

分散型 Web サーバでの負荷変動を考慮した 省電力化のためのノード状態制御

今 田 貴 之^{†1} 佐 藤 三 久^{†1}
木 村 英 明^{†1} 堀 田 義 彦^{†1,*1}

本研究では、分散型 Web サーバ上において負荷変動を考慮しながら消費エネルギーを削減する手法について提案を行う。近年ではデータセンタ等における Web サーバの消費電力削減は電力制限や予算等の観点から解決すべき事項となっており、そのための省電力化手法が提案されている。従来手法では負荷変動の性質等想定される事前情報を先に与えることで QoS (Quality of Service) 性能維持およびエネルギー削減を行っているが、実運用時には事前の想定と現実との誤差が QoS 維持性能とエネルギー削減効率を大きく変動させると考えられる。そこで、我々は負荷増加量の考慮をトラフィック予測に基づいて時系列に変化させ、さらにサーバの消費電力プロファイルと併用することでサーバノード状態（スタンバイ/アクティブ/プロセッサ性能）を制御する手法を提案する。提案手法を評価した結果、十分な QoS 性能を維持しながら単純なワークロードでは約 20%、現実的なワークロードでは約 15–60%のエネルギー削減を達成することができた。

Server Node State Management Tracking Load Fluctuation for Energy-efficient Distributed Web Servers

TAKAYUKI IMADA,^{†1} MITSUHIKA SATO,^{†1}
HIDEAKI KIMURA^{†1} and YOSHIHIKO HOTTA^{†1,*1}

In this paper, we propose a scheme based on server nodes state control tracking load fluctuation on distributed Web servers for energy saving. Decreasing power consumption on Web servers is now a new challenging problem to be solved in a data center or warehouse. Some schemes for the servers has been proposed but they need pre-defined parameters about assuming load in order to achieve efficient energy saving and to keep QoS (Quality of Service) performance. However, such parameters is difficult to be set statically in advance because these parameters may have a large effect on possible QoS performance and energy saving. We proposed a novel scheme which manage server node

states (Stand-by/Active/Processor performance) employing a statistical prediction method and the server power profile. As a result, our scheme achieved about 20% (in synthetic workloads) and 15–60% (in realistic workloads) energy saving with keeping high QoS performance.

1. はじめに

近年、多数の計算機資源を有する計算機センタやデータセンタのような施設において消費電力に関する問題が表面化し、これが計算機運用における課題の一要因となってきている。一般に、インターネットサービスやデータベース等に用いられる各種サーバは処理能力を優先する構成となっているため、消費電力も増加してしまう。また、このような消費電力の増加がさらなる冷却装置の必要性を生み出す原因となっている。高解像度動画等のリッチコンテンツやサービスの多様化等アプリケーション側からの性能への要求は今後も増え続け、このような施設に対しさらにより多くの計算資源が必要となる。しかし、これ以上サーバを増やすことは電力そして容積的にも現状では厳しくなっている。そのため、性能–電力効率の良いシステムおよび効率を高めるための省電力化手法の必要性が高まっている。このような性能–電力効率の良いシステムを構築するためには、「低消費電力なシステムの開発」と「ソフトウェアによるハードウェア制御を用いた省電力化」が必要であると考えられる。前者はハードウェアによるところが大きく、マルチコア等の技術による性能–電力効率の向上があげられる。一方、後者についてはこれまでにソフトウェア側で省電力化を行う手法がモバイル用途を中心として提案されてきており、最近では各種サーバにもその省電力化の範囲が広がってきている。

本研究ではサーバの種類として Web サーバに焦点を当て、特に Web サーバの性能向上のためによく用いられている分散型 Web サーバの消費電力削減機構について提案を行う。従来手法では想定されるトラフィックの特徴を定数として与えることで十分な QoS 性能維持とエネルギー削減を達成することができたが、事前に与えられた特徴と実際のトラフィックの特徴との差異は当然ながら実際のトラフィック変動によって変化する。つまり、特徴の

^{†1} 筑波大学大学院システム情報工学研究科

Graduate School of Systems and Information Engineering, University of Tsukuba

*1 現在、株式会社ルネサステクノロジ

Presently with Renesas Technology Corporation

与え方により達成できる QoS 性能（追従性能）とエネルギー削減率が大きく変化し、場合によっては大幅な QoS 性能低下やエネルギー削減の制限として影響するおそれがある。このような影響を避けるためには、実際のトラフィック変化を短期的にそのつどとらえ、与える特徴を変化させながらサーバを制御していくことが望ましい。また、最近ではプロセッサの TDP（Thermal Design Power）の減少等によってサーバシステムにおいてもプロセッサ消費電力削減の余地が徐々に減少してきている。効果的な消費電力削減のためにはプロセッサの DVFS 機構制御に加えて不必要なアクティブ状態のサーバをスタンバイ状態へ移行させなければならない。しかし、スタンバイ状態からアクティブ状態への移行には十数 sec 程度必要となるため、状態変更に要する時間オーバーヘッドを考慮しながらサーバ状態を制御しなければ QoS 性能が低下する。そこで、本提案手法では局所的な統計的予測を用いて状態変更時間内に起こりうるサーバ負荷増加量をそのつど変化させて考慮する。これによりトラフィック変動に追従しながら効果的なエネルギー削減を実現する。

本研究では、最初に分散型 Web サーバにおける消費電力特性等の評価を行い、スタンバイ状態の有効性について確認した。実際、サーバ状態をスタンバイ状態へ移行させることによる消費電力の削減は非常に効果的であることが分かった。これらの評価を基にして、プロセッサ DVFS 機構とアクティブな台数の制御によって QoS を維持しながらエネルギー削減を行う手法について提案する。そして、提案手法によってサーバ電力状態を制御するシステムを実装し、SPECWeb99 のファイルを使用するワークロードによって提案手法の評価を行った。

本論文の構成は以下のようになる。まず次章で Web サーバ上でのエネルギー削減についての関連研究について述べる。3 章では、分散型 Web サーバの消費電力特性について述べる。そして 4 章において消費電力プロファイルを判断基準とするエネルギー削減手法について提案する。5 章において本提案手法の評価および考察について述べる。6 章では実運用における提案手法の適用について述べ、最後に 7 章で本研究のまとめと今後の課題について述べる。

2. 関連研究

近年、シングルもしくは分散型の Web サーバにおいてエネルギー削減を行う手法がいくつか提案されてきている。

Elnozahy らは Web サーバがそれぞれのリクエストに費やした応答時間を記録しておき、目標となる応答時間との関係により DVFS 機構を制御して省電力化を行う手法を提案して

いる¹⁾。また、“Request batching”と呼ばれるプロセッサが低電力状態に到着したパケットの処理を遅らせることでプロセッサ状態を idle よりもさらに低電力状態へ移行させる手法についても述べている。Kolta らはプロセッサ動作周波数での性能低下を計算するために Instruction Per Clock の予測値を用いて、電力供給削減時における性能低下を抑制する手法について提案している²⁾。Bertini らはヘテロジニアスな分散型 Web サーバにおいて DVFS 変更時の消費電力プロファイルを用いて制約応答時間を満たしながらエネルギーを削減する手法について提案し³⁾、約 10% のエネルギー削減が可能であること示している。この手法ではそれぞれのプロセッサに対する性能係数値の組がすべて同値の場合と同程度のシステム性能を達成する一方で消費電力を最小化するように算出される。Sharma らは、分散型 Web サーバ上で個々の HTTP リクエストではなく HTTP1.1 より使用可能なセッションを単位とするリクエスト制御とともに DVFS 制御を行う手法について提案している⁴⁾。これらの手法はすべて DVFS 制御によるプロセッサ性能制御のみを行っており、処理サーバ台数については考慮していない点が我々の手法とは異なる。

