全体の性能が下がる可能性がある [17]．Linux は各プロセスの優先度に応じて各プロセスに与えられるタイムスライスを動的に変更しており，低優先度のプロセス同士では，高い頻度でコンテクストスイッチが発生するためである．プロセスの優先度と与えられるタイムスライスの関係を図 3 に示す．本研究では実験に用いるプロセスは全て nice 値を 0 に固定しており，論文 17) で指摘される現象は起こらない．

本プロトタイプ PCCS では，動的に各プロセスの負荷を見積もるためのパラメータとして Linux Kernel 2.6 の内部変数 sleep_avg を用いる．sleep_avg は各プロセスが持ち，そのプロセスが I/O 待ち等のスリープ状態であるときには増加し，逆に CPU リソースをプロセスが使用した場合には減少する．最大値は 1 秒に相当し，それ以上 sleep している場合にも sleep_avg は増大しない．Kernel 空間で正確にプロセスのスリープ時間の平均をとることは難しいため，これはあくまで目安としての値である．名前から想像される「平均値」を表すわけではない．Kernel 内部ではこの値を動的優先度の決定の際に使用している．

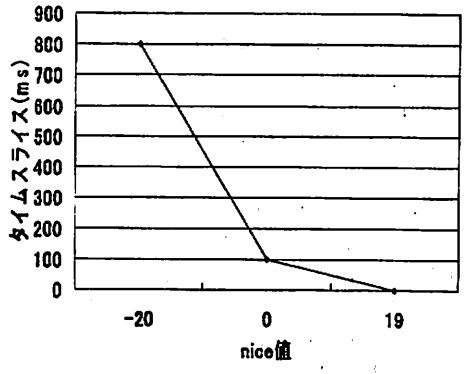

図 3 nice 値とタイムスライスの関係

PCCS はプロセス切替の度に次プロセスの sleep_avg メンバを参照し，その値を元に周波数を決定する．今回の実装では sleep_avg を線形にスケールさせ，sleep_avg が最小値のとき最低周波数に，sleep_avg が最大値のとき最高周波数になるようにした．本研究で使用する Thinkpad X31 はそれぞれ 600MHz と 1600MHz が最低周波数と最高周波数となる．例えば sleep_avg が最大値の 80%の値になっていた場合，計算式 $600 + (1600 - 600) \times (80/100) = 1400$ より，そのプロセスの周波数は 1400MHz になる．

DVFS で利用できる周波数は離散値を取る．例えば今回用いる PC では，600MHz から 200MHz 刻みで周波数を変更することが出来る．sleep_avg から決まる値をスケールさせる方法として，今回は値を切り上げる実装を用いる．

論文 17) で著者らは Linux Kernel 2.6 の既存実装を本研究に用いる際の問題点を指摘した．sleep_avg はプロセスに関して比較的短い期間の情報しか反映しておらず，I/O によって短期間のスリープを頻繁に行なうバッチプロセスを，応答性能を必要とするプロセスであると誤認識する問題である．本論文ではこの現象が実際に観測されるかを実験で調べる．

3. 評価方法の検討

PCCS が有効と考えられるのは，応答性能を必要とするプロセスと必要としないプロセスが混在し，低消費電力化が重要と考えられる環境である．また応答性能を必要としないプロセスの時間制約が緩やかで，動

作速度を下げても問題がないことも条件の一つである．応答性能を必要としないプロセスを最低電圧で実行しても時間制約を満たすと仮定する．

考えられる環境はいくつかある．例として，バッテリー駆動のラップトップPC上でプログラミングするケースを考える．ソースコード内を検索するバッチジョブをバックグラウンドで実行させつつ，エディタ上でプログラミングを行なう．このとき，バッチジョブの実行を若干遅らせてでも，バッテリーの残り容量を優先させたい．一方で，検索を行なっている状況でもエディタやシェルの応答性能は下がって欲しくない，というケースが存在する．

またある種のWebサーバもPCCSが有効なシステムである．Webサーバ上で定期的，不定期的を問わずサーバプロセスを稼働させつつ，その背後でサービスに用いるデータを更新するためのバッチジョブを走らせるというケースは現実に存在する．この際，デッドラインを越えなければ，バッチジョブは多少終了が遅くなっても良い．むしろそのバッチジョブを遅く動作させてでも消費電力を削減するべきである．

