Vol.2013-OS-126 No.8
2013/7/31

KVMを用いた仮想化環境における省電力制御の研究

DOUANGCHAK SITHIXAY[1,a)]　佐藤 未来子[1]　並木 美太郎[1]

概要：本稿では，仮想化環境における省電力化を目的とし，消費エネルギー予測に基づいた省電力化を行なう仮想マシンモニター（VMM）の設計，実装と評価について述べる．従来手法では，OSやアプリケーションを修正する必要があるが，本研究ではVMMレイヤにおいて動的電圧・周波数制御（DVFS）時のVMごとの性能・消費電力予測モデルを用いた省電力制御を行なうことで，仮想マシン（VM）のゲストOSやアプリケーションに対して透過的に省電力を実現できるのが特徴である．本手法では，VM実行時のキャッシュミス率やプロセッサ全体のメモリアクセス頻度をハードウェアカウンタから取得し，回帰分析手法により求めた最適な性能・消費電力をもとにDVFS制御を実施する．実装には，Linuxカーネルを用いたVMMであるKVMを用いた．評価より，コア単位でDVFS制御可能なマルチコア環境において，性能条件の範囲内でメモリバウンドベンチマークが最大29. 49%，ネットワークベンチマークが38.15%の消費エネルギーの削減を確認した．

1. はじめに

近年，計算機の複雑化と性能向上に伴って，システムの消費電力の増加を招いている．計算機の効果的な利用を通じて環境保護をしつつあり，地球に優しい活動であるグリーンコンピューティングが活発に行われている．グリーンコンピューティングに関して，資源の有効活用や消費エネルギーの削減など様々な手法が提案された．その一つの手法として，仮想化技術が挙げられる．仮想化技術とは，ハードウェアモデルをベースにCPU，メモリ，I/Oなどの物理ハードウェアを仮想化する技術である．クラウドでは，仮想化技術を利用することにより，運用効率を向上し，必要なハードウェアを減らすことで，コストを削減する．しかし，クラウドの成長に伴い，計算機の台数増加によって，莫大な電力が消費されたため，仮想化環境でも省電力化の要求が高まっている．

計算機システムの中で，システム全体の消費電力のうち，プロセッサは大きな割合を占める場合が多いことから，省電力化の研究が盛んに行われている．省電力化手法は，大別して，ハードウェアレベルの手法とソフトウェアレベルの手法に分類することができる．ハードウェアレベルの手法は，プロセス技術の改善やハードウェアが自動的に電源制御を行なう．例として，Intel社のEIST(Enhanced Intel Speedstep Technology[1]，AMD社のCoolCore[2]，様々な手法が存在する．一方，ソフトウェアレベルの手法は，ハードウェアが提供する省電力化機構である動的な電源電圧・周波数制御DVFS(Dynamic Voltage and Frequency Scaling)がある．これは一般に，プロセッサ等の動作周波数を下げると電圧も低下させることができ，消費電力を低下させることができることを利用する．当然ながら動作周波数を下げると処理性能も落ちるため，必要な時にのみ周波数制御を行なうことで省電力化が可能である．DVFSを用いたソフトウェアによる省電力化の例としては，Linuxシステムに搭載されたcpufreq[3]モジュールと各種governorシステムを併せたものとWindowsのCool'n'Quietがある．

従来の計算機単体の省電力化の研究はDVFS手法を利用して省電力化を行なうことが中心であり，OSやアプリケーションを修正して，直接にハードウェアをコントロールした．仮想化環境の場合は，複数のVMが存在し，VMやゲストOSは直接にハードウェアをコントロールすることができない．ゲストOSとハードウェアの間に，VMMレイヤがあり，VMMレイヤにおいてゲスト環境向けに仮想デバイスを提供する．このように，従来の省電力手法をそのまま仮想化環境に適用することが難しい．省電力化は仮想化環境という新しい分野に挑戦することになる．仮想化は，準仮想化と完全仮想化に大別することができる．準仮想化は，オーバーヘッドを最小にするために，必要最低限な仮想化的なハードウェアを提供する．高いパフォーマンスを提供できるが，VMM用にゲストOSを改変する必要がある．対して，完全仮想化は，ハードウェアをソフトウェア的に完全にエミュレートし，仮想化的なハードウェ

[1] 東京農工大学
Tokyo University of Agriculture and Technology
[a)] joe@namikilab.tuat.ac.jp

アを提供する．準仮想化と比較したら，オーバーヘッドが大きいが，ゲスト OS を改変する必要がない．準仮想化を提供する代表的なものは Xen[4] があり，完全仮想化を提供する代表的なものは KVM[5] がある．本研究は，完全仮想化を提供する KVM を用いて省電力化を目指す．KVM は，Linux にマージされて Linux カーネル自体を VMM とする仕組みである．KVM はホスト OS の Linux のスケジューラ，メモリ管理などをそのまま利用できる．さらに，Linux カーネルの進化とともに，新しい機能が提供されている．今後，KVM を含む Linux をベースとしたクラウドに発展していくことが予想され，KVM の適用はさらに増えていくと考えている．

性能を予測し，性能をできるだけ落とさずに省電力化を目指す研究としては，多くの研究 [8][9] がある．また，実際の Linux システムに適用する研究 [10] もある．従来の省電力化研究では，可能な限り周波数を落すことを前提としている場合が多い．しかし，DVFS により周波数を落すと，VM の処理性能も落ちるため，必ずしも周波数を最低にすれば消費エネルギーを最も削減できるとは限らない．性能のみでなく，DVFS 制御を行なった時に，消費電力についても予測を行なう必要がある．また，予測を行なう方法として，定性的に対象システムを性能・消費電力をモデル化する方法と，定量的にモデル化する方法が存在する．しかし，定性的にシステムを分析する方法では，モデル式を作ることが困難で，多くのプラットホームへの適用が難しい．そこで，本研究では，性能・消費電力予測の方法として，予測のモデル化を容易・機械的に行なうことのできる，定量的なモデル化手法に注目した．また，VMM レイヤで統一的に省電力制御を適応するため，VM のゲスト OS ごとに別の省電力制御を用意する必要がない．

