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

2008－OS－108 (20)
2008／4／24

L4マイクロカーネルにおける省電力スケジューラの開発

林 和宏[†1] 金井 遵[†2]
丸山勝巳[†3] 並木美太郎[†4]

近年，サーバや組み込み機器などのあらゆる分野における省電力化技術の必要性が高まっている．我々は，これらの電力問題を解決する手段の一つとして，L4マイクロカーネル上でDVFS制御により省電力化を行うスケジューラ機構を提案する．本スケジューラは，L4スレッド単位で実行時情報に基づいた性能予測を行い，ユーザによってあらかじめ定められた性能制約を満たす範囲内で最適なCPU周波数を選択することにより省電力化を図る．また，性能予測モデルを実行環境に応じて自動的に最適化する学習機構，及び性能予測に誤差が生じる際にこれを補正するフィードバック機構を有する．本稿では，本スケジューラの設計と実装及び評価について述べ，ユーザレベルのOSサーバをアプリケーションプログラムなどと同等に電力制御することによるマイクロカーネルにおける本システムの適用性ならびに省電力化の可能性について考察する．評価では，I/Oベンチマーク実行時に最大32%のCPU消費エネルギー量の削減を実現した．

Implementation of a Power-saving Scheduler on L4 Microkernel

Kazuhiro Hayashi,[†1] Jun Kanai,[†2] Katsumi Maruyama[†3]
and Mitaro Namiki [†4]

Recently, the low power technology is needed in many fields such as server computers or embedded systems. We propose a power saving scheduler with DVFS(Dynamic Voltage and Frequency Scaling) on L4 microkernel. The scheduler reduces power consumption by predicting the performance of each L4 thread based on its run-time information and selecting the best voltage and frequency that satisfies the performance constraint defined by users. The scheduler has the "learning system" optimizing the performance estimation model automatically in each environment, and the "feedback system" compensating the performance estimation error. This paper describes the design, implementation and evaluation of the scheduler, and then, considers the applicability of the scheme to L4 microkernel and the possibility of power saving in microkernel by applying power-aware scheduling to user level OS servers similarly to applications. In evaluation, we achieved 32% CPU energy reduction of an I/O benchmark.

1. はじめに

近年，コンピュータ機器の消費電力の増加が問題視されている．ネットワーク上で高速な処理が必要とされるサーバコンピュータやクラスタシステムにおいては，性能向上に伴った消費電力と発熱の増大により，電力コスト及び冷却のための空調コストの増加が深刻な問題となっている．また，使用可能な電源の量に制限のあるノートPCや携帯電話といったポータブル機器においては，消費電力の増加は機器の長時間利用を実現するために不可避な問題である．

このような消費電力に関する問題に対して，計算機システムにおける様々な省電力化技術の開発・導入が行われている．電力削減手法としては，ハードウェアによる自律的な電力制御と，ハードウェアの提供する機能を用いたソフトウェアによる電力制御の二通りに大別される．ハードウェアレベルでの電力制御として，代表的なものには，CPU負荷に応じて動作周波数を自動で変化させるIntel社のSpeedStepなどの技術が

†1 東京農工大学工学府情報工学専攻
Department of Computer and Information Sciences, Graduate School of Engineering, Tokyo University of Agriculture and Technology
†2 東京農工大学工学府電子情報工学専攻
Department of Electronic and Information Engineering, Graduate School of Engineering, Tokyo University of Agriculture and Technology
†3 国立情報学研究所
National Institute of Informatics
†4 東京農工大学大学院共生科学技術研究院
Division of Systems and Information Technology, Institute of Symbiotic Science and Technology, Tokyo University of Agriculture and Technology

ある．また，回路構成要素の一部に対して電力供給を停止することで電力削減を図るパワーゲーティングを用いた手法[6]なども提案されている．ソフトウェアレベルでの電力制御としては，CPUの周波数・電圧の動的調整機能DVFS(Dynamic Voltage and Frequency Scaling)を用いたOSなどによる省電力化手法が代表的である．DVFS機能の具体例としては，Intel社の拡張版Intel SpeedStep[3]や，AMD社のCool'n'Quiet及びPowerNow!などが挙げられる．ソフトウェアによるDVFS制御の例としては，Linuxカーネル2.5以降に搭載されているCPUFreq機構，ARM社のIEMソフトウェアなどがある．CPU以外にも，ACPI規格に準拠した外部デバイスなどでは，未使用時にOSによって自動的にスリープ状態へと推移することで不要な消費電力を削減する技術などが実用化されている．同時に，オンチップ/オフチップメモリを用いて利用頻度の低いデータを低電力チップ上に退避することにより省電力化を図るページングアルゴリズム[7]や，ハードディスクのスリープやウェイクアップ処理をタスクによる使用状況に基づいて効果的に行う手法[8]など，OSによる各種オーバヘッドを考慮した電力制御に関する研究も為されている．

