WWWサーバ管理支援システムの開発

Develope WWW Server Management Suport System

中村 豊 [1)]
Yutaka NAKAMURA
yutaka-n@rd.center.osaka-u.ac.jp

中山 貴夫 [2)]
Takao NAKAMURA
takao-na@is.aist-nara.ac.jp

藤井 邦浩 [2)]
Kunihiro FUJII
kunihi-f@is.aist-nara.ac.jp

森田 直樹 [2)]
Naoki MORITA
naoki-mo@is.aist-nara.ac.jp

持田 啓 [2)]
Kei MOCHIDA
kei-mo@is.aist-nara.ac.jp

1) 大阪大学大学院基礎工学研究科 リサーチ・アソシエイト (〒 567-0047 大阪府茨木市美穂ヶ丘 5-1 大阪大学サイバーメディアセンター)

2) 奈良先端科学技術大学院大学情報科学研究科 (〒 630-0101 奈良県生駒市高山町 8916-5 情報科学センター)

ABSTRACT. The World Wide Web (WWW) has been emerged as a vital service in the Internet. The quality of WWW service is now a first priority. The performance measurements through the benchmark or any mechanisms installed inside the server itself are not practical for applying them to "running" WWW servers. WWW service consists of WWW server system input/output packets. Therefore, we proposed the WWW server performance measurement method by packet monitoring. In this paper, we discuss the design and implementation of the Enhanced Network Measurement Agent(ENMA), and show an example in which we applied this system to actual WWW servers operated in the Internet.

1 背景

インターネットは社会基盤の一つとして定着しつつあるほど普及している．インターネットで様々なサービスが提供される事により，サービスを行うサーバは大規模化している．この様なサーバシステムの大規模化は今後も続くと予想される．

現在，この様な大規模サーバシステムの管理・運営には多くの人材が投入されている．なぜなら，サーバ管理者は顧客に対して「サーバシステムの信頼性の保証」を行う義務があるからである．多くの管理者は「サーバの不具合の検知」や「障害発生時の対応」に追われている．一方，負荷分散のためにサーバシステムのネットワーク的な分散化が進んでいるため，管理者は分散しているサーバシステムを集中的に管理する必要に迫られている．

従来，多くの管理者は上で述べた障害に対して，勘と経験を頼りに管理業務を行ってきた．なぜなら，サーバの性能や管理の指標となるものが，これまでは存在しなかったからである．ベンチマークシステムを用いる事で，サーバの定量的評価は可能である．しかし，運用されているシステムとベンチマークでの結果は大きく事なる．

このことから，運用中のサーバシステムに対して「サーバの不具合の検知」や「障害発生の理由」を明確にできるシステムが望まれる．さらに，広域に分散したサーバシステムの情報をリアルタイムに集中管理できるシステムが望まれる．そこで我々は，「サーバの不具合の検知」を容易に行い，「障害発生時の対応」に何らかのヒントを与える事ができ，広域に分散したサーバ群を集中管理できるシステムを開発する．本システムは，実環境で分散サーバシステムの稼働状況を把握し，サーバ管理者にその状況を伝えるシステムである．本システムを用いる事で，サーバ管理者の負荷を劇的に減らす事が可能であると考えられる．

2 目的

本開発の目的は，広域に分散したサーバ群を集中管理し，稼働状況を把握することで，サーバ管理者の管理に対する負荷を軽減するためのツールを開発することにある．

3 WWWサーバにおける性能指標

本節では，WWW サーバシステムの性能を代表するいくつかの性能指標について述べる．

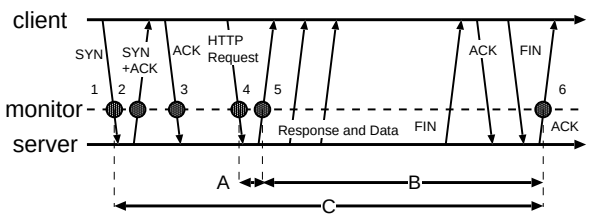

図 1: 性能指標

3.1 1本のコネクションに関する性能指標

- コネクション継続時間 (T_c)

  コネクション継続時間は，TCP コネクション確立から終了までの時間である．すなわち，クライアントによってサーバへ送信された最初の SYN パケット (図 ?? の 1) を観測してから，最後の ACK パケット (図 1 の 6) を観測するまでの時間である (図 1 の区間 C).