Elnozahy らは、各サーバ上で動作している DVFS 制御が決定するプロセッサ周波数がある閾値を超えたときに処理サーバ台数を制御する手法について提案している⁵⁾。しかし、この手法では未来の負荷増加やサーバ起動時間の考慮は行われていない。また、Rajamani らは処理サーバ追加時間を考慮し、さらに追従性能向上のためにスベアを保持しておく手法について提案している⁶⁾。この手法ではコネクション数の時間増加量を時不変定数もしくはそれぞれの時点におけるワークロードのプロファイルとして保持しており、DVFS は利用せずの追加/削除のみを対象としている。Rusu らは静的および動的コンテンツの平均 CPU 負荷とそれに対する消費電力プロファイルを利用して処理サーバ台数最適化を行い、さらに DVFS 制御を処理サーバ側で独立に判断する手法を提案している⁷⁾。また、Rusu らの手法を基にして DVFS 制御にフィードバック制御を導入した手法が Bertini らによって提案されているが⁸⁾、処理サーバ台数制御は Rusu らの手法と同様である。文献 7)、8) の手法は本提案手法と同様に消費電力プロファイルを用いて最適化を行っているが、処理サーバ台数決定時に考慮する未来の負荷増加を時不変定数として与えている点が異なる。一方で、Xu らはプロセッサ周波数に基づく消費電力モデルによって最適な処理サーバ台数を決定する手法について提案している⁹⁾。この手法ではサーバ起動時間を考慮した処理サーバ台数制御が行われなため、サーバ起動時間が長いほど突発的なリクエストレート増加に対する QoS 性能維持が困難となる。

Chen らは、未来の負荷を予測して処理サーバ台数制御を行う手法を提案している¹⁰⁾。本

表 1 Web サーバの仕様
Table 1 Specification of Web server.

Processor	Intel Core2Duo E6750 (2.66 GHz)
Memory	DDR2-800 4 GB
Chipset	Intel G33
Network	Intel Gigabit Ethernet MT1000 Server (PCI-Express)
HDD	HGST HDP725025GLA380
OS	Linux kernel 2.6.24.2
Web Server	Apache 2.2.3-11

提案手法との本質的な違いは予測対象を次の時区間における平均リクエストレートとして
いることである。Chen らの手法では次の時区間における負荷を予測してその時区間にお
ける処理サーバ台数を決定し、その時区間内では DVFS 制御のみを行う。予測値と実際値と
の誤差が QoS 性能の大幅低下やエネルギー削減率に大きく影響するため、十分な予測精度
を前提としている。しかし、Chen らは「十分な予測精度の実現には十分に大きな時区間の
長さが必要」と述べている。この場合、十分に大きな時区間内でのリクエストレート変化に
対して処理サーバ台数の制御を行うことができない。一方、提案手法では処理サーバ台数を
制御する時区間単位の長さを小さくすることに制限はなく、細粒度な制御が可能である。

3. 分散型 Web サーバの消費電力特性

本章では、分散型 Web サーバ上での省電力化を考えるための指針として、サーバ性能と
消費電力特性を調べるためのベンチマークによる評価を行い、その結果について述べる。

3.1 評価環境

表 1 に使用したサーバの仕様を示す。このサーバに用いられているプロセッサは 2.66 GHz、
2.33 GHz および 2.0 GHz と 3 段階の周波数で動作可能である¹¹⁾。また、ベンチマークには
SPECWeb99¹²⁾ を使用した。SPECWeb99 は Web サーバが対応できる最大同時セッシ
ョン数を計測するための Web サーバベンチマークソフトウェアである。そして、Web サー
バへのアクセス分布を表すものとしてよく用いられる Zipf 分布¹³⁾ に基づいて静的な html
ファイルと Perl による動的な CGI スクリプトにアクセスを行う。本評価での実サーバにお
ける CGI の処理では、通常の Perl のインタプリタによる処理速度を向上させるためにサー
バ側で FastCGI プロトコルの実装の 1 つである FCGI-0.67¹⁴⁾ と Apache 向けのモジュール
である mod_fastcgi-2.4.6¹⁴⁾ を用いて高速化を行った。

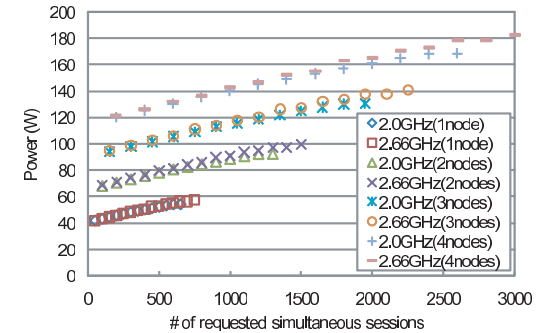


図 1 実サーバの待機消費電力を考慮したときの要求同時セッション数と消費電力の関係

Fig. 1 Relation between requested simultaneous sessions and server power consumption.

消費電力測定には我々が開発した PowerWatch¹⁵⁾ を用いた。PowerWatch ではホール素
子を使用することで電源コードを断線させることなく電流値の測定をすることが可能であ
る。本評価ではマザーボードへ供給される ATX+12 V DC, ATX+5 V DC, ATX+3.3 V
DC, CPU+12 V DC, そして HDD へ供給されている +12 V DC および +5 V DC それぞ
れの電流値を測定し、これらの値より上記 Web サーバ 1 台の消費電力を算出した。

3.2 評価結果

図 1 に 1-4 台の Web サーバに対して要求同時セッション数を変化させたときの消費電力
推移を QoS 満足範囲内で示す。なお、SPECWeb99 で定める QoS は

「要求同時セッション数のうち 95%以上が 320 kbps 以上のスループットを得る」
として定義されている。本評価では、最初にサーバ 1 台でセッション数 S/N 処理時の消費
電力 $P(1, g, S/N)$ 、サーバ 1 台の待機消費電力 P_w (スタンバイ時, 3.96 W) の値を実測
し、Gear g で動作するサーバ N 台でセッション数 S 処理時の消費電力 $P(N, g, S)$ を、

$$P(N, g, S) = N \times P(1, g, S/N) + (N_{total} - N) \times P_w \quad (1)$$

として上記評価式により算出した (N_{total} はあらかじめ用意されたサーバ台数を意味する)。

図 1 より要求同時セッション数が高くない場合には、DVFS 制御もしくは処理サーバ (以
下、アクティブなサーバ) の台数制御によって消費電力の削減が可能であることが分かる。
Barroso らの文献 16) によれば、数千台の Web サーバの調査によってそれらは大半の時間
において利用率が 10–50%の間で動作していると報告されている。たとえば、図 1 において
QoS を満たすシステムの最大性能を利用率 100%とすれば、利用率が 40–50%程度の場合に

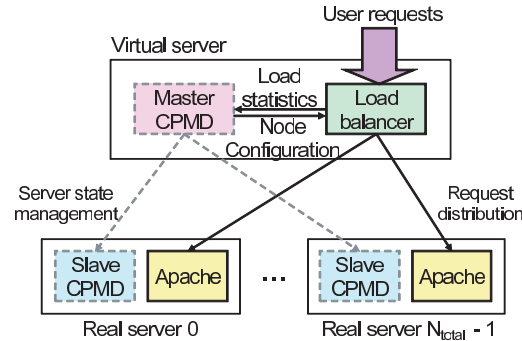


図 2 分散型 Web サーバ上での電力制御機構

Fig. 2 Power management framework on distributed Web servers.

は要求同時セッション数が 1,200–1,500 程度のときに該当する。このとき DVFS 制御のみの手法では 2 W 程度の消費電力削減にとどまるのに対し、アクティブなサーバ台数の制御手法では約 52 W, 両手法を併用することで約 54 W の消費電力を削減することが可能である。そして、アクティブなサーバ台数を N_{total} 台必要としない要求同時セッション数の場合には、つねに同様の状況となっている。つまり、DVFS 制御のみの手法では効果的な消費電力の削減を行うことは困難であり、積極的にアクティブなサーバ台数を制御していく手法のみが効果的なエネルギー削減に対して有効であることが分かる。

4. Power-Aware な分散型 Web サーバシステム

4.1 分散型 Web サーバにおける省電力化機構

一般的に分散型 Web サーバはユーザからのリクエストをバックエンドの処理サーバへ中継する「仮想サーバ」と、実際にリクエスト処理を行う複数の「実サーバ」から構成される。本研究では実サーバ状態制御の枠組みとして仮想サーバと実サーバ群が協調して状態を制御する機構を考える。図 2 に今回実装した Cooperative Power Management Daemon (CPMD) におけるサーバ状態制御の仕組みを示す。仮想サーバ上の Master-CPMD がリクエスト負荷状況の情報をロードバランサから入手して QoS を満たす実サーバの電力最適状態（アクティブな実サーバ台数、プロセッサ性能）の決定および制御を行い、同時にロードバランサへ中継先実サーバの変更を依頼する。実サーバ上の Slave-CPMD はその決定に従い自サーバ状態を変化させる。本手法ではスタンバイ状態への移行を Suspend to RAM