完全仮想化を用いて仮想化環境における省電力化を行なう場合に，直面しなければならないことは，ゲスト OS に改変することができないことと，どのように VM の特性と消費電力の関係性を把握するかという問題がある．これに対して，本研究では VM とハードウェアの中間レイヤである VMM レイヤに省電力制御を適応する．実際に，仮想化環境において DVFS 制御を行なう Kamga ら [13] の研究が存在する．Xen の VMM レイヤと管理ドメインに実装，VM の CPU 負荷状況を判断する．しかし，VM 上で動作するアプリケーションにより周波数を落してもほとんど性能が落ちない場合がある．このような場合は，DVFS 制御が行なわれていない欠点がある．VM 上で動作するアプリケーションにより，DVFS 制御時に性能や消費電力の増加が異なる．DVFS 制御時に，VM とプロセッサの消費電力はどのような関係を持つかを予測することができれば，省電力化の効果を最大に引出すことが可能となる．それに対して，本研究は VMM レイヤにおいて DVFS 制御時にパフォーマンスカウンタを用いて VM ごとの性能・消費電力予測をモデル化した．予測モデルを用いた省電力制御を行なうことで，VM のゲスト OS やアプリケーションに対して透過的に省電力化を実現する．本研究で用いる KVM により生成された VM はホスト OS のスケジューラにより，管理される．KVM はホスト OS が提供した機能を利用することができるため，本研究は，VM に対して独自なポリシーに合わせて省電力制御を設ける必要がない．

本手法では，まず，ハードウェアから得られる各種情報を元に，定量的にモデル化を行なう．性能モデルと消費電力モデルに分けて，性能モデルを元に VM 単位で DVFS 制御を行ない，与えられた性能閾値を上回る性能を保ちつつ，消費電力モデルから消費エネルギーが最小となるようなシステムを設計，実装した．

2. 仮想化環境に省電力制御のモデル

本研究では，仮想化環境に省電力制御を適用することにより，各 VM の性能閾値を上回る性能を保ちつつ，消費エネルギーが最小となるように，VM 単位およびコア単位で DVFS を行なう．完全仮想化環境を提供する KVM の VMM レイヤにおいて統一的に電力制御を行なうことで，VM のゲスト OS やアプリケーションに変更を加える必要がない．

また，パフォーマンスカウンタにより得られた性能・消費電力に関する値に対して定量的に回帰分析を行い，それらの値を予測するモデルを提示する．この予測モデルにより，未知のゲスト OS やアプリケーションに対しても，随時省電力制御ができる．また，予測モデルの最適化により，様々な計算機に対して，必要な計算機性能や，その計算機の消費電力特性に応じて，最適な周波数制御を行なうことができる高度な省電力化機構を仮想化環境へ適用できる．

3. 設計

本章では，仮想化環境における省電力制御に用いる予測モデルの最適化とその適用について述べる．先行研究として，金井ら [11] は Linux を対象とし，林ら [12] は L4 マイクロカーネルを対象とし，定量的に予測モデル化を行ない，計算機システム上で動作するプロセスを対象にして，省電力化を行なった．本研究では，プロセス単位ではなく VM に適用し，VM 単位に異なる性能条件を設定することができて，より細かい粒度で制御を行なうことで，リアルタイム性を有する仮想化環境における省電力制御を提案する．

3.1 予測モデルの最適化

仮想化環境における省電力制御を行なうために，個々の VM の消費エネルギーを予測する必要がある．消費エネルギー予測に使われる VM の性能・消費電力予測は定量的に回帰分析し，モデル化を行なう．また，定量的な分析手法を採用することで，モデルを機械的に最適化できるように

なる．以下，予測モデルの最適化を行なうために必要となる消費エネルギー予測，統計情報による VM の性能・消費電力の予測について述べる．

3.1.1 消費エネルギー予測

マルチコア上での仮想化環境における VM 単位の消費エネルギー予測は，VM の処理単位毎に，その処理に要する時間と消費電力の積から求める．本研究は，コアを n 個持つホモジニアスマルチコアプロセッサ上で仮想化環境があると想定する．任意 VM_i がコア c での周波数を f_c，その時の消費電力を W'_{vm_i,c,f_c}，コア c で VM_i の処理に要する時間 T_{vm_i,c,f_c} とすると，VM_i の処理に要するコア c の消費エネルギー E_{vm_i,c,f_c} は，

$$E_{vm_i,c,f_c} = W'_{vm_i,c,f_c} \times T_{vm_i,c,f_c} \tag{1}$$

と表すことができる．また，最高周波数 fmax での消費エネルギー比 R'_{vm_i,c,f_c} は，

$$\begin{aligned} R'_{vm_i,c,f_c} &= (W'_{vm_i,c,f_c}/W'_{vm_i,c,fmax}) \\ &\quad \times (T_{vm_i,c,f_c}/T_{vm_i,c,fmax}) \end{aligned} \tag{2}$$

となる．ただし，実際に，各コアの消費電力を求めることが困難である．本研究は，全体のプロセッサの消費電力 $\sum_{c=1}^{n} W'_{vm_i,c,f_c}$ をコア n で割った値を各コアの消費電力と見なす．VM_i の処理性能あたりのプロセッサ全体の消費電力 $R_{vm_i,c,(f_1,...f_n)}$ ($(f_1,...f_n)$ は各コアの周波数の組合せとする)．また，$T_{vm_i,c,f_c}/T_{vm_i,c,fmax}$ の逆数 P_{vm_i,c,f_c}=$T_{vm_i,c,fmax}/T_{vm_i,c,f_c}$ は，周波数 f_c での最高周波数に対する VM_i の性能比となる．VM_i の処理に消費電力比は，次のとおりとなる．

$$\begin{aligned} R_{vm_i,c,(f_1,...,f_n)} &= (\sum_{c=1}^{n} W'_{vm_i,c,f_c} / \sum_{c=1}^{n} W'_{vm_i,c,fmax}) \\ &\quad /P_{vm_i,c,f_c} \\ &= W_{vm_i,(f_1,...,f_n)}/P_{vm_i,c,f_c} \end{aligned} \tag{3}$$

式 (3) より，消費エネルギー比は，VM の性能比および消費電力比から求めることができる．消費エネルギーを最小化するために，この値が最小化となる周波数を選択し，VM の VCPU が動作するコアに設定する．

3.1.2 統計情報による VM の性能の予測

VM の性能予測は，性能に影響を与える全コア共有 L3 キャッシュミスを用いて，回帰分析による性能予測のモデル化を行なう．あらかじめ VM 上で多数のベンチマークプログラムを実行し，統計的な学習を行ない，あるパフォーマンスカウンタの値と VM の性能の間の相関関係を回帰分析によりモデル化を行なう．パフォーマンスカウンタの中には，キャッシュミス回数など性能に大きく影響し，高精度での性能予測が可能となっている．本研究は，定量的な手法を用いることで，様々なプラットフォームで容易に性能予測式を立てることができる．