- 応答時間 (T_r)

  応答時間の定義は，クライアントからの HTTP 要求パケットを観測してから，サーバの最初の HTTP 応答パケットを観測するまでの時間である．我々が定義した応答時間は図 1 の区間 A である．

- データ転送時間 (T_d)

  データ転送時間は，最初のデータパケットをサーバが観測してから，最後の HTTP データパケットを観測するまでの時間と定義する (図 1 の区間 B)．データ転送時間に FIN パケットとその ACK を含んでいるのは，データが含まれていることが多いからである．

3.2 サーバシステム全体の性能指標

- コネクション処理率

  コネクション処理率はサーバで単位時間あたりに処理したコネクション数である．

- コネクション到着率

  コネクション到着率はクライアントからの単位時間あたりのコネクション到着数である．

- 現在処理中のコネクション数

  現在処理中のコネクション数はサーバでの同時に処理している TCP コネクションを意味している．

コネクション処理率とコネクション到着率を比較する事で，サーバが十分に WWW サービスを提供しているかどうかが明らかとなる．

3.3 サーバ内部の性能指標

サーバの内部パラメータに関する性能指標を以下に示す．

- 未使用メモリ

  サーバシステム内部の未使用メモリ量

- 使用メモリ

  サーバ内部で使用されているメモリ量．カーネルが使用しているメモリ量は含まれない．

- プロセス数

  サーバシステム内部で起動しているプロセスの総数

- コネクション到着数

  サーバに到着したコネクション数．

- タイムスタンプ

  計測した時刻．秒単位で計測する．

マルチプロセッサシステムの場合は CPU 毎の以下の項目についてモニタリングする．

- アイドル時間率

  CPU が休んでいる割合．この時間が多ければ，CPU にはまだ余力があると判断できる．

- カーネル時間率

  カーネル内部が処理している割合．この時間が多くなると CPU の余力は少ないと判断できる．

- ユーザ時間率

  CPU がユーザプロセスを処理している割合．

- コンテキストスイッチ数

  CPU で 1 秒間にコンテキストスイッチが発生している回数．コンテキストスイッチとはプロセス切替のことであるので，この回数が多いとサーバの負荷が高いと言える．

- システムコール数

  CPU で 1 秒間にシステムコールが発行されている回数．システムコールが発行されるとユーザ空間からカーネル空間への切り替わりが発生する．したがって，この回数が多いとサーバの負荷が高いと言える．

- 割り込み数

  CPU に対して 1 秒間に発生している割り込みの回数．割り込みが発生するとシステム内部の割り込み処理ルーチンに処理が移る．したがって，この回数が多いとサーバの負荷が高いと言える．

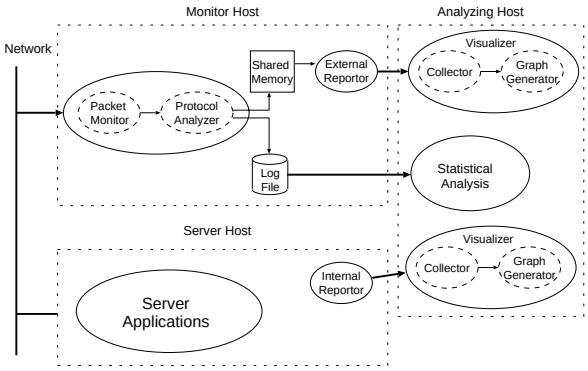

図 2: 全体のモジュール図

- ページフォルト数

  CPU に対して 1 秒間に発生しているページフォルトの回数．ページフォルトが発生するとシステムはメモリ確保や解放のためのルーチンに処理が移る．したがって，この回数が多いとサーバの負荷が高いと言える．