前者の評価はユーザの動きが直接関係するため難しい．本論文では後者のサーバに関する状況をエミュレートすることで，プロトタイプ実装の有効性を評価する．バッチジョブの電力消費を抑えつつ，Webサーバの応答性能が下がらないシステムであることが望ましい．

本論文では各プロセスの応答性能に関する要求は起動後変化しないものとする．すなわち，Webサーバは常に応答性能を要求され，バッチジョブは常に低速で動いて良い．手動による制御は事前にこのポリシーに基づいて各プロセスの電力を制御しているため理想の結果を示す．PCCSおよび既存実装はそれが知らされていない状況で動的に計算機の電力を制御する．よって本論文の評価では，PCCSおよび既存実装は，手動による電力制御の結果に近いほうが良い実装であると判定される．

4. 実験方法

実験には3台のPCを使用する．以下では3台を各PCの機能にあわせてそれぞれserver，client，measurementと呼ぶこととする．本実験で用いる環境を表3に，使用するアプリケーションを表4に示した．

serverではWebサーバとしてApacheが常時稼働している．また，serverは定期的にバッチプロセスを走らせる．今回の実験ではこのバッチプロセスに図4に示すスクリプトを用いた．ここで用いるprelinkは

表3 本実験に使用した計算機

用途	server	client
計算機名	IBM 社製 Thinkpad X31	NEC 社製 MY30V/C-F
CPU	PentiumM 1.6 GHz	Pentium4 3.0GHz
メモリ	1GB	512MB
OS	Debian GNU/Linux 3.1	Debian GNU/Linux 3.1
Kernel	2.6.16.20	2.6.16.20
コンパイラ	gcc 4.0.4	gcc 4.0.4

表4 本実験に使用したコマンド

プログラム名	Apache	prelink
バージョン	2.0.54-5	0.0.20060522-1

```
#!/bin/bash
prelink -ua
prelink -a
```

図4 prelinkによるバッチスクリプト

表5 serverにおけるprelinkの結果

動作周波数 (MHz)	平均消費電力 (W)	所要時間 (秒)
600	41.9	127.1
800	42.8	112.4
1000	44.6	103.2
1200	46.6	98.4
1400	50.2	95.2
1600	52.5	93.6

各プログラムの動的リンクに関する情報を書き換え，プログラムの起動速度を上げるコマンドで，実行ファイルを書き換えることに伴い，I/O処理が多い．速度を均質化するため，prelinkは-uaオプションで実行後，-aオプションをつけて再度実行する．-aオプションは全てのプログラムを対象にすることを意味し，-uオプションは対象となったプログラムをprelink実行前の状態に戻すことを意味する．このスクリプトを単独で実行した場合の消費電力を表5に示す．

serverは応答性能を要求されないprelinkと応答性能を要求されるApacheが共存した状況にある．prelinkの動作速度を下げて消費電力を抑えるとともに，ApacheへのWebアクセスの応答速度が下がらないことが期待される．

clientはserverに対してApache Benchを用いてWebアクセスを行ない，応答性能を評価する．実験に用いたコマンド行を図5に示す．オプションは，リクエスト数を10000，並列数を10を表す．ここで使われるindex.htmlのサイズは1457Bytesである．結果はApache Benchの算出するTransfer rate (KB/sec)

```
ab -n 10000 -c 10 target/index.html
```

図 5 Web アクセスプログラム

表 6 消費電力と平均応答性能

	消費電力量 (W 秒)	平均応答性能 (KB/sec)
powersave	74262.250	4867.114
performance	91404.325	12057.544
ondemand	83420.243	12036.066
conservative	84850.351	9555.217
ideal	75599.064	12557.551
PCCS	77585.875	12440.336

を用いる．

measurement は server の消費電力の変化を常時監視する．電力の測定にはシナジェティック社の ST-33100 を用いる．これは計算機に供給される電圧と電流を入力電源から取得して，測定用計算機に測定結果を送る装置である．measurement は ST-33100 から届く値を元に消費電力を計測する．