ハードウェアレベルでの省電力化手法では，回路レベルでの細粒度なシステム構築が行える他，クロック供給がないときにも消費されるリーク電力の削減が可能である．反面，ハードウェア単体による電力制御においては，ソフトウェアによって仮想化されるタスクなどの各種リソースに関する情報取得が困難であり，プログラムの実行時性能への影響などを考慮した最適な省電力化技術の構築が難しい．一方で，ソフトウェアレベルでの省電力化技術は，ハードウェアの提供する機能に依存するものの，各デバイスの使用状況やその使用者であるタスク，負荷などの情報を用いてより最適な電力制御を行うことができる．ハードウェアの利用状況やCPU負荷，キャッシュミス率などは実行されるプログラムごとに大きく異なるため，多様なアプリケーションが複数同時実行されるコンピュータシステムにおいて，タスクごとの動作特性を考慮した電力制御を行うことは効果的な省電力化を実現する上で重要である．

Linuxによる CPUFreqなど，一般的に普及しているDVFSを用いたソフトウェアの省電力化技術においては，CPU負荷や温度，動作プログラムの種類などの情報に応じて，あらかじめ定められた電力制御ポリシに従って周波数の調整を行っている．しかし，これらの情報だけではタスクごとの動作特性を把握できず，適切な周波数決定を行えない可能性がある．ユーザが独自の電力制御ポリシーを作成することもできるが，設定が複雑な上，プログラムごとの詳細な電力制御は困難であり，効率的ではない．

本稿では，コンピュータ機器の電力問題を解決する手段の一つとして，CPUのDVFS機能を用いたOSスケジューラによる省電力化技術を提案する．先行研究[1]では，DVFSを用いてプログラムごとに最適な動作周波数を決定することにより省電力化を行うスケジューラ機構をLinux上で実現している．この省電力スケジューラでは，Linuxプロセス単位で実行時情報に基づいた性能予測を行い，ユーザより定められた性能制約を満たす範囲内で周波数調整を行うため，プログラムごとの動作特性及び周波数低下による性能への影響を考慮したより細粒度での電力制御が可能である．本研究では，この省電力手法を用いて同等の機能を持つ省電力機構をマイクロカーネルを基盤とする計算機システムに対して導入する．

マイクロカーネルOSにおいては，Linuxなどのモノリシックカーネル OSと比べてカーネルの持つ機能は最小限に抑えられ，それ以外のOSサービスはユーザレベルで実装される．このような構造を持つことにより，移植性や拡張性，柔軟性に富んだOSの開発が可能となる．また，用途や目的に応じた必要な機能のみを備えるシンプルなOSの開発が容易であるため，組み込み・制御システムとしても適用性が高い．組み込み分野においても省電力技術は必須であり[7]，リアルタイム性などの性能制約との兼ね合いもまた必要となる．そこで本研究では，マイクロカーネル内のスケジューラに対して省電力機構を実装し，実機上での電力評価を通じてマイクロカーネルシステムにおける省電力化の可能性について考察する．

2. 目標と方針

マイクロカーネルOSでは，カーネルは割り込み管理やプロセス間通信(IPC)などの必要最低限の機能を持ち，これ以外の割り込みハンドラやファイルシステム，デバイスドライバなどといったOS機能はユーザレベルでのOSサーバにより実装される．すなわち，マイクロカーネル内のスケジューラに省電力機構を導入することによって，各種OSプログラムに対しても個別に電力制御が可能となる．モノリシックカーネルOSにおいては，デバイスドライバなどのOSサービスの多くが一つのカーネルプログラムとして存在し，割り込みや例外を通じて不定期に呼び出されるため，スケジューラによる個別の電力制御は困難である．そ

こで本研究では，マイクロカーネルに対して先行研究[1]の提案する省電力スケジューラを導入することによって，プログラムの実行時性能を考慮した効率的な省電力化を実現するとともに，各種OSサーバをアプリケーションプログラムなどと同等に電力制御するシステムを構築し，その評価を行うことを目標とする．

実装対象として，著名なマイクロカーネルであるL4[4]を利用する．一般的に，マイクロカーネルOSにおいては，OS機能がユーザタスクとして分離されているためにIPCやそれに伴うカーネル・ユーザモードの切り替えが頻繁に発生し，モノリシックカーネルと比較して性能低下を招きやすいという欠点がある．この点，L4は高速なIPCの実装によってOSサーバ間通信の高速化を実現しており，数あるマイクロカーネルの中でも特に性能面で優れているという特徴を持つ．