(ACPI S3 state) によって実現する。また、アクティブな状態への移行は Master-CPMD から発行される Wakeup On LAN パケットによって行われる。

4.2 省電力化アルゴリズム

本節では提案する省電力化アルゴリズムについて述べる。本アルゴリズムは先行研究^{6)–8)}と同様、実際のリクエストレートだけでなく実サーバ追加時間内に増加しうるリクエストレートの量（以下、マージン）を考慮して電力最適な実サーバ状態を決定する。しかし、本アルゴリズムがこれらの先行研究手法と大きく異なるのは実際のワークロード変化をとらえてマージンを時系列に変化させている点である。

本研究ではサーバシステム的环境として

- Web サーバシステムに対するユーザからの最大リクエストレート R_{total} が与えられ、その値はサーバシステム性能の飽和を起こすリクエストレート値を超えることがない、という条件を仮定する。一般に、Web サーバシステムの構築では上記の値を評価基準の 1 つとしてサーバの構成等を決定することが多い。よって、上記のような仮定は妥当であると考えられる。本アルゴリズムではスイッチ等のネットワーク機器や仮想サーバの省電力化は行わず実サーバのみについて注目する。また、リクエスト負荷分散手法には既存のものを用いるものとする。

本アルゴリズムは先行研究^{7),8)}と同様に実サーバの消費電力プロファイルに基づき実サーバ状態を決定する。最初に、アクティブな実サーバ台数 N ($N = 1, 2, \dots, N_{total}$) およびプロセッサ Gear g のすべてのペアに対し、その条件下における QoS を満足する最大のリクエストレート $R_{max}(N, g)$ とそれを処理しているときの実サーバ群の消費電力 $P(R_{max}(N, g))$ (スタンバイ状態の消費電力を含む) を取得する。具体的にはリクエストレートを段階的に上昇させながらそのときの QoS 満足により $R_{max}(N, g)$, $P(R_{max}(N, g))$ の値を算出する。特に実サーバに均等な負荷分散アルゴリズムを適用する場合には、

$$\begin{aligned} R_{max}(N, g) &= N \times R_{max}(1, g) \\ P(R_{max}(N, g)) &= N \times P(R_{max}(1, g)) + (N_{total} - N) \times P_w \end{aligned} \quad (2)$$

と仮定することができ、 $P(R_{max}(N, g))$ の値を実サーバ 1 台時の消費電力より求めることができるとする。

次に電力最適な実サーバ状態を決定する手法について述べる。実サーバ状態の決定に実サーバの追加/削除が必要な場合かの判断により、以下 2 種類の方法を考える。

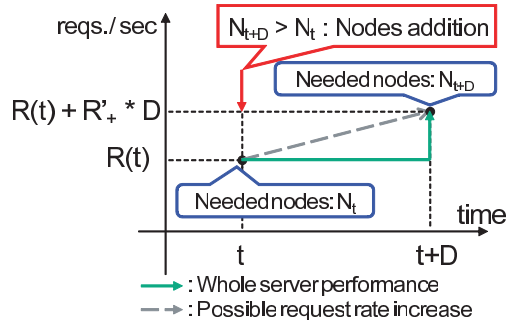


図3 実サーバ追加/削除時の実サーバ状態決定
Fig. 3 Decision making for adding / deleting real servers.

4.2.1 実サーバ追加/削除が不要な場合

時刻 t におけるアクティブな実サーバ台数が N_{total} に等しく、リクエストレート $R(t)$ を処理するために必要とされる実サーバ台数も N_{total} となる場合は実サーバの追加/削除を考慮する必要がない。そのためプロセッサの DVFS 制御をすることだけを考え、この場合では $P(R_{\max}(N_{total}, g))$ の最適化を行うことによって時刻 t において最適な Gear g を以下のように求めることができる。

$$\min_g P(R_{\max}(N_{total}, g)) \quad (3)$$

$$s.t. \quad R(t) \leq R_{\max}(N_{total}, g) (\leq R_{total}) \quad (4)$$

4.2.2 実サーバ追加/削除が必要な場合

図3に実サーバ追加/削除時の実サーバ状態決定方法を示す。時刻 t におけるリクエストレート reqs./sec が $R(t)$ のとき、スタンバイ状態の実サーバが完全に復帰する最小時間 $Dsec$ の間に増加しうる（減少は考慮しない）リクエストレート量、つまり考慮すべきマージンについて考える。1 sec あたりのリクエストレート増加量 reqs./sec² を R'_+ とすると、増加しうるリクエストレート量は $R'_+ \times D$ と計算できるため、この値を考慮しながら実サーバ状態を決定していく。このとき、以下の最適化式を解くことによって最適な数 N およびプロセッサ Gear g を得ることができる。

$$\min_{N, g} P(R_{\max}(N, g)) \quad (5)$$

$$s.t. \quad R(t) + R'_+ \times D \leq R_{\max}(N, g) \quad (6)$$

なお、リクエストレートの減少を考慮しない ($R'_+ \geq 0$ とする) のは、QoS 性能低下の回避を消費電力削減よりも優先させるためである。

式 (6) においてマージンを大きく設定すれば消費電力は多くなるが急激なリクエストレート増加に対しても QoS 性能を低下させることなく運用が可能である。逆にマージンを小さくすれば消費電力はおさえられるが、急激なリクエストレート増加に対して実サーバの追加が必要な場合に QoS 性能を低下させる可能性がある。先行研究^{(6)–(8)}では想定されるワークロードの特徴情報（リクエストレート増加量の最大値等）を事前に定数として与えてマージンが一意に決定されるため、マージンが時系列に変化することはない。たとえば特徴情報をリクエストレート増加量の最大値で事前に与えた場合、前章で示したような消費電力特性を持つ実サーバ群からなるシステムではマージンの過大見積りによって効果的なエネルギー削減が達成できない可能性がある。

理想的にはリクエストレートの増加が大きい傾向の期間にマージンを大きくして実サーバの追加を行い、リクエストレートの増加が小さい傾向の期間にマージンを小さくして余剰な実サーバ台数を制限する方が望ましい。そこで、本手法では R'_+ がリクエストレートの変動に対応するような仕組みを統計に基づく予測のアルゴリズムにより実現する。現在では自己回帰 (AR) モデルや自己回帰平均移動 (ARMA) モデルのような統計的予測手法が知られており⁽¹⁷⁾、これらの手法により過去の数ステップの値に対して、次（もしくはさらに先の）ステップ値を求めることができる。たとえば、ステップ s において、過去 M ステップの値 $x(s-M+1), \dots, x(s)$ を用いて次ステップの値 $x(s+1)$ は予測手法を表す関数 f により、
$$x(s+1) = f(x(s), x(s-1), x(s-2), \dots, x(s-M+1)) \quad (7)$$
 として表される。このような統計的予測手法はネットワークトラフィック予測にも応用されている。

CPMD 機構では、時刻 t の時点で次のステップ（期間）における 1 sec あたりの平均的リクエストレート増加量 $\tilde{R}'_+(s)$ を過去 M 期間の値によって予測する。図4に $\tilde{R}'_+(s)$ の概要を示す。 s 番目の期間 s -th period は U 個からなる等間隔（長さ T_{short} sec）のさらに小さな期間 i -th tick ($i = 0, 1, \dots, U-1$) より構成されると考える。式 (6) の場合と同様に $\tilde{R}'_+(s) \geq 0$ を想定するため、それぞれの tick におけるリクエストレートの傾き $R'(i, s)$ に関して 0 より大きい値の平均値を $\tilde{R}'_+(s)$ と定義する。単純な正の増加量のみを考慮することで予測を誤った際のリスクを低減させることができる。なお、すべての tick において $R'(i, s)$ の値が 0 以下の場合は $\tilde{R}'_+(s) = 0$ であるとする。つまり、 $\tilde{R}'_+(s)$ は