電力計測は各ケースで 30 分間行なった．全ての実験において，この時間内に起こるバッチスクリプトの実行回数と，client から server へ行なわれる Apache Bench の負荷回数は同一とする．30 分あたりのバッチプロセスの起動回数は 12 回，Apache Bench の起動回数は 60 回である．

実験は，手動によって各プロセスに最適な周波数を設定した理想な結果，PCCS プロトタイプによる結果，表 2 に示した governor を用いた結果の計 6 種類に対して行なわれた．以降では手動設定による最適な結果を ideal と書く．

5. 実験結果

結果を表 6 に示す．また，消費電力量順にソートした結果を図 6 に，平均応答性能順にソートした結果を図 7 に示す．ideal における消費電力量に対する平均応答性能を 1 としたときの各テストケースの結果を図 8 に示す．

図 8 より，本論文で実装した PCCS プロトタイプが消費電力対応答性能で ideal に最も近い結果を出せていることが分かる．以下では表 6 に記載された順に各テストケースの結果について述べる．

powersave の電力消費がもっとも少ない理由は明らかである．一方で応答性能も最大値の 39%足らずに落ちている．全てのプロセスの動作周波数が最低なため，Apache の応答性能が著しく下がったものと考えられる．

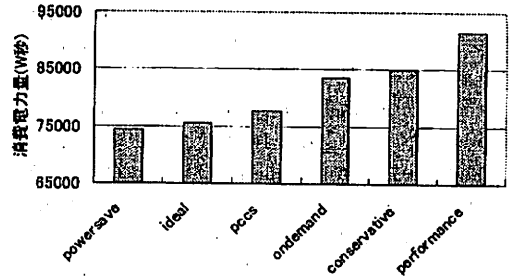

図 6 消費電力量順の結果

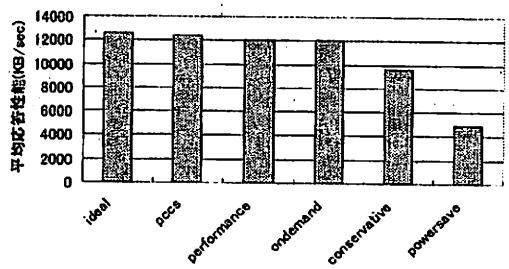

図 7 平均応答性能順の結果

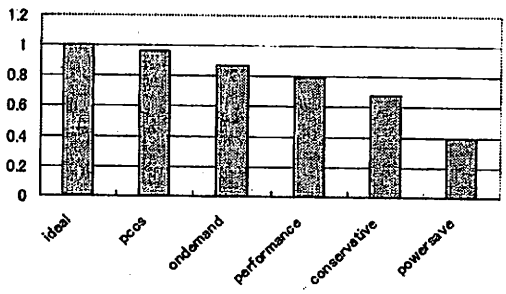

図 8 ideal の電力性能対応答性能を 1 としたときの比較

performance は応答性能は良いが消費電力が最も大きい．ほぼ同等の応答性能を示した ideal が powersave より約 1300W 秒しか余分に消費していないのに対して performrmance は約 17000W 秒と，10 倍以上余分に電力を消費している．計算機全体で見ても，ideal が powersave に対して電力量を 1.8%増大させているだけに対して，performance は 23.1%の電力量増大を招いている．

ondemand は ideal と同等の応答性能を示すものの，消費電力が依然として高い．応答性能は ideal の 96%と性能低下は比較的小さいが，計算機全体で見て 12.3%の電力増大を招いている．prelink の CPU 負

荷を Apache の負荷と区別できず，全て一律に動作速度を上げたことによって，ideal よりも余分に電力を消費したものと見られる．

conservative は，応答性能が ideal の 76%と低い上，計算機全体の消費電力を 14.3%増大させており，明らかに効率が悪い．ondemand と比較して CPU 負荷を計測する間隔が長いため，バッチプロセス prelink の影響を ondemand より大きく受けてしまっている．また，応答性能を要求されるような瞬間的な負荷に対して ondemand は反応できないため，Apache の応答性能が低下していると考えられる．

ideal は performance より 4%高い応答性能を示しており，想定通り Apache を performance 並に高速に動作させている．ただし，この結果だけからは ideal の方が応答性能に優れているとはみなせない．消費電力は powersave に比べて 1.8%増えているが，これは Apache を高速に動作させるのに必要な最低限の消費電力と考えられる．