本研究では，このL4をベースとする二つのOS，L^{4}Linux及びLP49に対して省電力スケジューラを導入する．L^{4}Linuxはドイツ・ドレスデン工科大学のDROPSプロジェクト，LP49は日本・国立情報学研究所(NII)の分散OSプロジェクトH_2Oによって開発が行われているマイクロカーネルOSであり，それぞれFiasco，Pistachioと呼ばれる異なる実装のL4マイクロカーネルを基盤としている．L^{4}Linux，LP49のシステム構成を図1，2に示す．L^{4}Linuxでは，ユーザ空間に割り込み管理やページャといった資源管理サーバを配置し，その上で従来のLinuxカーネルとしての機能を持つL4タスクを動作させることにより，仮想的なLinuxシステムをL4のAPIを用いて実現している．LP49では，同じく基本的な資源管理サーバを動作させた上で，Coreプロセスと呼ばれるPlan9のカーネル機能を持つユーザサーバや，各種ファイルサーバなどがそれぞれ個別のL4タスクとして実装される．本研究では，FiascoとPistachioの両マイクロカーネルに対して省電力スケジューラを導入することにより，Linuxサーバや Coreプロセス，資源管理サーバといった各種OSタスク・スレッドに対して，アプリケーションプログラムなどと同等に電力制御を行う．

実装及び評価には，Intel社製品の中でも省電力効果の高いとされるPentium Mプロセッサを用いる．本CPUには，拡張版Intel SpeedStepテクノロジ(EIST)が搭載されており，9段階の周波数調整が可能である．また，パフォーマンスカウンタによりI/O実行やキャッシュミス，TLBミスなど種々のイベントの発生回数を同時に最大2つまで計測できる．実装する省電力スケジューラでは，まずパフォーマンスカウンタを用いてプログラムの実行時情報を取得し，これ

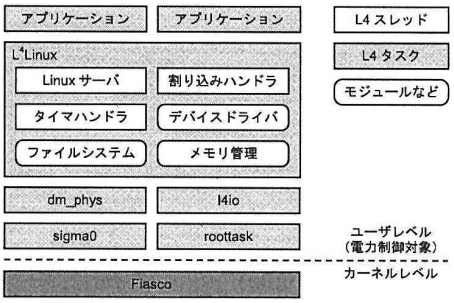

図 1 L^{4}Linux のシステム構成

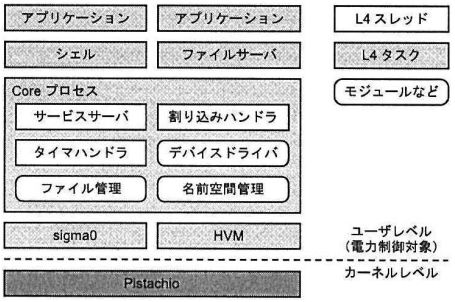

図 2 LP49 のシステム構成

に基づいた最適な動作周波数を決定した後，EISTによって実際に周波数の変更を行う．

3. システム設計

本研究における省電力化手法は，先行研究[1]の提案する方式に従うものとする．本章では，そのシステムの設計概要について述べる．

3.1 性能予測モデルを用いた省電力化手法

DVFS制御を用いてCPU周波数を下げることで省電力化を図る場合，周波数低下によるプログラムの実行時性能への影響を考慮する必要がある．本スケジューラでは，システムが提供すべき最低限の性能をあらかじめユーザによって指定し，本制約を満たす範囲内で周波数調整による省電力化を行う．ユーザにより決定される性能制約を以下では性能閾値と呼び，τと表記する．

一般的に，性能の指標としてIPS(Instructions Per Second)が用いられることが多く，本研究においてもIPSの値をプログラム実行時の実性能Pとして利用する．プログラムの実行コードがループなどによって常に一様であると仮定した場合，同一周波数f下でのIPS値は常に一定であると予想される．ここで，最高周波数$f_{\max}$で動作した時の性能を$P_{f_{\max}}$，実性能をPとしたとき，性能閾値τを最高周波数時に対して実

性能が満たすべき性能比

$$\tau = \frac{P}{P_{f_{\max}}} \quad (1)$$

と定義する．また，プログラムが同一コードを実行するという前提の下では，性能比は実行時間の比の逆数と等価となる．すなわち，常に最高周波数で動作したときのプログラムの実行時間を $t_{f_{\max}}$，実際に要した実行時間を t とすると，性能閾値 τ は

$$\tau = \frac{t_{f_{\max}}}{t} \quad (2)$$

と表わされる．例えば，性能閾値が50%と設定された場合，常に最高周波数で動作したときの半分の性能，すなわち2倍の実行時間でプログラムが終了することを条件に周波数調整を行えばよいことになる．