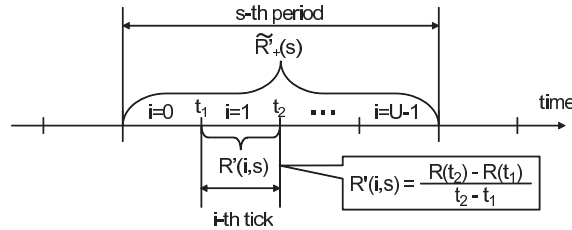


図 4 ある期間におけるリクエストレート増加量 $\tilde{R}'_+(s)$
Fig. 4 Request rate increment $\tilde{R}'_+(s)$ in a period.

$$\tilde{R}'_+(s) = \begin{cases} \frac{\sum_{i=0}^{U-1} \{R'(i,s) | R'(i,s) > 0\}}{|R'(i,s) | R'(i,s) > 0|_i} & (\text{if } \exists i R'(i,s) > 0) \\ 0 & (\text{otherwise}) \end{cases} \quad (8)$$

で与えられる．たとえば $U = 3$ の場合，

$$(R'(0,s), R'(1,s), R'(2,s)) = (1, -3, 7)$$

であれば $\tilde{R}'_+(s)$ は $(1 + 7)/2 = 4$ となる． $R'(i,s) \leq 0$ となる値を排除することは， $\tilde{R}'_+(s)$ の正值性を保証すること，および平均値の低下を回避させて QoS 性能低下のリスクを低減させるためである．

このようにして得られる $\tilde{R}'_+(s)$ により，その次の期間における予測値 $\tilde{R}'_+(s+1)$ は，統計的予測手法を表す関数 f によって，

$$\tilde{R}'_+(s+1) = f(\tilde{R}'_+(s), \tilde{R}'_+(s-1), \dots, \tilde{R}'_+(s-M+1)) \quad (9)$$

として求められる．この予測値 $\tilde{R}'_+(s+1)$ を用いて，

$$\begin{aligned} \min_{N,g} \quad & P(R_{\max}(N_{\text{total}}, g)) \\ \text{s.t.} \quad & R(t) + \tilde{R}'_+(s+1) \times D \leq R_{\max}(N, g) \end{aligned} \quad (10)$$

を解くことで最適実サーバ台数 N と最適プロセッサ Gear g を求める．このような手法を T_p sec ごとに適用し，定期的に動的に実サーバ状態を更新していく（つまり， s -th period の長さは $T_p = U \times T_{\text{short}}$ となる）．

5. 評価実験

提案手法の評価を行うために CPMD の実装を行い，分散型 Web サーバ上においてその

表 2 仮想サーバの仕様
Table 2 Specification of virtual server.

CPU	Intel Xeon 5140 (2.33 GHz)
Memory	DDR2-677 2 GB (ECC)
Chipset	Intel 5000X
Network	Intel Gigabit Ethernet MT1000 Server Dual (PCI-X)
OS	Linux kernel 2.6.25.4
Load Balancer	Linux Virtual Server (ipvs-1.2.1)

性能およびエネルギー削減の評価を行った．この評価では先行研究^{6)–8)}に見られるマージンが定数で与えられる手法との比較を中心に行った．

5.1 評価システム

評価に用いたシステムは，ソフトウェアロードバランサを有する仮想サーバ 1 台とリクエスト処理を行う実サーバ 4 台を Gigabit Ethernet で接続したものである．なお，実サーバには 3 章での消費電力評価で用いたものに一部変更を加えて使用した（一連の実サーバ削除・追加処理にかかる最小時間 D は約 13 sec であった）．表 2 にシステムで用いた仮想サーバの仕様を示す．仮想サーバ上で動作するソフトウェアロードバランサには Linux Kernel で提供されている Linux Virtual Server (LVS)¹⁸⁾ の実装である ipvs-1.2.1，およびその制御には ipvsadm-1.24¹⁸⁾ を用いた．LVS は Layer-4 におけるロードバランサであり，IP アドレスとポート番号に基づいてロードバランシングを行う．また，ネットワーク構成には実サーバで処理されたリクエストがロードバランサを介してユーザへ返される NAT 方式を使用した．負荷分散アルゴリズムには ipvs-1.2.1 で実装されている Least-Connect (LC) に変更を加えたものを使用した．LVS における LC アルゴリズムは処理が終わったコネクション (TCP における “TIME_WAIT” 状態のもの) を考慮して次の中継先実サーバを決定するため，これを考慮しないように変更を行った．これは長期間スタンバイ状態だった実サーバを追加するとそのサーバにのみリクエストが転送され続ける可能性があるためである（この問題は文献 6) で述べられているように，LVS 特有のものである）．

提案手法のリクエストレート増加量予測には AR モデル

$$x(s+1) = \sum_{i=0}^{M-1} \varphi_i x(s-i) + \epsilon_{s+1} \quad (11)$$

(ϵ_{s+1} は予測誤差) を適用して次ステップ値の予測を行った．このとき，与えられるワークロードの全体から得られる統計情報ではなく局所的な統計情報のみを用いてパラメータ推定を行った．具体的には，時間 T_p ごとにそのつど $\tilde{R}'_+(s-i)$ ($i = 0, 1, \dots, 7$) として取得

し、それらよりパラメータ推定を行った。

提案手法では実サーバ状態の更新間隔 T_p を短くして細粒度な制御を行うことでより高い QoS 性能維持およびエネルギー削減を行うことが可能である。しかし、あまりに細粒度すぎる制御はそのオーバーヘッドにより本来の仮想サーバとしてのリクエスト中継性能の低下を生み出す可能性がある。本評価では T_p を 3sec として仮想サーバ性能への影響が出ないように値を設定した。また、パラメータ推定の際の M の値は実サーバの追加時間 D_{sec} におけるリクエストレート変動より予測を行うものとして $M = 4$ ($M \times T_p = 12\text{sec}$) とした。

5.2 ワークロード

本提案手法の性能を評価するために、3 章で用いた SPECWeb99 と同様のファイルセットに対するリクエストを行うベンチマークプログラムを httpperf¹⁹⁾ をベースにして作成した。このベンチマークプログラムは SPECWeb99 のワークロードジェネレータと同様のファイルアクセス頻度（ファイルサイズに依存）に基づいて Web サーバに対してリクエストを発行し、さらに 1sec 単位でリクエストレートを段階的に変化させることが可能である。本評価において SPECWeb99 のワークロードジェネレータを使用しなかった理由は、SPECWeb99 が Web サーバの最高性能を評価するためのベンチマークであるためである。SPECWeb99 では想定するアクセスユーザ数をパラメータとして与え、その数をサーバが処理可能かどうか判断するだけである。つまり、長時間にわたるアクセスユーザ数の変化をシミュレートするものではない。

評価に用いるワークロードには「Synthetic」および「Realistic」の 2 種類を作成した。以下に両ワークロードの特徴を示す。

(1) Synthetic ワークロード

このワークロードは、

- リクエストレートを数 sec 間隔で段階的に増加させる、
- そのまま比較的高いリクエストレートを 20sec 維持させる、
- リクエストレートを数 sec 間隔で段階的に 0 reqs./sec 付近まで減少させ、そのまま無アクセスレートに近い状態を 20sec 維持する、

という一連の流れから構成される。本提案手法がどの程度の負荷増加まで QoS 性能を維持しながらエネルギー削減を行えるかどうかを評価するものである。

(2) Realistic ワークロード

1998 年ワールドカップの Web サイトへのアクセス履歴^{20),21)} を基に作成したワークロードである。このワークロードにより本提案手法が現実的なリクエストレート変動

表 3 1 台の実サーバで応答時間が 80 msec 以下のリクエストが総リクエスト数の 95%以上という QoS を満たすときの最大リクエストレートとそのときの消費電力

Table 3 Power consumption of a real server under satisfying required QoS.