PCCS プロトタイプは ideal と応答性能で 1%しか差がない．powersave と比較したときの電力量の増大は，ideal の 1.8%に対して PCCS プロトタイプでは 4.5%と若干大きくなっているが，ondemand の 12.3%と比較すると明らかに減少している．ondemand の powersave に対する電力増大量は 9158.0W 秒，それに対する pccs プロトタイプの電力増大量は 3323.6W 秒であるため，pccs プロトタイプは既存実装の 36%まで電力増大を抑えていることになる．これは既存実装に対する十分な改善と言える．ondemand の応答性能と PCCS プロトタイプの応答性能はほぼ同一，あるいはわずかに PCCS プロトタイプ側の方が良い．本実験において，PCCS プロトタイプは既存実装より明らかに優れている．

論文 17) で著者らは，既存の Linux 実装は比較的短い区間の情報しか参照しておらず，ファイル I/O の多いコマンドを応答性能を必要とするプロセスと認識してしまうと主張した．これは本論文に関して言えば，I/O の多いプロセスの sleep_avg が意図せず大きくなってしまうことを意味する．prelink コマンドはファイルシステム上の実行ファイルを実際に書き換えるため，I/O の多いバッチプロセスの一つとみなせる．論文 17) の主張の通りであれば，今回のような実験では prelink は応答性能を必要とするとみなされ，省電力効果は不十分なものとなるはずである．

しかし本実験の結果によると，PCCS プロトタイプは prelink コマンドを低速に動作すべきプロセスとみなして消費電力を下げており，結果として消費電力量対応答性能の面で良い結果が得られている．論文 17) の予想との乖離について，今後検討する必要がある．

6. 関連研究

DVFS を用いた低消費電力化の研究，実装は数多い．DVFS でを用いて低電力化を実現するための手法には，1 で記述したように以下の 3 つが考えられる．

- ハードウェアレベル手法
- ソフトウェアレベル手法
- OS レベル手法

DVFS の既存研究はハードウェアレベル，ソフトウェアレベルの手法に分類することが出来るものが占める．

ハードウェアレベルの手法として，例えば論文 12), 15) などがある．電力制御に関する指標を前もって静的にプログラムに組み込んでおき，その情報を参考にして消費電力を制御する手法もある [2),6)~11),13)]．

ハードウェアレベルの手法では，プロセッサの負荷情報を元にハードウェアが動作電圧を変更する．そのため，プロセッサ内部の局所的な情報を用いたアルゴリズムには適しているが，OS 内のプロセスの意味を個別に追うことは出来ない．特に，本研究が提案するようにプロセス毎の情報を動的に取得してアルゴリズムに利用することは不可能である．

ソフトウェアレベルの手法として，例えば論文 6) は，手動で電力を制御した場合とそれ以外の自動制御手法における省電力性能について評価した．また，ユーザレベルから電力を制御するアプリケーションとして Cpufreq Daemon[1)] が広く利用されている．これはシステムに電力管理用デーモンを稼働させ，ユーザレベルから疑似ファイル経由で DVFS 機構を呼び出すことで CPU の動作周波数を制御し，電力制御を行なうものである．

ソフトウェアレベルの手法では，OS 上で実行しているプロセスの実行状況をユーザ空間のプロセスが監視し，電力を制御する．監視するプロセス自体もユーザ空間にあるため，OS が公開する API の範囲でしか電力を制御出来ず，スケジューラの干渉により粒度の細かい制御は出来ない．前者の問題は公開する API を増やすことによって対処できるが，後者の問題は本質的に不可避である．

ハードウェアレベルの手法，ソフトウェアレベルの手法共に，速度に対して異なる要求をもつ複数のプロセスが混在する環境では，十分な結果は期待できない．「高速に動作するべきプロセスに合わせて低速に動作させて良いプロセスまで速度を上げて，必要以上の電

力消費を行なう」か,「低速に動作するべきプロセスに合わせて高速に動作するべきプロセスまで速度を落とし，要求を満たせない」かのどちらかしか選択出来ないからである．

OSレベルの手法として，Linux Kernel 2.6のDVFSモジュールに含まれるondemandおよびconservativeがある．この二つは本論文の実験でも取り上げた通りである．