性能閾値の概念を導入することによって，プログラムの動作周波数を決定する際，各周波数で実行した場合に予測される性能が性能閾値を満たすかどうかの判定を行う必要がある．本スケジューラでは，周波数ごとの性能予測に下記の線形回帰モデルを利用する．

$$Y_f = A_f X + B_f \quad (3)$$

ここで，説明変数 X をプログラムの実行時パラメータ，目的変数 Y_f を周波数 f でプログラムを実行したときに予想される性能の最高周波数時に対する性能比とする．本スケジューラでは，まずプログラムの実行時情報 X を取得し，式(3)より各周波数 f における性能比 Y_f を予測する．次に，得られた Y_f のうち，ユーザにより定められた性能閾値 τ に対して

$$Y_f \geq \tau \quad (4)$$

を満たすような最低の f を動作周波数として選択することにより省電力化を行う．このとき，説明変数 X はプログラムの実行時性能との相関が強いパラメータであることが望ましい．先行研究[1)2)]では，L2キャッシュミスヒット数が性能に支配的であることを述べており，性能予測モデルにおける説明変数として1命令辺りのL2キャッシュミス率を用いている．本研究においても，説明変数 X にはL2キャッシュミス率を利用する．なお，実装及び評価に用いるPentium Mプロセッサでは，パフォーマンスカウンタによる実行命令数及びL2キャッシュミス回数の計測が可能であり，これを用いてプログラムごとのIPS及びL2キャッシュミス率の値を算出する．

性能予測に基づいた周波数決定は，プログラムの動作挙動の変化を検出するためにある程度の頻度で行う必要がある．本スケジューラでは，毎回のコンテキストスイッチ時に動作周波数を決定する．実装対象としたL4のスケジューラにおいては，プログラムの実行単位はすべてL4スレッドとして管理される．本研究の最も大きな特徴は，コンテキストスイッチごとにすべてのL4スレッドに対して電力制御を施すことにより，図1及び図2に示す各種OSサーバプログラムに対しても省電力化スケジューリングを行える点にある．

3.2 学習機構

前節で述べた性能予測モデルは，異なる特性を持つ複数のプログラムを実行し，得られた統計データをもとに回帰分析によって構築する．モデル構築の一つの手段として，事前に収集した統計データをもとにあらかじめ回帰係数を静的に決定しておく方法がある．しかし，性能予測モデルにおける係数値は異なるプラットホームやアーキテクチャの間で同一とは限らず，特定の環境下であらかじめ構築されたモデルを汎用的に用いることは好ましくない．学習機構では，テストプログラムを実行することで性能予測モデルの構築を自動的に行う．これにより，ユーザが独自の環境下で学習を行うことで最適な性能予測モデルを構築できる機能を提供する．

学習機構では，毎回のコンテキストスイッチ時にパフォーマンスカウンタから式(3)における説明変数 X 及び目的変数 Y のデータ x，y をスレッド単位で収集し，回帰分析により係数 A_f，B_f を決定する．性能予測モデルは周波数ごとに構築する必要があるため，コンテキストスイッチ時にはスレッドの動作周波数を一段階ずつ変更し，各周波数 f におけるデータ (x_f, y_f) を取得する．回帰係数 A_f，B_f は，スレッド毎に得られたすべてのデータから最小二乗法により算出する．また，学習の対象となるスレッドはユーザがあらかじめ指定し，必要最低限のプログラムに対してのみ学習を行うものとする．

3.3 フィードバック機構

本方式における性能予測モデル(3)は，L2キャッシュミス率がスレッドの実行時性能に大きく影響するものとして構築される．しかし，実際のプログラム性能はこれ以外のパラメータに大きく依存する場合も考えられ，キャッシュミス率のみが常に性能に対して支配的であるとは限らない．例えば，I/Oを多用するプログラムなどにおいては，I/O実行に要する時間がプログラムの実行時間に大きく影響するため，I/Oの頻度などの実行時情報はキャッシュミスなどと比べて性能に対してより相関の強いパラメータとなることが予想される．したがって，キャッシュミス率などの一つのパラメータのみを用いて多様な動作挙動を示す種々のプログラムの実行時性能を予測することは困難であり，結果として性能見積り誤差による消費電力の浪費