Gear	Request rate (reqs./sec)	Power (W)
0 (2.66 GHz)	1,380	56.81
1 (2.33 GHz)	1,260	54.39
2 (2.0 GHz)	1,140	51.21

に対してどれほど効果的であるかについて評価を行う。

実サーバ 4 台使用時にネットワーク帯域幅がボトルネックとなったため、Compress-Zlib-1.4.2²²⁾ を用いて CGI アクセス時に返される内容に gzip 圧縮を行うように変更を加えた。文献 23) で述べられているように、このような圧縮処理は実際の Web サーバにおいて行われているものである。現在は特に gzip 圧縮を用いたものが多い。

本評価における QoS の定義は

「ワークロード中の総リクエストのうち、95%以上のリクエストの応答時間が 80 msec 未満である」

と設定した。SPECWeb99 を高負荷で実行したときの応答時間（QoS 満足下）が gzip 圧縮ありの場合で約 85 msec、gzip 圧縮なしの場合に約 60 msec 程度であったことをふまえ、つねに実サーバが高負荷時でないことを考慮してこのような値を用いた。また、95%という値は SPECWeb99 と同様の設定である。

表 3 に 1 台の実サーバ上での上記 QoS を満たすときの最大リクエストレート $R_{\max}(1, g)$ とそのときの消費電力 $P(R_{\max}(1, g))$ を各 Gear ごとに示す。これらの値を用いてそれぞれの N および g について $P(R_{\max}(N, g))$ の値を式 (2) より算出し、これらの値を基に提案手法の評価を行った。

5.3 Synthetic ワークロードによる評価

Synthetic ワークロードによる評価は、どの程度のリクエストレート増加まで提案手法がワークロード変化に追従してエネルギー削減を行えるかどうかを調べるためのものである。本評価では

「リクエストレートが実サーバ追加時間 D_{sec} の間平均的に増加したとき実サーバ 1 台で処理可能な最大リクエストレートに達する」

というリクエストレート増加量（約 106 reqs./sec²）を基準として、計 5 種類（80, 106, 128, 160, 213, 360 reqs./sec²）のリクエストレート増加量をとる山型のワークロードを

表 4 比較対象における $\tilde{R}'_+(s+1)$
Table 4 $\tilde{R}'_+(s+1)$ values used in all Static methods.

increment	80	106	128	160	213	360
Static	80	106	128	160	213	360
Static_half	40	53	64	80	106	160
Static45	45	45	45	45	45	45

作成した。

5.3.1 比較対象

先行研究⁶⁾⁻⁸⁾に見られるマージンが時不変な手法の場合、そのマージン値が達成できる QoS 性能とエネルギー削減率を左右する。この評価では実際の特徴情報が想定のもとの関係するかどうかを基に以下に示す Static, Static_half および Static45 の 3 手法を用意した。表 4 に 3 手法で設定した $\tilde{R}'_+(s+1)$ の値を示す。Static 手法は $\tilde{R}'_+(s+1)$ をワークロードの値で与えた手法である。Static_half 手法は Static 手法で設定した値の 1/2 を $\tilde{R}'_+(s+1)$ としたものであり、Static 手法よりもエネルギー削減を優先した手法を想定している。両手法は実際のワークロード特徴情報が想定のものに関係する場合である。一方、Static45 手法は 1998 年ワールドカップ Web サイトの運用 61-62 日目（1 日あたりの総リクエストが比較的多かった期間²¹⁾）から得られたリクエストレートの正の傾き R'_+ の平均値である 45 reqs./sec² を $\tilde{R}'_+(s+1)$ としたものである。この手法は実際のワークロードより得られた値を適用した場合を想定しており、実際のワークロード特徴情報が事前のものに関係していない。なお、これらの 3 手法における実サーバ追加/削除の制御は式 (10), (11) と同様であり、 T_p の値も提案手法と同様に 3sec とした。

また、提案手法および上記 3 手法よりも疎粒度で単純な実サーバ制御を想定したのが Simple 手法である。この手法では疎粒度な制御において QoS 性能低下を回避するため、サーバ 1 台の実サーバで処理可能な最大リクエストレート値をマージンとして設定した。つまり、条件式 (11) を

$$s.t. R(t) + R_{\max}(1, Gear0) \leq R_{\max}(N, g) \quad (12)$$

で置き換え、 $T_p = 15$ sec としたものである。

5.3.2 QoS 性能評価

図 5 に 6 手法 (Proposed, Static, Static_half, Static45, Simple および No_control) 適用時における 6 種類のワークロードを実行したときの QoS 性能（応答時間が 80 msec 以下のリクエスト数）を示す。ここで、Proposed は提案手法、No_control はつねに 4 台の実

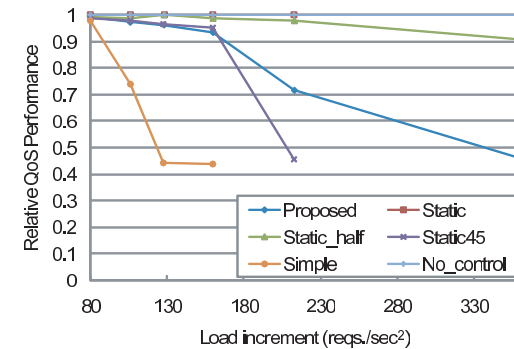


図 5 QoS 性能の相対比較

Fig. 5 Comparison of relative QoS performance.

サーバおよび最高性能 Gear のままの手法を意味し、No_control 手法の QoS 性能を 1 としたときの相対性能を表している（QoS 性能の大幅な低下によりシステムが不安定になり、評価が正当に行えなかったものは図中に掲載していない）。図 5 より、提案手法では基準となるリクエストレート増加量（128 reqs./sec²）までは若干の QoS 性能低下が見られるが、与えられた QoS 性能（95%）を満足していることが分かる。しかし、リクエストレート増加量が 160 reqs./sec² の場合では若干ではあるが QoS を満たさなくなり（性能低下は約 7%）、さらにリクエストレート増加量が 213 reqs./sec² の場合では大幅な QoS 性能低下が起きた。Static45 手法においても同様な QoS 性能低下が起きた。Simple 手法ではリクエストレート増加量が 80 reqs./sec² の場合は QoS を満足することができたが、それ以外では大きく QoS 性能が低下して 50%を下回る場合が見られた。一方で、Static 手法および Static_half 手法では極端な QoS 性能低下は見られなかった。

図 6 に増加量（128 reqs./sec²）のワークロード実行時における、No_control 手法以外の 5 手法が処理可能な最大リクエストレートの変化およびクライアントからのリクエストレート変化（Requested）を示す。図 6 より提案手法では要求リクエストレートに応じて処理可能なリクエストレートを変化させていることが分かる。しかし、リクエストレートが増加するとき、要求リクエストレートよりも処理可能なリクエストレートが下回る場合が実サーバ追加中に若干存在しており追従遅延が起きていることが分かる。このとき、提案手法適用時の実際の平均応答速度は約 14.2 msec であり、No_control 手法の約 2.0 msec と比較して増加した。その主な原因は上記追従遅延に加え、QoS 満足下でもアクティブな実サーバ台

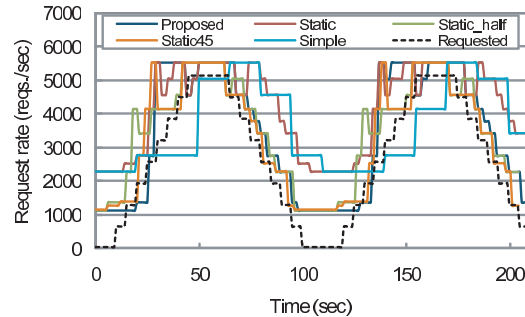


図 6 処理可能な最大リクエストレートの変化 (128 reqs./sec² 時)

Fig. 6 Time variation of the maximum request rate value a system can handle (128 reqs./sec² workload).

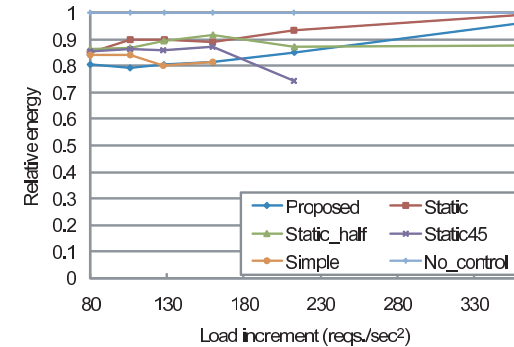


図 7 消費エネルギーの相対比較

Fig. 7 Comparison of relative energy consumption.