ターゲット周波数毎に VM 単位で性能予測を行ない，ユーザにより与えられた個々の VM の性能閾値を下回らない最低周波数で動作させることで，省電力化を行なう方法を提案している．性能予測を行なうためには，様々なベンチマークプログラムを様々な周波数で動作させ，学習を行ない，性能予測式をモデル化する必要がある．実行時間の最高周波数時との相対比（つまり性能比）を目的変数，キャッシュミス回数を命令数で正規化したものを説明変数として学習を行い，回帰分析を行なう．回帰分析を行なった VM の性能予測式は，コア c での周波数 f_c 時の任意 VM_i の性能を $Y_{(vm_i,c,f_c)}$ とおく．性能 $Y_{(vm_i,c,f_c)}$ を目的変数とする．この性能 $Y_{(vm_i,c,f_c)}$ は，VM_i 上で，ある処理を行なった場合，コア c のみで最高周波数の場合に要する時間と周波数 f_c の場合に要する時間の比と定義する．つまり，VM_i の処理の最高性能に対する相対的な性能比となる．命令当たりの L3 キャッシュミス回数を説明変数 $X_{vm_i,c,q}$，a_{q,f_c} を回帰係数とすると，VM_i の性能比は，

$$Y_{vm_i,c,f_c} = a_{0,f_c} + \sum_{q=1}^{p} a_{q,f_c} X_{vm_i,c,q} \tag{4}$$

と表すことができる．

3.1.3 統計情報による VM の消費電力の予測

VM の消費電力予測は，消費電力に影響を与えるメモリアクセス，各コア L2 キャッシュミス，全コア共有 L3 キャッシュミスを用いて，回帰分析による消費電力予測のモデル化を行なう．

VM の消費電力は，VM の特性により，つまりゲスト OS 上で動作するアプリケーションにより異なる．今回の実装対象に選択した Phenom プロセッサをはじめ，多くのシステムではプロセッサとメモリのクロックが非同期で動作することが多い．このような環境では，DVFS によるプロセッサの周波数変更での前後で，メインメモリへアクセス頻度の異なる VM では消費電力比が異なると予想される．例として，メモリへアクセス頻度の高い VM は，元々プロセッサ内の ALU/FPU など各種機能，キャッシュアクセスに要する動作時間が少なく，プロセッサの DVFS 制御を行ない，動作周波数を下げても，その分の動作時間の消費電力しか減らない．一方，メモリへアクセス頻度の低い VM は，消費電力が大幅に削減できると予想される．そこで，回帰分析に用いる説明変数 $Z_{vm_i,q}$ は，全コア合計の命令当たりのキャッシュヒットした回数を含む全メモリアクセス数，L2 キャッシュミス回数，L3 キャッシュミス回数の 3 要素にした．

VM の性能予測と同様に，あらかじめ VM 上で多数のベンチマークプログラムを実行し，消費電力とプロセッサから得られた統計情報の間の関係を回帰分析によりモデル化を行なう．さらに本方式は，定量的に消費電力を分析するため，キャッシュの構成などを意識することなく，予測

ができる．回帰分析に用いる目的変数は，VM_i が全コア高周波数動作時に対する，任意の周波数組合せの場合のプロセッサ全体の消費電力比とする．コア単位で DVFS 可能なコア数 n のプロセッサを仮定し，説明変数を $Z_{vm_i,q}$，$b_{q,(f_1,...,f_n)}$ を回帰係数とする．各コアの周波数 $(f_1,...,f_n)$ のとき，VM_i の消費電力比 $W_{vm_i,(f_1,...,f_n)}$ は，

$$W_{vm_i,(f_1,...,f_n)} = b_{(f_1,...,f_n)} + \sum_{q=1}^{r} b_{q,(f_1,...,f_n)} Z_{vm_i,q} \quad (5)$$

と表すことができる．

3.2 仮想化環境に省電力制御の適用

本研究では，仮想化環境と物理ハードウェアの中間にある VMM レイヤに省電力制御を設ける．前節で述べた予測モデルを行なうために，仮想化環境に関する情報と物理ハードウェアに関する情報が必要となる．仮想化環境では，各 VM に関するコンテキストスイッチ情報，処理性能などが必要であり，物理ハードウェアでは，プロセッサから得られるパフォーマンスカウンタの値と DVFS 情報が必要である．VM 上で，あらかじめ様々な特性を持つベンチマークを動作させ，学習機構で性能・消費電力に関する学習をさせて予測モデル化を行なう．その予測モデルから，周波数決定機構で消費エネルギーをが最小となるような周波数を決定する．

省電力制御の全体構成は図 1 に示す．実装では，完全仮想化環境を提供する KVM の VMM レイヤに本手法を適応する．本システムは，リストを用いて，全ての VM に関する情報を管理する．KVM により，VM が起動される場合，リストに性能・消費エネルギーに関する情報を格納する．VM がシャットダウンされた場合，リストからその VM の情報を削除する．

仮想化環境における省電力制御には，VM 性能・消費電力予測式をモデル化する学習モードと消費エネルギーが最小化となるような省電力モードからなる．ホスト OS のユーザインタフェイスから，キャラクデバイスを介してモードの切替えが可能である．以下，この二つのモードについて述べる．

学習モードでは，省電力モードで用いる予測式の学習を行なう．VM 単位でパフォーマンスカウンタと性能・消費電力の統計を取り，性能・消費電力のための回帰分析の回帰係数を算出し，予測式モデル化する．学習モードは，