4 設計

本章ではこれまで述べてきた性能指標を計測するためのモジュールについて説明する．図 2 にシステム全体の構成を示す．

- パケットモニタモジュール

  サーバシステムに対して発生する通信のトラヒックに関して，リアルタイムにパケットモニタリングを行う．モニタしたパケットはプロトコル解析モジュールへ渡される．

- カーネルモニタモジュール

  サーバ内部の状態を抽出するためのモジュール．内部状態抽出モジュールからの要求を受けとると，CPU 利用率やコンテキストスイッチ数などを外部ホストへ転送する．

- プロトコル解析モジュール

  コネクション解析モジュールは以下の機能を提供する．

  - HTTP コネクション上のパケットの到着時間を記録する．
  - 各々の TCP コネクションで観測された IP データグラムのペイロード長を記録する．また，IP データグラムのペイロードの総数を TCP コネクションでの上りと下りを区別して記録する．
  - TCP ペイロードについてもコネクションの上りと下りを区別して，ペイロード長を記録する．
  - 1 つのコネクション情報を 1 つのエントリとしてログファイルに出力する．
  - リアルタイムにデータの視覚化と解析を行うために，性能解析プログラム群へ共有メモリを介してデータを提供する．

- 外部情報抽出モジュール

  プロトコル解析モジュールが共有メモリけ出力したリアルタイムデータを外部の解析ホストへ転送する．

- 内部状態抽出モジュール

  より詳細なサーバの状態把握のために，サーバシステム内部の情報を解析ホストへ転送する．

- 視覚化モジュール

  サーバを外部から観測した結果および，内部情報の抽出した結果をリアルタイムにデータを視覚化する．

- 統計解析モジュール

  コネクション時間や同時コネクション数の分布といった，全体を読み込まないと解析できないデータの算出のために，ENMA デーモンの性能解析プログラムを用意した．これらのプログラムは ENMA デーモンのログファイルを読み込み，データの解析をバッチ形式で行う．

5 実装

我々は，様々な UNIX プラットフォームで動作するシステムを目標として本システムを実装した．ENMA デーモンは C 言語，性能解析プログラム群は AWK，C 言語で実装した．開発および実行環境は FreeBSD，Solaris や IRIX である．この章では ENMA の実装について説明する．

- パケットモニタモジュール

  パケットモニタモジュールでは，ネットワーク上のパケットを監視するために，Lawrence Berkeley National Laboratory(LBNL) のパケットキャプチャライブラリ (libpcap [1]) を用いている．このライブラリは BSD 上の Berkeley Packet Filter(BPF [2]) や System V での Network Monitoring Protocol(the packet snooper) の機能を利用し，統一された API を提供している．このライブラリを用いることによって，IP 層でのパケットが得られる．また，多くの UNIX プラットフォームで ENMA を動作させることが可能となる．