や過度な時間超過を招く恐れがある．

そこで，キャッシュミス率以外のパラメータが性能に大きく影響するようなプログラムが実行された場合に，必要に応じて性能予測における誤差を補正するフィードバック機構を実装する．フィードバック機構では，理想性能と実性能の誤差を自動的に検出し，これを補正することにより動作周波数の更なる最適化を行う．理想性能とは，プログラムを最高周波数で実行したときの性能 $P_{f_{\max}}$ に対する比率が性能閾値 τ と等しくなるような性能 P_{ideal} を指す．すなわち，

$$P_{\mathrm{ideal}} = \tau P_{f_{\max}} \tag{5}$$

となる．まず，理想性能 P_{ideal} の算出に必要な $P_{f_{\max}}$ の計測を行う．このため，フィードバック適用時には各スレッドは最低一回のタイムスライスを最高周波数で動作する必要がある．次に，コンテキストスイッチごとに式 (5) より P_{ideal} を計算し，実性能 P_{real} との誤差 P_{diff} を算出する．ここで，

$$P_{\mathrm{diff}} = P_{\mathrm{real}} - P_{\mathrm{ideal}} \tag{6}$$

とする．最終的に，毎回のコンテキストスイッチにおける P_{diff} の総和 $\sum P_{\mathrm{diff}}$ を 0 に近づけるように周波数を一段階ずつ変更する．今，$\sigma = \sum P_{\mathrm{diff}}$ とすると，σ が条件式

$$\sigma_{\min} \leq \sigma \leq \sigma_{\max} \tag{7}$$

を満たす場合，実性能と理想性能に大きな誤差はないものとして周波数調整を行わない．なお，$\sigma_{\min}$ 及び $\sigma_{\max}$ は，それぞれ σ の 0 に対する許容誤差の最小値，最大値とする．

$$\sigma < \sigma_{\min} \tag{8}$$

となるとき，実性能が理想性能を下回ることから性能制約が満たされていないと判断し，動作周波数を一段階上げる．

$$\sigma > \sigma_{\max} \tag{9}$$

となるとき，理想性能を上回るパフォーマンスにより過度な電力消費が行われているとみなし，更なる省電力化のために周波数を一段階下げる．以上の周波数調整によって，性能予測が外れる場合に対しても実性能の補正による適切な省電力化が可能となる．

3.4 全体構成

本システムの全体構成を図 3 に示す．L4 内の省電力スケジューラの動作モードとして，学習機構の動作する学習モードと，省電力化機構の動作する省電力モードの二つを実装する．本スケジューラは，二つの動作モードのうちいずれか一方で動作し，ユーザの要求に応じてそのモードを変更する．

学習モードでは，3.2 節で述べた手法に従って性能予測モデルの最適化を行う．ユーザは，必要な学習用

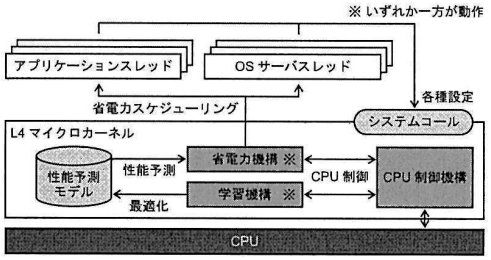

図 3 省電力スケジューラの全体構成

プログラムを一度ずつ実行するだけで，環境に応じた性能予測モデルを構築することができる．省電力モードでは，学習機構により構築された性能予測モデルを利用し，3.1 節で述べた手法に基づいた省電力化を行う．また，必要に応じて 3.3 節のフィードバック機能を適用し，性能予測のみでは適切な周波数決定が行えないスレッドプログラムに対しても，実性能と理想性能の誤差を補正することにより最適な動作周波数を選択する．周波数の変更やスレッド実行時のパフォーマンスカウンタの値の取得といった処理は，アーキテクチャごとに設計される CPU 制御機構を介して行う．

また，本スケジューラを利用するにあたり，動作モードの決定や学習プログラムの選択，性能閾値の設定やフィードバックの有無など，ユーザがシステムに対していくつかの設定を行う必要がある．このためのシステムコールインタフェースを L4 に新たに作成し，ユーザプログラムはこれを利用することで本スケジューラに対して各種設定を行う．

4. 実　　装

3 章で述べた設計に従い，L4 内のスケジューラに対して電力制御機構の実装を行った．今回実装対象とした Fiasco 及び Pistachio は，細かい実装方法や設計理念は異なるものの，L4 としての基本原理は同じである．いずれのカーネルもソースコードは主に C++で記述され，カーネル内の多くのコンポーネントがクラスとして抽象化されている．本研究では，DVFS 制御により省電力化を行う電力制御機構をひとつの新たなクラスとして L4 内に定義し，スケジューラなどの他の既存の処理から本クラスの提供する電力制御メソッドを呼び出す形で実装を行った．以下では，本クラスを DVFS クラスと呼ぶ．

学習機構及び省電力機構は，DVFS クラスの主なメソッドとして定義される．両機構による処理は，いずれもコンテキストスイッチ時に行うため，L4 スケジューラのコンテキストスイッチコードから DVFS ク