数が少なければ 1 台あたりのリクエスト処理量が増加して平均応答時間が延びるためである。上記の現象は同様に QoS 性能が低下した Static45 手法にも見られる。また、Simple 手法では実サーバ状態更新間隔が長いことにより適切な処理可能リクエストレートを保てておらず、大きな追従遅延が発生して QoS 性能の大幅な低下を招いたと考えられる。一方、Static 手法および Static_half 手法では実サーバ追加時のマージン考慮が成功しており追従遅延も見られなかった。

5.3.3 消費電力およびエネルギー評価

図 7 に 6 手法 (Proposed, Static, Static_half, Static45, Simple および No_control) 適用時における 6 種類のワークロードを実行したときの消費エネルギーを示す。なお、この図は No_control 手法の値を 1 としたときの相対値を表している。図 7 より提案手法では増加量が 160 reqs./sec² までの場合には No_control 手法と比較して約 20% のエネルギー削減を達成していることが分かる。このエネルギー削減率は他の比較手法よりも高い値である。つまり、提案手法は QoS が満足される範囲内では効果的なエネルギー削減がワークロードによらず達成できていることが分かる。同様な傾向は Static45 手法でも見られるが、エネルギー削減率は提案手法の方が 5% 以上高かった。一方で、Static 手法は 80 reqs./sec² の増加量の場合を除いて、エネルギー削減率は約 10% にとどまっている。同様に Static_half 手法でもエネルギー削減率は最大でも約 14% であった。また、Simple 手法ではすべてのワークロードでエネルギー削減率が 15% を超えているが、前項で述べたように増加量が 80 reqs./sec² のワークロード以外では適切な実サーバ状態 (N, g) を保てなかったことによるものである。

図 8 に 128 reqs./sec² の増加量のワークロード実行時における 6 手法の実サーバ消費電力変化を示す。クライアントからのリクエストレートが低い期間では、提案手法、Static_half 手法および Static45 手法では 1 台の実サーバのみで処理しており消費電力が約 48 W であった。しかし、Static_half 手法ではリクエストレート増加量の考慮が影響し、リクエストレート増加時には提案手法よりも早い時間に実サーバを追加している。同様にリクエストレート減少時には提案手法よりも遅く実サーバを削除している。この現象が主な原因となり、エネルギー削減率に対して約 9% の差が生じた。また、リクエストレートがほぼ 0 reqs./sec の状態から増加していく期間に着目すると、Static 手法と Static_half 手法では実サーバを追加する時間が異なる。これはあらかじめ存在するアクティブ状態の実サーバ台数が異なるためである。このとき Static 手法は最初 2 台の実サーバがアクティブ状態であったにもかかわらず、Static_half 手法が先に実サーバを追加するためにワークロード全体としてのエネルギー削減率の差は約 1% 程度という結果が得られた。

5.3.4 考 察

達成できた QoS 性能という観点に着目すると、提案手法を適用した場合ではワークロード中のリクエストレート増加量がある値を超えたときに、QoS 性能が著しく低下した。この現象は Static45 手法適用時にも見られ、その主要因はマージンが決定するアクティブな最少の実サーバ台数にある。提案手法および Static45 手法ではリクエストレートが 0 reqs./sec の状態から増加し始めるときにアクティブな実サーバは 1 台しか存在せず、処理可能なリクエストレートを超えた場合にスタンバイな実サーバが復帰するまで 1 台の実サーバで処理

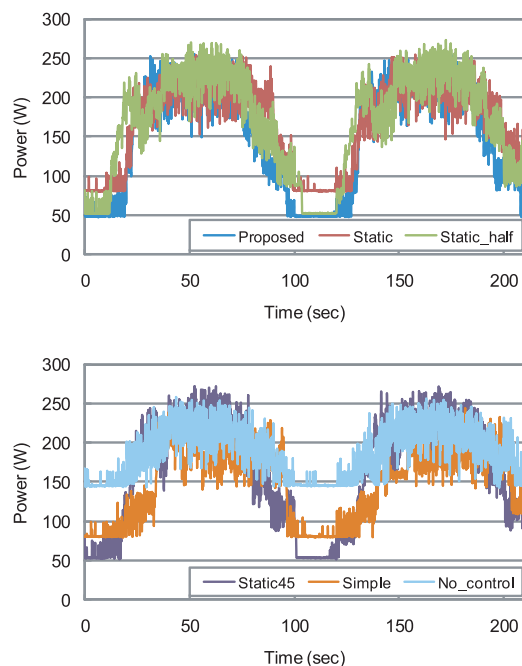


図 8 消費電力の変化 (128 reqs./sec² 時)
Fig. 8 Time variation of the power consumption (128 reqs./sec² workload).

しなければならなかった。一方, Static および Static_half 手法ではアクティブな実サーバが 2 台存在していた。つまり処理可能なリクエストレートが元々高いことに加え, 処理可能なリクエストレートを超えた場合にその超過リクエストレートは 2 台の実サーバに分散されることになる。したがって, 提案手法の追従性能を向上させるためには, その時点でのアクティブな実サーバ台数を制約式 (11) におけるマージン部分の一変数として考慮していく必要があると考えられる。

エネルギー削減率について着目すると, 提案手法を適用した場合では QoS 性能が高いワークロードにおいては他手法よりも削減率が高くなった。このとき Static および Static_half 手法ではワークロードごとにつねにアクティブであるべき実サーバ台数にばらつきが生じ, 結果としてエネルギー削減率のばらつきが生じた。提案手法ではマージンを時系列に変化させることでワークロードによらず安定したエネルギー削減率を達成できたといえる。

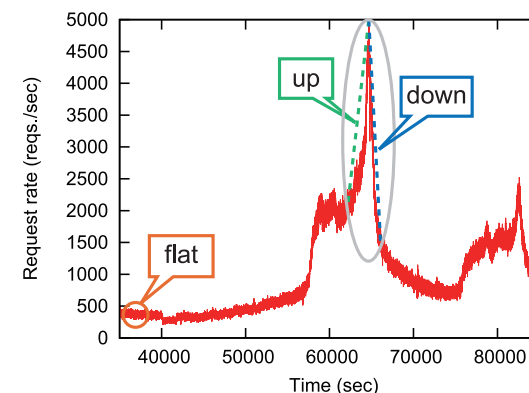


図 9 1998 年ワールドカップ Web サイト運用 61 日目におけるリクエストレート変化
Fig. 9 Time variation of request rate at the 1998 World Cup web site.

5.4 Realistic ワークロードによる評価

Realistic ワークロードを用いた評価は, 実際の運用時を想定した提案手法の追従性能とエネルギー削減率について調べるためのものである。本評価では 1998 年ワールドカップ Web サイトにおけるアクセス履歴²¹⁾のうち, サーバ運用中で 1 日の総リクエスト数が比較的高かった 61 日目のデータを基にワークロードを作成した。具体的には, 得られたデータより 1 秒ごとのリクエストレートを算出し, 評価対象のサーバシステムが処理可能な最大リクエストレートに対してスケールを合わせている (本評価システムではそれぞれの時点のリクエストレート値に対して 1.3 倍したもの)。図 9 に運用開始 61 日目のリクエストレート変化 ($t = 35000 - 86000$ sec) を示す。文献 20), 24) で述べられているように, Web サーバに対するリクエストの特徴として図 9 で確認できる山型のリクエストパターンがたびたび訪れることがあげられる。そこで, 本評価ではこのデータから 3 カ所の期間 (up, down, flat とする) に着目して 3 つのワークロードを作成した。up ワークロードはサーバ負荷平均値の上昇期間, 同様に down ワークロードはサーバ負荷平均値の減少期間, flat ワークロードはサーバ負荷平均値が平坦な期間を意味する。表 5 に各ワークロードの概要を示す。ここで, “Time” はワークロードの実行時間 (単位 sec), “Total reqs.” はワークロード全体での総リクエスト数を表す。また, “ave. R'_+ ”, “ave. R'_- ” はワークロード中でのリクエストレートのそれぞれ正の傾き R'_+ および負の傾き R'_- の平均値 (単位 reqs./sec²) を表す。

表 5 3 種ワークロードの概要
Table 5 Properties of three workloads.