- VM 管理機構
- パフォーマンスカウンタ管理機構
- 学習機構
- DVFS 制御機構
- 予測モデル生成機構

からなる．VM 管理機構，パフォーマンスカウンタ管理機構，DVFS 制御機構は省電力モードと共通である．VM 管理機構は，稼働中の VM の情報を管理する．VM はホスト OS から見れば，単一のユーザプロセスである．VM 管理機構は，このプロセス ID を利用し管理を行なう．パフォーマンスカウンタ管理機構は，VM 単位で回帰分析に用いるパフォーマンスカウンタの集計を行ない，予測学習データに格納する．一般的に，VM とハードウェアの間に VMM が存在し，ある VM の動作によってパフォーマンスカウンタがどう変動するかを把握することが難しい．しかし，パフォーマンスカウンタ管理機構は，VM 管理機構から得られた VM の情報と VMM から VM のディスパッチタイミングで正確にコンテキストスイッチを計算し，パフォーマンスカウンタを求める．学習機構は，コンテキストスイッチ毎に DVFS 制御機構を連携して，動作周波数を変更し，パフォーマンスカウンタの値と性能・消費電力の関係を導出する．予測モデル生成機構では，電力測定器と連携し，測定された計算機のプロセッサ全体の消費エネルギーと学習機構から得たパフォーマンスカウンタの値を利用して回帰分析を行ない，最適な回帰係数を求め，性能・消費電力の予測式を立てる．

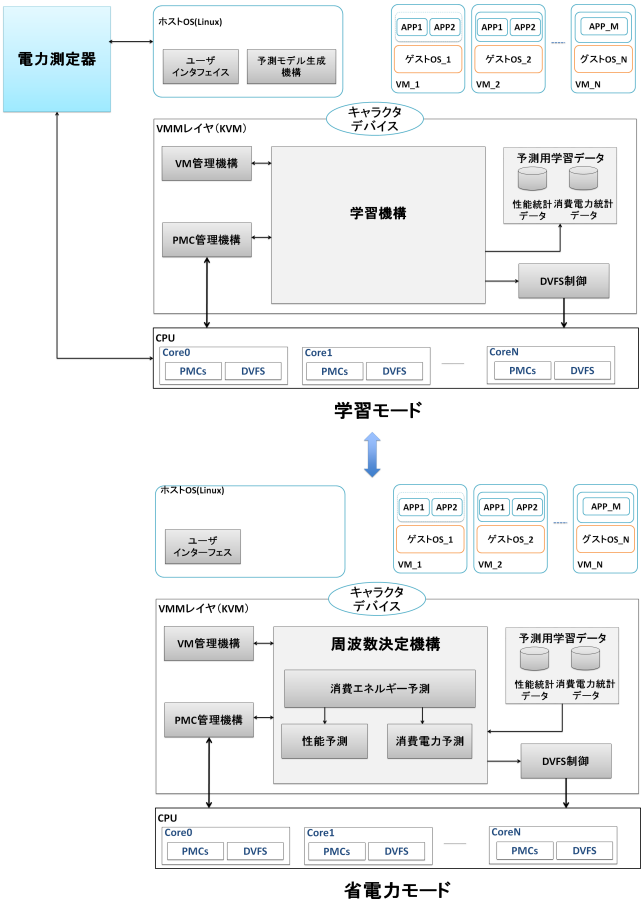

図 1 省電力制御の全体構成

省電力モードでは，学習モードでモデル化した性能・消費電力予測式を用いて，ユーザが与えた性能制約の中で実際に消費エネルギーが最小となるように省電力化を行なう．省電力モードは，

- VM 管理機構
- パフォーマンスカウンタ管理機構
- 周波数決定機構
- DVFS 制御機構

からなる．周波数決定機構は，コンテキストスイッチ毎にパフォーマンスカウンタ管理機構から得られた統計情報と性能・消費電力予測式から VM の性能・消費電力予測を行なう．ユーザにより設定された性能制約の範囲で，消費エネルギー予測を行ない，消費エネルギーが最小なるような周波数をを算出する．そして，VM の VCPU が動作するコアの周波数を変更する．

3.3 VM の性能閾値

VM の性能閾値を設定することにより，各 VM の性能閾値を上回る性能を保ちつつ，消費エネルギーを最小化することができる．

ホスト OS のユーザインタフェイスから，キャラクデバイスを介して VM の性能閾値を設定することができる．設定された閾値は周波数決定機構により，以下のアルゴリズムを用いて，最適な周波数を決定する．

VM の性能閾値とは，最高周波数で動作した場合に，その VM 上である処理を行なうのに要する時間に対する時間の比として定義する．例として，VM_i 上である処理が，最高周波数かつ，他のコアで何も実行されていない際に 1 分で終了する処理について，他のコアの状況にかかわらず，2 分で終了すればいい場合には，VM_i の性能閾値として 50%を指定する．この時，与えられた VM_i の性能閾値 THD に対して，コア c の周波数を式 (4) を利用して，

$$THD \leq Y_{vm_i,c,f_c} \tag{6}$$

を満たす周波数の中から選択する．

ただし，仮想化環境で複数の VM が動作することが多い．さらに，VM の VCPU が複数のコアで処理が実行されることを想定する．このとき，

$$THD \leq Y_{vm_i,c,f_c}/Y_{vm_i,c,fmax} \tag{7}$$

を満たす周波数の中から選択し，該当コアに周波数を変更する．

3.4 周波数決定機構

本システムは周波数決定機構を通して，消費エネルギー予測に基づき，VM のタイムスライス単位・コア単位で周波数選択を行なう．以下，この設計について述べる．

周波数決定機構では，コンテキストスイッチ毎に 3.1 節で述べた方法により，VM の性能・消費電力予測を行ない，その値から予想される消費エネルギーが最小となる周波数を決定する．性能・消費電力予測にはパフォーマンスカウンタの統計情報を用いるが，VM ごとに統計データの傾向

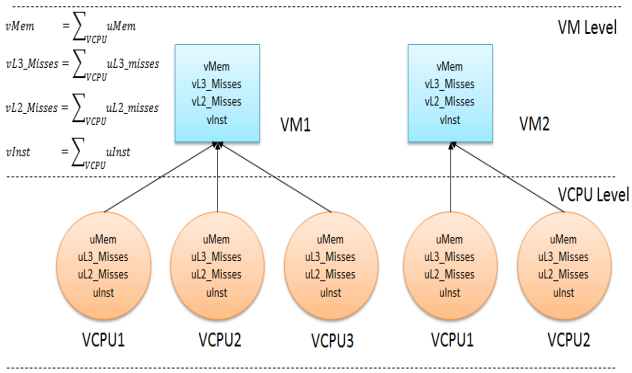

図 2 VCPU スケジューリング時のパフォーマンスカウンタ取得

は異なるり，性能や消費電力の特性や最適な周波数は異なる．そのため，本機構では VM 単位パフォーマンスカウンタ管理機構を利用して，VM ごとに VM のタイムスライス単位で統計情報の保存する．