ラスのハンドラメソッドをコールし，モードに応じた処理を行う形で実装した．電力制御に必要なスレッドごとのデータなどは，DVFS クラスをスレッドの実態となるクラスのメンバとすることで保持した．また，CPU 制御機構の実装は，アーキテクチャ依存コードとして隔離することでシステムの移植性を高めた．

DVFS クラスの他に，ユーザにより各種設定を行うためのシステムコールインタフェースの作成を行った．L4 に本システム制御用のシステムコールエントリを新たに追加し，そこから DVFS クラスのシステムコールハンドラを実行する形で実装した．また，ユーザが本システムを効率的に操作するためのシステムコールライブラリを作成した．主要な API を表 1 に記す．

表 1 システムコールライブラリの関数例

`dvfs_set_mode`	動作モードを指定する．
`dvfs_feedback_on`	フィードバックを有効にする．
`dvfs_feedback_off`	フィードバックを無効にする．
`dvfs_study_on`	対象スレッドの学習を有効にする．
`dvfs_study_off`	対象スレッドの学習を無効にする．
`dvfs_set_thd`	対象スレッドの性能閾値を設定する．

5. 評　　価

省電力スケジューラを実装した L^{4}Linux，LP49 を用いて，実機上での電力測定による本システムの評価を行った．CPU は Intel Pentium M 760 を利用し，周波数は 0.8GHz から 2.0GHz までの全 9 段階を使用した．電力測定にはシナジェテック社の電流計測システム ST-30400 を利用し，非接触型センサによりシステム電力の計測を行った．ベンチマークプログラムには，行列演算 `matrix`，整数演算 `Dhrystone`，MiBench より高速フーリエ変換 `FFT` とハッシュ計算 `SHA`，及びディスク読み込みプログラム `disk read` を用いた．

まず，キャッシュミス率の異なる四つの CPU ベンチマーク `matrix`，`Dhrystone`，`FFT` 及び `SHA` を用いて学習機構による性能予測モデルの構築を行った．学習の結果得られた性能予測式 (3) における回帰係数 A_f 及び B_f の値を表 2 に示す．キャッシュミス率がほぼ 0 に近いプログラムの予測性能比となる B_f については両 OS 間で同等の結果が得られているが，キャッシュミス率に対する予測性能比の増加率となる A_f については周波数の低いモデルほど L^{4}Linux の方が大きくなる結果となった．これは，キャッシュミス発生時に要するオーバヘッドが L^{4}Linux の方が長いため，キャッシュミス率の高いプログラムを実行した際の CPU の周波数低下による性能への影響が LP49 に比べて小さいためであると考えられる．このように，性能予測モデルはプログラムの実行環境によって変化する可能性があり，学習によってシステム環境に適合する性能予測モデルを動的に構築することは，最適な電力制御を行う上で重要であるといえる．

表 2 性能予測式 (3) の学習結果

f[GHz]	L^{4}Linux		LP49	
	A_f	B_f	A_f	B_f
0.79	43.35	0.40	31.61	0.39
1.06	36.80	0.53	29.02	0.53
1.20	32.14	0.60	26.27	0.59
1.33	28.47	0.67	22.02	0.66
1.46	23.22	0.74	17.47	0.73
1.60	17.92	0.80	13.64	0.80
1.73	10.54	0.87	9.28	0.86
1.86	4.31	0.93	4.64	0.93
2.00	0.00	1.00	0.00	1.00

次に，表 2 の性能予測モデルを用いて省電力機構を適用し，ベンチマーク実行時の電力評価を行った．このとき，性能予測モデルを用いた効果を確認するため，比較対象として性能比が純粋に周波数比となるような単純な予測モデル

$$Y_f = \frac{f}{f_{\max}} \tag{10}$$

を用いた方式を実装した．評価では，式 (10) 及び式 (3) の二つの性能予測モデルとフィードバックの有無とを組み合わせた表 3 に示す四つの方式を用いて行った．`matrix`，`FFT`，`SHA` 及び `read disk` 実行時の CPU エネルギー消費量の評価結果を図 4 に示す．結果は，性能閾値を 90%，70%，50%に設定したときの，各方式における最高周波数時に対するパーセンテージで示してある．また，性能閾値に対する実性能の誤差を測定した結果を図 5 に示す．ここで，性能閾値 τ における実行時間 t_τ の性能誤差率 e_τ は

$$e_\tau = \frac{t_{100}}{t_\tau} - \tau \tag{11}$$

として算出している．

表 3 省電力化方式

方式名	性能予測モデル	フィードバック
FREQ	式 (10)	無効
FREQ+FB	式 (10)	有効
PMC	式 (3)	無効
PMC+FB	式 (3)	有効