Workload	Time	Total reqs.	ave. R'_+	ave. R'_-
up	2284	6339680	90	87
down	1308	3498864	81	86
flat	300	116704	33	31

5.4.1 比較対象

実運用時において先行研究手法^{6)–8)}を適用する場合、特徴情報を得るために想定するワークロードの定義は手法適用者に委ねられる。このとき、その定義の仕方がマージンの大きさを決定する。さらに実運用において図 9 のようなワークロード全体を処理するとき、本提案手法とは異なり先行研究手法では一意に定められた時不変のマージンを適用したまま実際の特徴が異なる *up*, *down* や *flat* のようなワークロードを処理することになる。この評価ではそのような条件下での各ワークロードに対する QoS 性能とエネルギー削減率を調べるため、以下に述べる Static42 および Static90 手法を比較対象とした。Static42 手法はワールドカップ Web サイト運用 61 日目におけるリクエストレート²⁾の正の傾き R'_+ の平均値 (42 reqs./sec²)、Static90 手法はより QoS 性能に重点をおくために *up* ワークロード中における R'_+ の平均値 (90 reqs./sec²) を $\tilde{R}'_+(s+1)$ としている (実サーバ追加/削除の制御は式 (10), (11) と同様であり、 T_p の値も提案手法と同様に 3 sec とした)。

また、Synthetic ワークロード評価時と同様に Simple 手法を用意し、疎粒度で単純な実サーバ制御を行ったときの 3 種ワークロードに対する QoS 性能およびエネルギー削減率についても評価を行った (パラメータ等は Synthetic ワークロード評価時と同様である)。

5.4.2 QoS 性能評価

図 10 に 5 手法 (Proposed (提案手法), 比較対象 3 種, No_control) での 3 種ワークロード実行時の QoS 性能を示す。ここで、No_control 手法は Synthetic ワークロード評価時と同様であり、No_control 手法の QoS 性能を 1 として正規化してある。図 10 より Static42 手法を除く手法ではどのワークロードに対しても良好な QoS 性能を得ていることが分かる。一方で、Static42 手法は *down* ワークロード実行時に QoS 性能が約 40%低下した。この主な原因は急激なリクエストレートの減少と増加が連続して起こったとき、実際には増加後に必要な実サーバ 1 台をリクエストレート減少の検知により削除したためである。このとき他手法ではマージンが大きく、実サーバ削除を行わなかった。

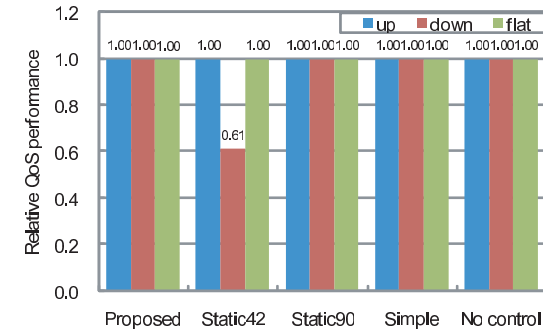


図 10 相対 QoS 性能 (Realistic ワークロード)

Fig. 10 Relative QoS performance (Realistic workloads).

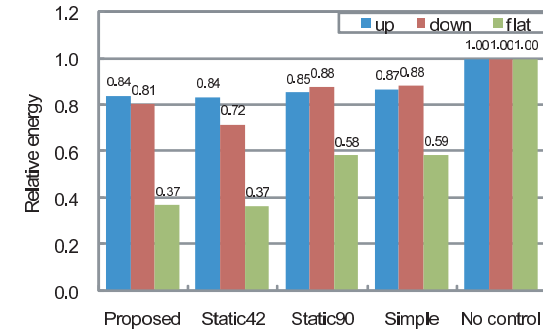


図 11 相対エネルギー (Realistic ワークロード)

Fig. 11 Relative energy consumption (Realistic workloads).

5.4.3 消費電力およびエネルギー評価

図 11 に QoS 性能比較と同様 5 手法での 3 種ワークロード実行時の消費エネルギーを示す。ここで、エネルギー値は No_control 手法のエネルギー値を 1 として正規化してある。図 11 より提案手法は他手法と比較して同等もしくはさらに大きくエネルギーを削減できていることが分かる。Static42 手法との比較では *up* および *flat* ワークロードにおいて同等のエネルギー削減率となった (*down* ワークロードは Static42 手法において大幅な QoS 性能低下が見られたため正当な比較はできない)。Static90 および Simple 手法との比較では、*down* および *flat* ワークロードにおいてエネルギー削減率の差は最低でもそれぞれ約 7 (= 88 - 81) %, 約 22 (= 58 - 37) %となった。しかし、*up* ワークロードではエ

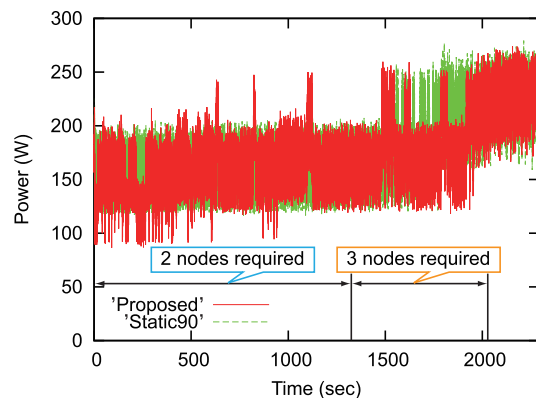


図 12 up ワークロード実行時の消費電力変化

Fig. 12 Time variation of power consumption in up workload.

エネルギー削減率の差が最高でも 3 ($= 87 - 84$) %にとどまった。図 12 に up ワークロード実行時での提案手法および Static90 手法の消費電力変化を示す。up ワークロードでは時刻 $t = 1350 \text{ sec}$ 程度までは実サーバが 2 台必要であり、その時点から時刻 $t = 2040 \text{ sec}$ 程度まで実サーバを 3 台必要とする。時刻 $t = 1350 \text{ sec}$ までの期間に提案手法では実サーバを 2 台のみアクティブ状態（消費電力が約 85 W）へ移行して消費電力の削減を行っているが、一方で実サーバを 4 台アクティブ状態にしている箇所（消費電力が約 250 W）が見られる。その結果、ワークロード全体におけるエネルギー削減率に大きな差として現れなかったと考えられる。

5.4.4 考 察

マージンを時不変定数として設定したとき、定数値の与え方が影響して QoS 性能とエネルギー削減率は個々のワークロードによって差が生じた。つまり先行研究手法を実運用時に適用して全体として良好な結果を得るためには、マージン設定に用いる想定ワークロードを適宜再定義していく必要があることを示している。一方、本提案手法では統計的予測手法を適用してマージンの再定義を行い、QoS 性能維持と効果的なエネルギー削減をワークロードの種類によらず達成できたといえる。また、用意したワークロードの特性により本提案手法と Simple 手法ではエネルギー削減率の差が見られないことがあった。しかし実運用上では前節での評価のような急激なリクエストレート増加に対する QoS 性能を考慮する必要がある、この点を含めればより細粒度な制御を行う本提案手法はより有用性が高いといえる。

本提案手法における改善すべき点は無駄な実サーバの追加を抑制することである。この問題を解決するためには、「急激なリクエストレート増加傾向がいつまで続くか」を推定する必要がある。推定値が正しい場合、ある時点において長期的に増加が見込まれると判断すればすぐに実サーバを追加すればよい。このような推定を行うためには本提案手法で用いている短期的なリクエストレートの増加量に加えて長期的なリクエストレートの増加量を考慮する必要があると考えられる。

6. 実運用における提案手法の適用

Web サーバの構成はその目的等に応じて様々であり、提案手法を Web サーバの実運用において適用する場合何らかの問題が生じるおそれがある。

たとえば、Web サーバでは Cookie を用いてのセッション情報維持が行われることがある。通常、Cookie は発行サーバに依存して処理される設定となっているため、分散型 Web サーバでは実サーバをスタンバイ状態へ移行させることでセッション情報不整合等の問題が発生する。このような場合、既存の分散型 Web サーバで行われているように、すべての実サーバからアクセスできるセッション情報保持サーバを用意し、Cookie の domain パラメータを仮想サーバのアドレスに指定する。これによりセッション情報を維持しながらリクエストが特定の実サーバに依存しない負荷分散が実現できるため、提案手法を適用して実サーバをスタンバイ状態へ移行させて運用が可能となる。