本研究では，仮想化環境において複数の VM が存在し，さらに，各 VM が複数の VCPU を持ち，各 VCPU は全ての物理 CPU（PCPU）に対応付けられることを想定する．そのため，周波数決定機構は VM コンテキストスイッチ毎ではなく，VCPU コンテキストスイッチ毎にパフォーマンスカウンタの取得を行なう．そして，VM のタイムスライス単位で周波数を決定する．VCPU コンテキストスイッチ時，つまり，KVM が VCPU を PCPU 上にスケジューリングされた時に，パフォーマンスカウンタの取得を開始する．VCPU が PCPU から解放された時，パフォーマンスカウンタの取得を終了し，統計情報として，VM 管理機構にあるリストに保存する．取得する情報は，各 VCPU のメモリアクセス数 (uMem)，各コア L2 キャッシュミス回数 (uL2_Misses)，全コア共有 L3 キャッシュミス回数 (uL3_Misses) と実行命令数 (uInst) である．この各 VCPU の情報の合計は VM のタイムスライス単位の情報となる (図 2 に示す)．本機構は式 (8) により，式 (4) または式 (5) の予測式に代入する説明変数を求める．予測式に代入する説明変数の値は，一つ前の VM コンテキストスイッチの値となる．

$$\begin{aligned} vMem\_rate &= vMem/vInst \\ vL2\_Misses\_rate &= vL2\_Misses/vInst \\ vL3\_Misses\_rate &= vL3\_Misses/vInst \end{aligned} \tag{8}$$

実際のコア c で VM_i の周波数 f_c 選択は図 3 に示すアルゴリズムにより行う（4 コアの場合）．ここで，VM 単位で DVFS 制御を行なうため，VCPU コンテキストスイッチが発生したコア c では自コアの周波数切替のみを行なうものとする．本機構は，次に動作するプロセスについて，設定可能な周波数について，式 (4) から VM_i 性能 Y_{vm_i,c,f_c} の

```
1 // VM i番目の性能予測と性能閾値を満たせる最低周波数の選択
2 for j ← MAX_FREQ..0
3   vm_perf[i][j] ← estimate_performance(j, pmc_perf[i])
4   vm_perf[i][j] ← vm_perf[i][j] / vm_perf[i][MAX_FREQ]
5
6   if Thresold[i] > vm_perf[i][j] then
7     break
8   endif
9 fmin[pcpu] ← j //各コアの性能閾値充足可能な最低周波数を保存
10
11 //自コア以外の周波数の降順ソート
12 for j ← 0..MAX_PCPU - 1
13   if j != pcpu then
14     fmin_sort[] ← fmin[j]
15   endif
16 sort(fmin_sort)
17
18 // VM i番目の消費電力予測と消費エネルギーが最小となる周波数の選択
19 energy_min ← FLOAT_MAX
20 for f[0] ← MAX_FREQ..fmin[pcpu]
21   for f[1] ← MAX_FREQ..fmin_sort[0]
22     for f[2] ← MAX_FREQ..fmin_sort[1]
23       for f[3] ← MAX_FREQ..fmin_sort[2]
24         vm_power[i] ← estimate_power(f,pmc_perf[i])
25         vm_energy[i] ← vm_power[i] / (vm_perf[i][f[0]] / vm_perf[i][MAX_FREQ]) // 式(3)
26
27         if vm_energy[i] < energy_min then
28           energy_min < vm_energy[i]
29             best_freq ← f[0]
30         endif
31
32 //コア番号 pcpuの周波数をbest_freqに変更
33 dvfs(pcpu, best_freq)
```

図 3 コア数 4 の場合の周波数選択アルゴリズム

予測を行い，式 (6) または式 (7) により各コアの性能閾値充足可能な最低周波数を保存する（2-9 行目）．他コアも含め性能閾値充足可能な周波数の組合せを列挙し（20-23 行目），式 (5) からその際の消費電力 $W_{vm_i,(f_1,\ldots,f_n)}$ の予測を行なう（24 行目）．これらの予測結果から，式 (3) より消費エネルギーを予測し（25 行目），消費エネルギー最小と予測される周波数の組合せのうち，自コアの周波数を動作周波数 best_freq として設計する（33 行目）．

4. 実装

以上で述べた設計に基づき，カーネルモジュールである kvm-kmod-3.4 とホスト OS のユーザ空間に機能追加を行なった．KVM は，Linux カーネル自体を VMM とする仕組みであり，Intel VT-x[6] や AMD-V[7] といった CPU の仮想化支援機能を活用し，完全仮想化による仮想化環境を提供する．現在，KVM は他の仮想化技術と比較すると，歴史的に若い技術であるが，KVM は Linux カーネルに統合されることから，注目を集める．ホスト側の Linux のスケジューラ，メモリ管理などをそのまま利用するため，KVM 自体のスケールは小さい．また，KVM は独自のポリシーを持たないため，省電力化の手法だけに集中することができる．本研究では，ターゲットとなるアーキテクチャは x86 とし，同じ x86 の中でも特に電源電圧・周波数制御は CPU やチップセット依存となるため，AMD の Phenom9500 をターゲットとした．

既存のカーネルモジュールに対するコード変更箇所は以下のとおりである．

- kvm のメインクラスの関数
  kvm_init()：KVM を読み込む時
  kvm_exit()：KVM をアンロードした時
  kvm_create_vm()：VM を起動した時
  kvm_destroy_vm()：VM をシャットダウンした時
  kvm_sched_in()：コンテキストスイッチ開始
  kvm_sched_out()：コンテキストスイッチ終了
  kvm_dev_ioctl()：キャラクタデバイス経由のデータやり取り
- kvm のライブラリー
  キャラクタデバイスの利用するためのプロトタイプ宣言

上記において，既存の KVM カーネルモジュールに 35 行を変更し，新規部分は合計 1818 行となった．新規追加部分について以下に述べる．

DVFS 制御機構では，CPU のモデル固有レジスタ (MSR, Model Specific Register) の読み書きを行なった．実際には，RDMSR と WRMSR という命令で MSR にアクセスし，CPU の周波数・電圧を変更した．パフォーマンスカウンタ管理機構では，DVFS 制御機構と同様に，MSR にアクセスしパフォーマンスカウンタの値を取得する．VM 管理機構では，KVM のメインクラスと連携し，Linux カーネルに実装されるリンクリストを用いて，VM の情報を管理する．学習機構では，性能の学習時に，VM に関する性能を計算する必要がある．VM の性能指標の単位として IPS(Instruction Per Second) を導入した．VM の性能は，VM タイムスライス単位で IPS を計測した．周波数決定機構では，図 3 のアルゴリズムを用いて，最適な周波数を決定する．