5.1 CPU ベンチマークに対する評価

CPU ベンチマークに関しては，`matrix` や `FFT` などのキャッシュミス率が高いプログラムに対して，FREQ

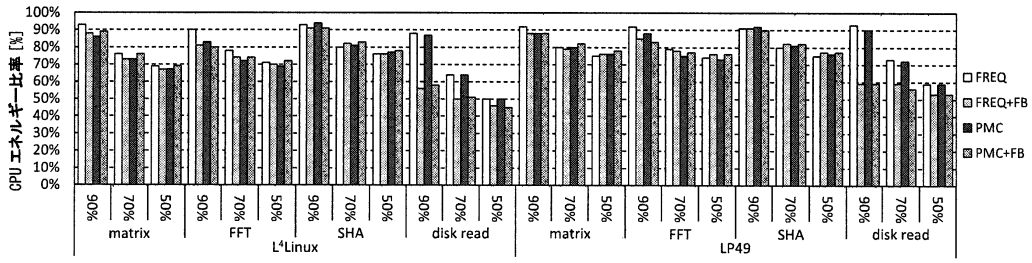

図 4 最高周波数時に対する CPU エネルギー消費量の比率

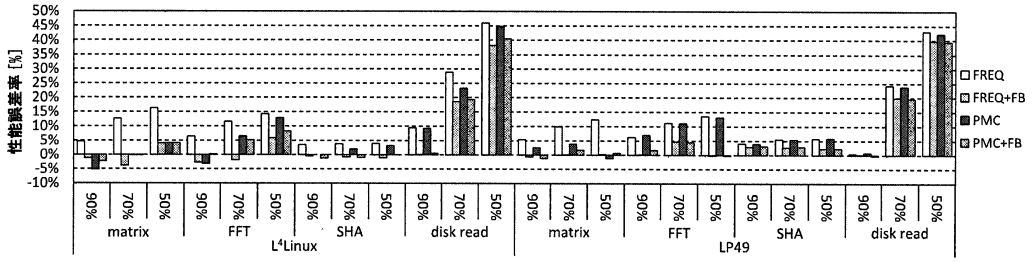

図 5 性能誤差率

方式に比べ PMC 方式の方がより高い省電力効果が得られている．また，最も負に大きい性能誤差率は −5% 程度であり，いずれの方式においても性能制約を大きくオーバーすることなく省電力化が実現できている．本結果から，キャッシュミス率などの性能に支配的な情報に基づく DVFS 制御によって，より効果的な CPU の省電力化を実現できる可能性が示された．一方で，`SHA` などのキャッシュミス率の低いプログラムに対しては，FREQ 方式と PMC 方式に大きな差はない結果となった．

また，先行研究[1] の Linux 上での電力評価と比較しても，CPU ベンチマークについてはほぼ同様の結果が得られている．CPU ネックなプログラムの実行については，OS 構造に依存する要素が極めて少ないため，マイクロカーネル OS においても Linux と同等の結果が得られるのは自明であるといえる．

5.2 I/O ベンチマークに対する評価

`disk read` では，L^{4}Linux，LP49 の双方において，フィードバック適用時に FREQ 及び PMC 方式に比べ高い省エネルギー効果が得られている．特に，L^{4}Linux において性能閾値を 90%に設定した場合では，FREQ 方式に比べ 32%の消費エネルギー削減を実現している．

`disk read` 実行時には，ディスクアクセスや割り込みにより多くの OS サーバが頻繁に動作することになる．L^{4}Linux 及び LP49 上でのディスク読み取りにおける OS 内部の処理は，細かい実装方法は異なるものの，その基本的な手順は同様である．L4 においては，すべての割り込みハンドラはカーネル外部の IRQ スレッドとして実装され，割り込みが発生すると対応するハンドラサーバは L4 からのメッセージを受けて処理を実行する．本実験においては，`disk read` の実行時間全体に対する IRQ スレッドの実行時間の割合が L^{4}Linux では 91.6%，LP49 では 87.6%と共に 9 割近くを占めていることがわかった．これらの IRQ スレッドは，ハードディスクに対するデータの入力命令によって実行時間の大半を消費しており，実性能比が CPU 周波数によらず常に 100%に近い状態となる．フィードバック機構では，実性能が理想性能よりも高い場合には周波数を下げ続けるため，性能閾値が 100%より低い今回のケースでは IRQ スレッドはほぼすべての実行時間を最低周波数で動作することになる．図 6 に，L^{4}Linux における IRQ スレッドの周波数ごとの実行時間内訳を示す．ここで，最高周波数を 8(2.0GHz)，最低周波数を 0(0.8GHz) とする．実際に，FREQ+FB 及び PMC+FB 方式においては，すべての性能閾値に対してほぼすべての時間を最低周波数で動作していることが確認できる．結果的に，フィードバックを適用することによって，性能低下を招くことなく最大限