また、SSL の処理においても同様にアクセスしたサーバに依存する処理が存在する。解決策の 1 つとして、Stunnel²⁵⁾ のような SSL ラッパを仮想サーバ上で導入すれば SSL の通信はクライアントと仮想サーバ間のみで行われることになる。これにより仮想サーバと実サーバ間では通常の通信で行われることから提案手法を適用して運用することが可能であると考えられる。

さらに、Web サーバの実運用ではリクエスト URL や Cookie の内容に応じた Layer-7 における負荷分散手法も存在する。このような場合にリクエストと処理する実サーバに依存関係が存在するため提案手法を単純に適用することはできない。そのため、省電力化を行うためには新たな方策を講じる必要があると考えられる。

7. おわりに

本研究では、複数から構成される分散型 Web サーバにおいて QoS 性能を維持しながらエネルギー削減を達成することを目的として、分散型 Web サーバにおける消費電力特性の

評価および仮想サーバと実サーバ群が協調してサーバ電力状態を制御する省電力化手法の提案を行った。

分散型 Web サーバにおける消費電力特性の評価では、評価システムにおいてプロセッサ Gear の制御よりもスタンバイ状態を利用することで、低リクエストレート時に QoS を維持しながら効果的な消費電力の削減を行えることが分かった。

そして、分散型 Web サーバ上で QoS 維持とエネルギー削減を行うための消費電力プロファイルを利用する CPMD 機構について提案および実装を行った。従来手法とは異なり、本提案手法では想定するリクエストパターンに関する前提知識に頼ることなく実際のリクエストパターンからその特徴を局所的にとらえ、リクエストに追従しながら分散型 Web サーバのノード状態を制御するようにアルゴリズムを設計した。評価実験では、提案手法は時系列で変化するリクエストレートに追従しながら十分な QoS 性能維持と効果的なエネルギー削減をワークロードの種類によらず達成できることが分かった。

今後の課題としては、追従遅延の向上および不必要な実サーバ追加制御の低減手法についての具体的な考案および実装を行う予定である。また、大規模な実サーバ環境における本提案手法の有効性についても確認を行う。

謝辞 本研究の一部は科学技術振興事業団・戦略的創造研究推進事業 (CREST) の研究プロジェクト「省電力でディペンダブルな組込み並列システム向け計算プラットフォーム」による。

参 考 文 献

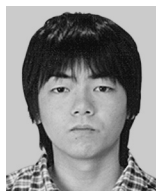
- 1) Elnozahy, M., et al.: Energy Conservation Policies for Web Servers, *Proc. 4th USENIX Symposium on Internet Technologies and Systems (USITS)* (2004).
- 2) Kotla, R., et al.: Scheduling processor voltage and frequency in server and cluster systems, *Proc. 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)* (2005).
- 3) Bertini, L., Leite, J. and Mossé, D.: Coordinated DVS for QoS Control in Energy-efficient Web Clusters, *9th Workshop on Real-Time Systems (WTR 2007)* (2007).
- 4) Sharma, V., et al.: Power-aware QoS Management in Web Servers, *Proc. 24th Real-Time Systems Symposium (TSS 2003)* (2003).
- 5) Elnozahy, M., et al.: Energy-Efficient Server Clusters, *Proc. 2nd Workshop on Power-Aware Computing Systems (PACS)* (2002).
- 6) Rajamani, K. and Lefurgy, C.: On Evaluating Request-Distribution Schemes for Saving Energy in Server Clusters, *Proc. 2003 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)* (2003).
- 7) Rusu, C., et al.: Energy-Efficient Real-Time Heterogeneous Server Clusters, *Proc. 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06)* (2006).
- 8) Bertini, L. and Mossé, D.: Statistical QoS Guarantee and Energy-Efficiency in Web Server Clusters, *Proc. 19th Euromicro Conference on Real-Time Systems* (2007).
- 9) Xu, R., et al.: Energy-Efficient Policies for Embedded Clusters, *Proc. 2005 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems (LCTES'05)* (2005).
- 10) Chen, Y., et al.: Managing Server Energy and Operational Costs in Hosting Centers, *ACM SIGMETRICS Performance Evaluation Review* (2005).
- 11) Intel Corporation: Intel Core2 Extreme Processor X6800 and Intel Core2 Duo Desktop Processor E6000 and E4000 Sequences Datasheet. <http://www.intel.com/design/core2duo/documentation.htm>
- 12) Standard Performance Evaluation Corporation: SPECWeb99. <http://www.spec.org/web99/>
- 13) Bestavros, A., et al.: Characterizing Reference Locality in the WWW, *Proc. IEEE Conference on Parallel and Distributed Information Systems (PDIS)* (1996).
- 14) Fast CGI: FastCGI Specification. <http://www.fastcgi.com/>
- 15) Hotta, Y., et al.: Profile-based optimization of power performance by using dynamic voltage scaling on a PC cluster, *Proc. High Performance Power-Aware Computing Workshop (HPPAC) in the 20th IEEE IPDPS'06* (2006).
- 16) Barroso, L.A. and Höltle, U.: The Case for Energy-Proportional Computing, *IEEE Computer* (2007).
- 17) Weigend, A.S. and Gershenfeld, N.A.: *Time Series Prediction: Forecasting the Future and Understanding the Past*, Addison-Wesley (1993).
- 18) The Linux Virtual Server project: Linux Virtual Server. <http://www.linuxvirtualserver.org/>
- 19) Mosberger, D. and Jin, T.: httpperf — A tool for measuring web server performance, *Proc. 1998 Internet Server Performance Workshop* (1998).
- 20) Arlitt, M. and Jin, T.: A Workload Characterization Study of the 1998 World Cup Web Site, *Journal of IEEE Network*, Vol.14 (2000).
- 21) Danzig, P.B., et al.: The Internet Traffic Archive — WorldCup98. <http://ita.ee.lbl.gov/html/contrib/WorldCup.html>
- 22) Comprehensive Perl Archive Network: CPAN — Compress-Zlib. <http://search.cpan.org/~pmqs/Compress-Zlib/>
- 23) Nielsen, H.F.: The Effect of HTML Compression on a LAN. <http://www.w3.org/Protocols/HTTP/Performance/Compression/LAN.html>

24) Iyengar, A.K., et al.: Analysis and Characterization of Large-Scale Web Server Access Patterns and Performance, *Journal of World Wide Web*, Vol.2 (1999).

25) Trojnara, M. and Hatch, B.: Stunnel — Universal SSL Wrapper.
<http://www.stunnel.org/>

(平成 20 年 10 月 3 日受付)

(平成 21 年 1 月 22 日採録)



今田 貴之 (学生会員)

昭和 58 年生。平成 18 年青山学院大学理工学部情報テクノロジー学科卒業。現在、筑波大学大学院システム情報工学研究科博士後期課程在学中。データセンタおよび高性能計算向け省電力クラスタシステムに関する研究に従事。



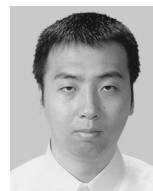
佐藤 三久 (正会員)

昭和 34 年生。昭和 57 年東京大学理学部情報科学科卒業。昭和 61 年同大学院理学系研究科博士課程中退。同年新技術事業団後藤磁束量子情報プロジェクトに参加。平成 3 年通産省電子技術総合研究所入所。平成 8 年新情報処理開発機構並列分散システムパフォーマンス研究室室長。平成 13 年より筑波大学システム情報工学研究科教授。平成 19 年より同大学計算科学研究センターセンター長。理学博士。並列処理アーキテクチャ、言語およびコンパイラ、計算機性能評価技術、グリッドコンピューティング等の研究に従事。IEEE、日本応用数理学会各会員。



木村 英明 (学生会員)

昭和 58 年生。平成 16 年小山工業高等専門学校電子制御工学科卒業。平成 18 年筑波大学第三学群情報学類卒業。平成 20 年日本学術振興会特別研究員。現在、筑波大学大学院システム情報工学研究科在学中。クラスタシステムの省電力化に関する研究に従事。



堀田 義彦 (正会員)

昭和 54 年生。平成 15 年筑波大学第三学群情報学類卒業。平成 20 年同大学院システム情報工学研究科コンピュータサイエンス専攻博士課程後期修了。博士(工学)。現在、(株)ルネサステクノロジにてカーナビ向け SoC 開発に従事。