また，ホスト OS の Linux のユーザ空間にユーザインタフェイスと予測モデル生成機構を新規ファイルとして作成した．ユーザインタフェイスでは，VM の性能閾値の設定，モードの切替などを行なう．予測モデル生成機構では，VMM からパフォーマンスカウンタの値，VM の性能比を表す値と電力測定器から消費電力の値を用いて，予測をモデル化する．モデル化されたものを周波数機構に伝えて消費電力の制御が可能となる．

今回は Phenom9500 を実装のターゲットにしているが，近年多くの CUP においてパフォーマンスカウンタを設計し，本手法を利用することで他のアーキテクチャでも適用である．また，VM やホスト OS から得られたネットワーク I/O に関する情報などを用いて，さらに精度の高い消費エネルギー予測が可能になると予想できる．なお，本手法ではあらかじめ，VM 上で複数のベンチマークプログラムについて実行し，パフォーマンスカウンタの値と性能・消費電力の相関関係を求めておく必要がある．

5. 評価

本手法について，SPEC2006 とネットワークのベンチマークを用いて，消費エネルギーの評価を行なった．本章

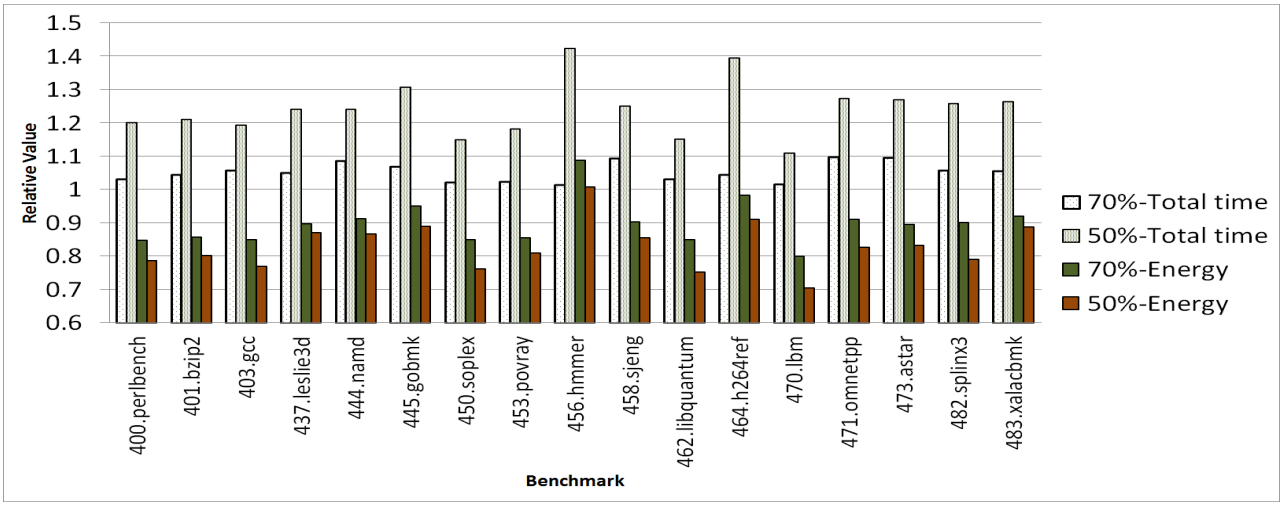

図 4 VM1 個の評価

ではこれらの詳細について述べる．評価には，Phenom9500 を搭載した PC を評価マシンとして利用した．評価マシンの環境を表 1 に示す．

表 1 評価マシン環境

CPU	AMD Phenom 9500(4 Cores)
メモリ	4GB
DVFS	Step0(2.2Ghz,1.200V), Step1(2.0Ghz,1.150V) Step1(1.8Ghz,1.125V), Step3(1.6Ghz,1.100V) Step4(1.4Ghz,1.050V)
仮想化	kvm-kmod-3.4
各 VM の VCPU	4
各 VM のメモリ	512 MB

5.1 SPEC2006 に関する評価

SPEC2006 の中から様々な特性を持つベンチマークを選んで，評価を行なった．VM の性能閾値=50%，VM が 1 個である場合，最大 29.49%の消費エネルギーの削減，VM が複数個である場合，最大 28.00%の消費エネルギーの削減を確認した．また，全ての評価に対して，VM の性能制約の範囲で処理を終了させることができた．VM が 1 個である時の評価結果は図 4 に示し，VM が複数個である時の評価結果は図 5 に示す．また，図 5 に表したシナリオ番号は，表 2 のベンチマークの組合せである．

図 4 において，X 軸は SPEC2006 のベンチマーク，Y 軸は各ベンチマークにおける処理時間および消費エネルギーの値を，全コアで最高周波数に設定した場合を 1 とした時の相対値を表している．消費エネルギーの削減は最大でメモリバウンド 470.lbm が 29.49%(VM の性能閾値=70%の時に 20.01%) となった．その次は，462.libquantum と 450.soplex となった．VM 上でメモリバウンドベンチマークが実行される場合，処理中でのキャッシュミスが頻繁に起きたとわかった．CPU のキャッシュに必要なデータが存在しないため，メインメモリからデータを取得してく必要がある．その時，本システムは，CPU の周波数を下げても性能がほとんど落ちないと判断し，低い性能閾値で消費エネルギーの削減率が高かった．また，メモリバウンドベンチマークの処理時間を見ると，増加率は低かった．一方，CPU バウンド 464.h264ref と 445.gobmk は消費エネルギーの削減が低かった．ただし，消費エネルギーが増加した 456.hmmer は例外だった．一般的に，VM 上で CPU バウンドベンチマークが実行される場合，キャッシュヒット率が高いとわかった．ほんとんどの処理時間でキャッシュアクセスを行なうため，周波数を下げることができないため，消費エネルギーの削減率が低かった．また，CPU バウンドベンチマークの処理時間を見ると，増加率は高かった．本システムは，できる限り消費エネルギーを削減するため，性能制約の上限まで処理時間を延ばそうとすることがわかった．456.hmmer について，キャッシュヒット率が頻繁に変化し，さらに，処理中でデータベース I/O とのやり取りも多かった．本手法でモデル化した予測には，I/O に関する情報を追跡しないため予測値が外れた．高周波数で処理を終わらせるべきところに，低周波数を設定したため，処理時間が延びて，結果は逆に消費エネルギーが増加してしまったと考えられる．