7. おわりに

本論文では低電力消費でかつ応答性能の高いシステムを実現するためのスケジューラ，PCCSのプロトタイプをLinux上に実装し，その消費電力対応答性能の評価を行なった．結果，応答性能を必要とするプロセスと必要としないプロセスが混在する環境で，既存実装が応答性能を理想値に対して4%下げ消費電力を12%増大させるところ，本プロトタイプは1%の応答性能低下と4%の消費電力増大に抑えるという実験結果が得られた．ただし，この結果は論文17)での予想とは食い違う．

本論文で取り上げた実験環境は，現実にスケジューラが与えられる条件の内一つでしかない．今後，本研究の成果を適用可能な他の状況において，このプロトタイプ実装がどのような挙動を行なうのかを検証し，同時に改善していく必要がある．

また，本論文では論文17)で提案した手法のうちI/Oの性質に関する検討が含まれていない．論文17)で著者らは，各プロセスが持つファイルディスクリプタやソケットによるI/O処理を元に各プロセスの応答性を判定することで，より精度の高い予測が出来ることを主張した．今後はこの部分も含めた実装を行ない，PCCSの有効性について検討を行なう予定である．

謝辞 本研究の一部は，科学技術振興機構(JST)の戦略的創造研究推進事業(CREST)の支援を受けた．

参考文献

1) cpufreq daemon. `http://sourceforge.net/projects/cpufreqd`.
2) Ana Azevedo, Ilya Issenin, Radu Cornea, Rajesh Gupta, Nikil Dutt, Alex Veidenbaum, and Alex Nicolau. Profile-based dynamic voltage scheduling using program checkpoints. In *Automation and Test in Europe Conference (DATE), March 2002.*
3) AMD Corporation. Powernow! technology. `http://www.amd.com/us-en/assets/content_type/DownloadableAssets/Power_Now2.pdf`.
4) Intel Corporation. Speedstep technology.
5) Transmeta Corporation. Longrun dynamic power/thermal management. `http://www.transmeta.com/crusoe/longrun.html`.
6) Rong Ge, Xizhou Feng, and Kirk W. Cameron. Performance-constrained distributed dvs scheduling for scientific applications on power-aware clusters, 2005.
7) Chung hsing Hsu and Wuchun Feng. A power-aware run-time system for high-performance computing, 2005.
8) Chung-Hsing Hsu and Ulrich Kremer. The design and implementation and and evaluation of a compiler algorithm for CPU energy reduction, 2003.
9) Chung-Hsing Hsu, Ulrich Kremer, and Michael Hsiao. Compiler-directed dynamic voltage/frequency scheduling for energy reduction in microprocessors. *ISLPED*, 2001.
10) Nandini Kappiah, Vincent W. Freech, and DavidK. Lowenthal. Just in time dynamic voltage scaling: Exploiting inter-node slack to save energy in mpi programs, 2005.
11) H. Saputra, M. Kandemir, N. Vijaykrishan, M Irwin, J. Hu, C-H Hsu, and U. Kremer. Energy-conscious compilation based on voltage scaling, 2002.
12) Yifan Zhu and Frank Mueller. Feedback edf scheduling exploiting hardware-assisted asynchronous dynamic voltage scaling, 2005.
13) 浅井雅史, 池田佳路, 佐々木広, 近藤正章, 中村宏. 統計処理に基づくコンパイラ協調型dvfs手法. In *ARC166*, Jan. 2006.
14) 宮川大輔, 石川裕. プロセス単位電力制御機構の設計と実装. In *OS99*, May 2005.
15) 近藤正章, 中村宏. 主記憶アクセスの負荷情報を利用した動的周波数変更による低消費電力化. 情報処理学会論文誌：コンピューティングシステム 2004, pp. 1–11, May 2004.
16) 宮川大輔, 石川裕. プロセス単位電力制御機構の予備評価. In *OS100*, Aug. 2005.
17) 宮川大輔, 石川裕. 低消費電力のためのスケジューリングアルゴリズム. In *OS102*, May 2006.