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

2005－OS－99（29）
2005／5／27

プロセス単位電力制御機構の設計と実装

宮川大輔[†]　石川　裕[†]

最新の計算機では，消費電力やその結果発生する熱の問題が無視できなくなっている．プロセス毎にCPUの動作電圧を制御することで，計算機全体の消費電力を減らせる可能性がある．本研究では，プロセス毎の電圧制御が動作し，実用上有用であること，すなわち i) 電力消費を抑えられること ii) オーバーヘッドが無視できるほど小さいこと を Linux Kernel 2.6 を用いて示す．

The Design and Implementation of Per-Process Power Consumption Controlling System

Daisuke Miyakawa[†] and Yutaka Ishikawa[†]

In modern computers, we cannot ignore the problem of power consumption and the emitted heat. By controlling process-based voltage, the power consumption of whole systems can be reduced. In this work, we present using Linux Kernel 2.6.8 that the per-process voltage control works well and is effective: i.e. i) we can control the consumption and ii) the overhead for controlling is moderate enough to ignore.

1. 背　景

近年のCPUでは，CPUの動作周波数と消費電力はトレードオフの関係にある．単一プロセスのみを動かすような計算機であれば，そのプロセスに併せてCPUの動作周波数を設定すればよいが，現実には複数のプロセスが混在し，それらの優先度もさまざまである．緊急度の高いプロセスAと低いプロセスBが混在する場合，現在のOSでは，次のどちらかを選ぶしかない

- プロセスAに合わせて，動作周波数を上げる．プロセスAを高速に処理できる一方，プロセスBまで高速に動作するため，消費電力を増大させることになる．
- プロセスBに合わせて，動作周波数を低いままにする．消費電力は抑えられるが，プロセスAの処理速度まで遅くなり，緊急の処理が間に合わない可能性がある．

一方，もしOSがプロセス毎にCPUの動作周波数を変える機構を持てば，次のような選択もあり得る．

- 緊急度の高いプロセスの実行時には動作周波数を上げ，緊急度の低いプロセスの実行時には動作周波数を下げる．

これによって，より細粒度な電力制御を行なうことができるが，現実には動作周波数変更そのものにオーバーヘッドを伴う．本研究では，Linux Kernel 2.6 を用いてプロセス毎にCPUの動作周波数を変更する仕組みを実装し，そのオーバーヘッドが無視できることを示す．

2. 設　計

実験を行なうために，Kernelの cpufreq モジュールを改造する．cpufreqモジュールは，ハードウェアの DVS (Dynamic Voltage Scaling) 機構にアクセスするためのインターフェースとドライバ群を提供する．本研究では，モジュールに含まれる動作周波数を変更する関数群を利用し，それをスケジューリング時に呼び出すよう Linux Kernel を修正する．

修正された Kernel では，スケジューリングを行なう関数 schedule() の最後に，速度変更をするための関数を呼ぶ．その関数は sched_struct に追加されたポリシー変数を参照し，実際に CPU の動作電圧を変える cpufreq_driver_target() 関数を呼ぶ．cpufreq_driver_target() 関数はドライバ毎の詳細を隠蔽し，CPU の動作周波数を変更する．

3. ユーザインターフェース

実装を簡単にするため，動作電圧を変えるためのイ

† 東京大学
The University of Tokyo

```
len=sprintf(buf, "375000");
d=open("/sys/.../cpufreq/scaling_setspeed",
       O_WRONLY);
write(d, buf, len);
```

図 1　プロセスを最低動作周波数で動かす例

表 1　実験環境

検証用端末	NEC 社製 My30V/C-F
CPU	Pentium-4 3.0GHz
OS	Debian GNU/Linux 3.1
Linux Kernel	2.6.8
コンパイラ	gcc 3.3.5
ベンチマーク	NPB 3.0 の逐次実行版

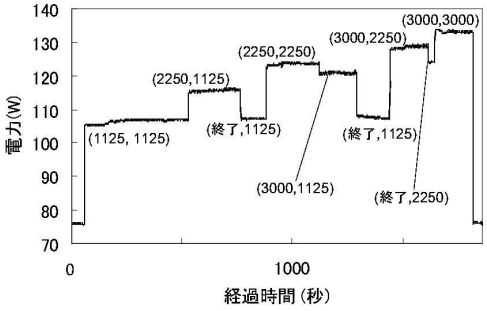

図 2　2 つのプロセスを同時に実行したときの消費電力の推移

ンターフェースは cpufreq のものをそのまま用いた．例えば図 1 のようにアクセスできる．本来の cpufreq モジュールにおいては，scaling_setspeed に書き込まれた値を元に CPU の動作周波数が変わるが，修正されたモジュールではプロセスの動作周波数が変わる．

4. 実験方法

実験環境は表 1 の通りである．この環境を用い，消費電力と NPB(NAS Parallel Benchmarks[1]) のスコア (実行速度を表す指標) の二点について，シングルプロセスのみ実行した結果と，二つのプロセスを同時に走らせたときの結果を比較した．

5. 結　　果

図 2 は，周波数を順に変えて 2 つプロセスを実行させた時の電力消費の推移である。括弧内の 2 つの値は各プロセスの周波数で，「終了」はそのプロセスが終了したことを示す．各時刻での消費電力は，2 つのプロセスが単一で動いたときの消費電力の総和平均に一致する．これは，各プロセスの動作周波数を規定することで，消費電力を細かくコントロールすることが可能であることを示している．

図 3 は NPB のスコアを比較した結果である．2 プロセス同時実行の場合には，片方のプロセス (A とする) を 375MHz に固定し，もう片方のプロセス (B とする) の動作周波数を変化させた．単一プロセスの場合と比較するため，グラフには B のスコアを 2 倍したものが示されている．この結果から，複数の動作周波数の異なるプロセスを同時に動かした場合の動作速度は単独で実行したときの速度とほぼ同じであり，動作周波数切替のオーバーヘッドは十分小さいことが分かる．

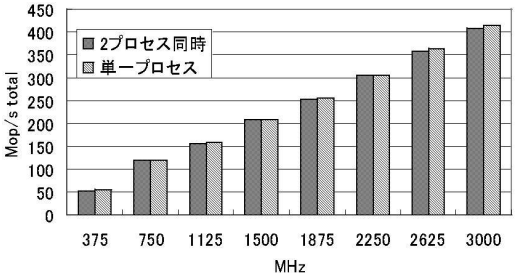

図 3　NPB の結果

6. 課　　題

本研究は，プロセス毎に動作周波数を変えたときの消費電力量，すなわち電力を時間積分したときの値については考慮していない．消費電力量が増えるか，増えるとすればどのくらいか，増えないケースはないかを引続き調査する必要がある．

また，本研究では，各プロセスの動作周波数は，実験中常に一定であった．一方，消費電力量を減らす研究として，特定のアルゴリズムを用いて動作電圧を動的に変化させる手法も提案されている [2]．それを Kernel レベルで実装したときにどうなるか検証することも，今後の課題である．

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) NAS Parallel Benchmarks. http://www.nas.nasa.gov/Software/NPB/.

2) 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, 2004.