図 5 において，X 軸はシナリオ番号，Y 軸は各シナリオにおける処理時間および消費エネルギーの値を，全コアで最高周波数に設定した場合を 1 とした時の相対値を表している．処理時間と消費エネルギーは全てのベンチマークが終了した時の値を示す．また，同じ組合せにある全ての VM には同じ性能閾値を設定した．VM の性能閾値=50%の時に，シナリオ 1 が最大 28.00%の消費エネルギーを削減でき，次はシナリオ 2 であった．シナリオ 1 にある 470.lbm，シナリオ 2 にある 450.soplex はメモリバウンドベンチマークであるため，複数の VM が起動しても，1 個の VM の時と同様に消費エネルギーの削減効果が高いとわかった．

表 2　ベンチマークの組合せ

シナリオ	ベンチマーク		
	VM1	VM2	VM3
1	453.povray	470.lbm	-
2	483.xalacbmk	450.soplex	-
3	458.sjeng	483.xalacbmk	-
4	464.h264ref	403.gcc	471.omnetpp

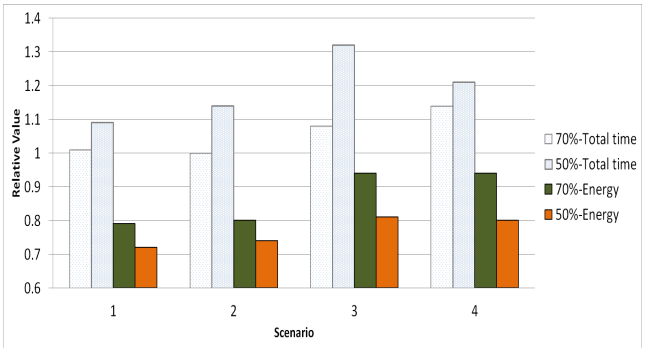

図 5　VM 複数個の評価

5.2　ネットワークベンチマークに関する評価

ネットワークベンチマークである iperf，httperf を用いて評価を行なった．評価方法として，VM1 を Web サーバとして起動しており，別のクライアントから VM1 にネットワークベンチマークを用いて負荷をかけた．httperf を使うことにより，VM1 にアクセスして index.html をリクエストした．また，httperf は 1 秒あたりのリクエスト数（以降，rate と呼ぶ）を変更するとこにより，VM1 に負荷を加減することができた．今回の評価で行なったリクエスト総数を 60000 回にし，rate=100 で 600 秒，rate=200 で 300 秒，rate=500 で 120 秒，接続を終了した．また，iperf を用いて，TCP コネクションでサーバとクライアントの間のスループットを測定した．評価結果は図 6 に示す．

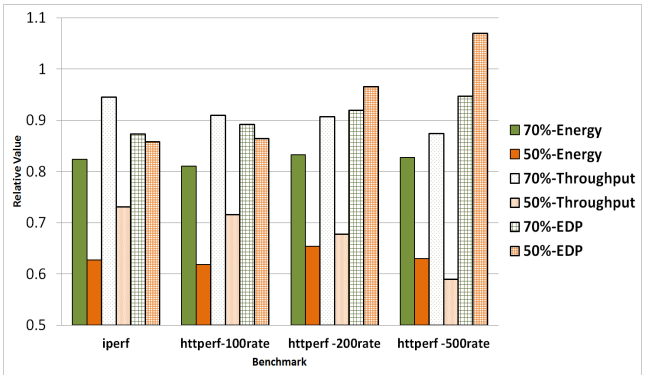

図 6　ネットワークの評価

図 6 において，X 軸はベンチマーク，Y 軸は各ベンチマークにおける消費エネルギー，スループットおよび EDP の値を，全コアで最高周波数に設定した場合を 1 とした時の相対値を表している．EDP(Energy Dealy Product) はスループット当たりの消費エネルギーを表す指標であり，EDP が小さいほどエネルギー効率が高い．VM の性能閾値=50%で，最大 38.15%の消費エネルギーの削減を確認した．消費エネルギーの削減率は性能閾値が低いほど効果が高いとわかった．VM 上でネットワークベンチマークが実行される場合，CPU の周波数を下げても性能がほとんど落ちないため，低い周波数で消費エネルギーを削減できたと考えられる．VM のスループットに関して，性能閾値が低いほど，スループットが低下した．VM の性能閾値=50%で，最大 41.06%に下がった．EDP に関して，VM の負荷にかかる量により変動する．rate 値が大きくすると，負荷が大きくなり，EDP も大きくなった．処理時間単位で，消費エネルギーを大幅に削減できたが，周波数を下げた場合は，EDP が得になっていない場合がある．

6.　関連研究

仮想化環境における省電力化は，VM の CPU 負荷状況から DVFS 制御を行なう研究として，Kamga ら [13] の研究が存在する．Xen の credit スケジューラに VM の CPU 負荷状況の監視機構と CPU 時間の制御機構を追加し，既存なプラットフォームの変更が少ないという利点がある．しかし，各 VM 上で動作するアプリケーションがメモリバウンドである場合，周波数を落しても，処理性能がほとんど落ちない場合が存在する．このような場合において，DVFS 制御を行わないため，省電力化が全く行わないという欠点がある．S.Kundu ら [14] は CPU やメモリの使用状況などに基づく VM 上で動作するアプリケーションの性能のモデル化を提案する．アプリケーションが動作する間に，物理 CPU やメモリから得た情報を用いて，VM の性能を予測した．これに対して，本研究はパフォーマンスカウンタを用いて，性能・消費電力の予測を行なった．VM の特性を判断するために，パフォーマンスカウンタを用いた他の研究も存在する．Bertran ら [15] は，パフォーマンスカウンタを用いて回帰分析で VM の消費電力を予測した．ただし，本手法と異なって自動的にコンテキストスイッチ単位ではなく，秒単位でパフォーマンスカウンタを取得するため，学習時に手間がかかった．Dhiman ら [16] は，混合正規分析モデルで VM の消費電力を予測した．これらの手法は予測の過程に終わり，実際のシステムに適用して省電力化を行わなかった．また，クライアントーサーバ型の仮想化環境における省電力制御の研究が存在する [17]．サーバが中心となり，クライアントの負荷状況により VM レベルのオン・オフを行なった．また，Nathuji ら [18] は，VM と VMM が協調して省電力化を行なう手法を提案する．