の CPU 省電力化を実現できたことになる．これは，IRQ スレッドを個別に省電力スケジューリングすることで I/O プログラムの実行における最適な電力制御を実現した結果であり，本システムのマイクロカーネルへの有用性を示す結果であるといえる．

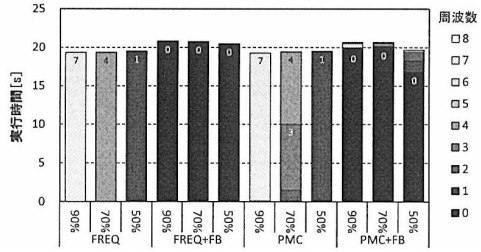

図 6 IRQ スレッドの周波数ごとの実行時間内訳 (L^4Linux)

先行研究[1] においても，ディスク読み込みプログラムに対してフィードバックを適用した場合に顕著な省電力効果が得られている．使用したハードディスクや評価方法の相違などにより本研究との明確な比較を行うことは困難であるが，いずれの OS 構造をもってしても，I/O ネックなプログラムに対するフィードバックの適用は効果的であることが理解できる．特にマイクロカーネル OS においては，割り込みハンドラなどの OS 処理そのものがスレッドとして分割されており，スケジューラによる細粒度な電力制御が可能であるため，より最適な周波数決定による省電力効果が期待できると考える．このため，本研究で用いたディスク読み込みプログラムの他にも，ネットワーク処理など，OS サービスを頻繁に使用するような種々のベンチマークプログラムに対して電力評価を行っていくことが重要だと考える．

6. おわりに

本稿では，L4 マイクロカーネルを対象とした省電力スケジューラの設計と実装，及び評価について述べ，マイクロカーネル OS における省電力化の可能性について考察を行った．評価では，特に I/O ベンチマーク実行時にフィードバック機構による顕著な省電力効果を実現し，マイクロカーネルに対する本スケジューラの有用性を示した．本システムは，特にマイクロカーネルの適用性が高いとされる組み込みなどの分野において省電力化技術としての利用が期待できる．

本研究で実装した省電力スケジューラでは，複数の異なるプログラムが並列実行されるような場合を想定しておらず，単一スレッドの動作のみを対象とした電力制御を行っていた．複数プログラムの同時実行に対しても，性能制約を満たす最適な省電力スケジューリングを実現することは今後の課題の一つである．さらに，組み込みシステムなどへの適用を想定し，リアルタイム性を考慮した省電力スケジューリングの実装を検討している．現システムにおいては，プログラムがユーザにより定められた性能制約を確実に満たすことが保障されていない．リアルタイムプログラムの時間制約と本システムにおける性能閾値の概念とを統合し，より実用的な省電力技術を考案することは今後の研究課題の一つである．

参考文献

1) 金井 遵, 並木 美太郎, et al. : 性能予測モデルの学習と実行時性能最適化機構を有する省電力化スケジューラ, 情報処理学会コンピュータシステムシンポジウム 2007 論文集, Vol.2007, No.14, pp.173-182 (2007).
2) 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 (省電力方式), 情報処理学会論文誌, Vol.47, No.SIG18(ACS16), pp.80-91 (2006).
3) Intel : Enhanced Intel SpeedStep Technology for the Intel Pentium M Processor, http://www.intel.com/design/intarch/papers/30117401.pdf
4) Alan Au, Gernot Heiser : L4 User Manual, http://l4hq.org/docs/manuals/l4uman.pdf (1999).
5) 日高宗一郎, 児玉和也, 丸山勝巳 : 実時間制御システム用のマルチサーバ型分散 OS の検討, 情報処理学会システムソフトウェアとオペレーティング・システム研究報告, 2001-OS-87-6, Vol.2001, No.65, pp.41-47 (2001).
6) 関 直臣, 天野 英晴, et al. : MIPS R3000 プロセッサにおける細粒度動的スリープ制御の実装と評価, 情報処理学会第 168 回「計算機アーキテクチャ」研究会, 2008-ARC-176, pp.71-76, (2008).
7) 小林 さとみ, 中西 恒夫, 福田 晃 : 組み込みシステムにおけるオンチップ/オフチップメモリアーキテクチャを対象とした省電力ページングアルゴリズム, 情報処理学会システムソフトウェアとオペレーティング・システム研究報告, 2002-OS-89-21, Vol.2002, No.13, pp.155-161, (2002).
8) Yung-Hsiang Lu, Luca Benini, Giovanni De Micheli : Operating-System Directed Power Reduction, Proceedings of the 2000 International Symposium on Low Power Electronics and Design (ISLPED'00), pp.37-42, (2000).