仮想化環境において，VCPU スケジューリングによる省電力化の研究も存在する．吉田ら [19] は，マルチコアの消費電力特性を考慮した VCPU スケジューラを提案した．その結果，全てのコアに低い周波数が下がる確率が高くなり，DVFS による消費エネルギー削減効果が高くなると示

した．ゲスト OS の負荷と消費電力特性をフィードバックとして，VCPU のスケジューリングを行なう省電力化システムを C.Wen ら [20] が提案した．

システム全体の消費電力のうち，CPU は大きな割合を埋める場合が多いことから，CPU に関する省電力化の研究が多かった．しかし，ストレージデバイスやネットワークカードの消費電力に関する研究も存在する．L.Ye ら [21] は，VMM と VM のディスク I/O 情報を追跡することにより，ディスク・スピンを減らしてディスクのスリープ時間を延ばすことで消費電力を削減した．

7. おわりに

本稿では，仮想化環境における省電力化を目的とし，消費エネルギー予測に基づいた省電力化を行なう VMM の設計，実装と評価について述べた．本手法では，VM 実行時のキャッシュミス率やプロセッサ全体のメモリアクセス頻度をハードウェアカウンタから取得し，回帰分析手法により求めた最適な性能・消費電力をもとに DVFS 制御を実施した．実装には，Linux カーネルを用いた VMM である KVM を用いた．評価より，コア単位で DVFS 制御可能なマルチコア環境において，性能条件の範囲内でメモリバウンドベンチマークが最大 29. 49%，ネットワークベンチマークが 38.15%の消費エネルギーの削減を確認することができて，本方式の有用性を示した．

今後の課題として，VM やホスト OS から得られたネットワーク I/O に関する情報などを用いて，マルチコアの消費電力特性とスループットを考慮する予測モデルを考えることが挙げられる．

参考文献

[1] Intel: Enhanced Intel SpeedStep Technology for the Intel Pentium M Processor, http://www.intel.com/design/intarch/papers/30117401.pdf
[2] AMD: BIOS and Kernel Developer's Guide For AMD Family 10h Processors, http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/31116.pdf
[3] D. Brodowski: Linux kernel CPUfreq subsystem, http://www.kernel.org/pub/linux/utils/kernel/cpufreq/cpufreq.html
[4] Xen project: http://www.xenproject.org/
[5] KVM for Server Virtualization: An Open Source Solution Comes of Age, http://www-03.ibm.com/systems/resources/systems_virtualization_idc__kvmforservervirtualization.pdf
[6] Intel VT-x: Intel Virtualization Technology and Intel Active Management Technology in Retail Infrastructure, http://www.intel.com/design/intarch/papers/316087.pdf
[7] AMD-V: http://sites.amd.com/uk/business/it-solutions/virtualization/Pages/amd-v.aspx
[8] M. Weiser, B. Welch, A. Demers, S. Shenker: Scheduling for reduced CPU energy, In Proc. of Sysmposium on Operating Systems Design and Implementation, Article No.2, 1994.
[9] W.Wu, M.Martonosi, D.W.Clark, V.J.Reddi, D.Connors, Y.Wu, J.Lee, D.Brooks: A Dynamic Compilation Framework for Controlling Microprocessor Energy and Performance, In Proc. of the 38th annual IEEE/ ACM International Symposium on Microarchitecture, pp. 271–282, 2001.
[10] W.Yuan, K.Nahrsted: Energy-efficient soft real-time CPU scheduling for mobile multimedia systems, In Proc. of the 19th ACM Symposium on Operation Systems Principles, pp. 149–163, 2003.
[11] 金井 遵，佐々木 広， 近藤 正章，中村 宏，天野 英晴，宇佐美 公良，並木 美太郎：消費エネルギー予測によるマルチコア環境向け省電力化 Linux スケジューラ，情報処理学会 第 20 回コンピュータシステム・シンポジウム，Vol.2008，No.12，pp. 77–86，2008.
[12] 林 和宏，金井 遵，丸山 勝巳，並木 美太郎：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，情報処理学会研究報告，「システムソフトウェアとオペレーティング・システム」，第 108 回研究報告，Vol.2008-OS-108，pp. 147–154，2008.
[13] C.M.Kamga, G,S.Tran, L.Broto:Power-aware scheduler for virtualized systems. In Proc. of Green Computing Middleware on Proceedings of the 2nd International Workshop, Article No.5, 2011.
[14] S.Kundu, R.Rangaswami, K.Dutta, M.Zhao:Application Performance Modeling in a Virtualized Environment, In Proc. of High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium, pp. 1–10, 2010.
[15] R.Bertran, Y.Bercerra, D.Carrera, V.Beltran, M.Gonzilez, X.Martorell, N.Navarro, J.Torres, E.Ayguade:Energy accounting for shared virtualized environments under DVFS using PMC-based power models, In Proc. of Future Generation Computer Systems, pp. 457–468, 2012.
[16] G.Dhiman, K.Mihic, T.Rosing: A system for online power prediction in virtualized environments using Gaussian mixture models, In Proc. of the 47th Design Automation Conference, pp. 807–812, 2010.
[17] G.Dhiman, G.Marchetti, T.Rosing: vGreen: A System for Energy-Efficient Management of Virtual Machines, In Proc. of the ACM Transactions on Design Automation of Electronic System, Vol.16 Issue 1, Article No.6, 2010.
[18] R.Nathuji, K.Schwan: VirtualPower: Coordinated Power Management in Virtualized Enterprise Systems, In Proc. of twenty-first ACM SIGOPS symposium on Operating systems principles, pp. 265–278, 2007.
[19] 吉田 哲也，山田 浩史， 佐々木，河野 健二，中村 宏：マルチコア CPU の電力消費特性を考慮した仮想 CPU スケジューラ，情報処理学会論文誌・コンピューティングシステム（ACS），Vol.4，No2，pp. 25–39，2011.
[20] C.Wen, J.He, J.Zhang, X.Long:PCFS: A Power Credit based Fair Scheduler under DVFS for Multi-core Virtualization Platform, In Proc. of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing, pp. 163–170, 2010.
[21] L.Ye, G.Lu, S.Kumar, C.Gniady, John H.Hartman: Energy-Efficient Storage in Virtual Machine Environments, In Proc, of the 6th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, pp. 75–84, 2010.