- カーネルモニタモジュール

  カーネル内部の情報は OS 毎に異なる．可能な限り OS を避ける実装にするために，rep2 ではカーネル情報を抽出するために kvm イ

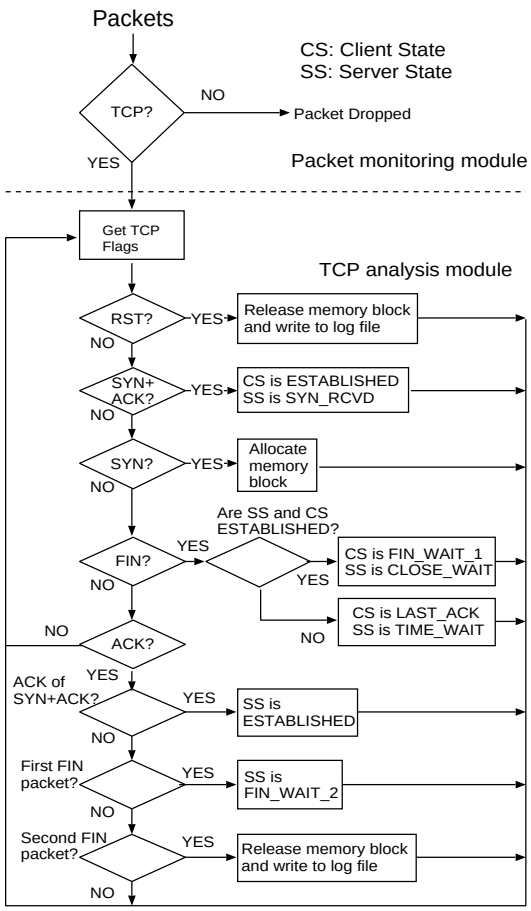

図 3: コネクション解析アルゴリズム

ンタフェースを用いた．kvm インタフェースを用いることで，Solaris 2.6 および FreeBSD 4.0 において統一した情報を得ることが可能となった．

しかし，kvm インタフェースだけでは十分にカーネル情報を引き出すことが出来ない．そこで，OS の実装によって以下に示す異なるインタフェースを用いて，カーネル情報を抽出した．

- Solaris の場合

  kstat インタフェースを用いたデータ出力．kstat インタフェースを用いて，ネットワーク関係の情報を得ることが可能．

- FreeBSD の場合

  sysctl インタフェースを用いたデータ出力．上と同様，sysctl インタフェースを用いて，ネットワーク関係の情報を得ることが可能．

- プロトコル解析モジュール

  コネクションアナライザでは，各々の TCP コネクションの TCP 状態遷移の監視を行う．このモジュールではシーケンス番号，確認応答番号，TCP ヘッダのフラグを用いてサーバとクライアントの状態を追跡する．コネクションアナライザの用いるアルゴリズムを図 3 に示す．

  図 3 に示すように，パケットが到着すると，最初に TCP ヘッダのフラグを見る．フラグの種類によって以下のような処理を行う．

  1) RST の場合，コネクションが終了すると判断するので，メモリブロックを解放し，ログファイルにコネクション情報を出力する．
  2) SYN+ACK の場合，サーバの状態変数を SYN_RCVD に遷移させ，クライアントの状態変数を ESTABLISHED に遷移させる．
  3) SYN の場合，メモリブロックを割り当て，状態の初期化を行う．
  4) FIN の場合は，サーバの状態とクライアントの状態によって，状態を遷移させる．
     (a) サーバとクライアントの状態が ESTABLISHED の場合，クライアントの状態を FIN_WAIT_1 へ遷移させ，サーバの状態を CLOSE_WAIT へ遷移させる．
     (b) サーバとクライアントの状態が ESTABLISHED でない場合は，クライアントの状態を LAST_ACK へ遷移させ，サーバの状態を TIME_WAIT へ遷移させる．
  5) ACK の場合
     (a) SYN+ACK に対する ACK であるなら，サーバの状態を ESTABLISHED へ遷移させる．
     (b) 始めの FIN パケットに対する ACK であるなら，サーバの状態を FIN_WAIT_2 へ遷移させる．
     (c) 2 つめの FIN パケットに対する ACK であるなら，コネクションが終了すると判断して，メモリブロックを解放しデータをログファイルへ出力する．

  このアルゴリズムでは，コネクションアナライザは SYN パケットを観測した時に，各々の TCP コネクションの状態を記録するためにメモリブロックを割り当てる．TCP コネクションが正常に終了する場合，メモリブロックは解放されデータファイルへコネクションの情報が出力される．正常に終了しなかった場合，メモリブロックが解放されない．そこで，コネクションアナライザは，ある一定間隔でメモリブロックを検査し，古いメモリブロックを解放する．

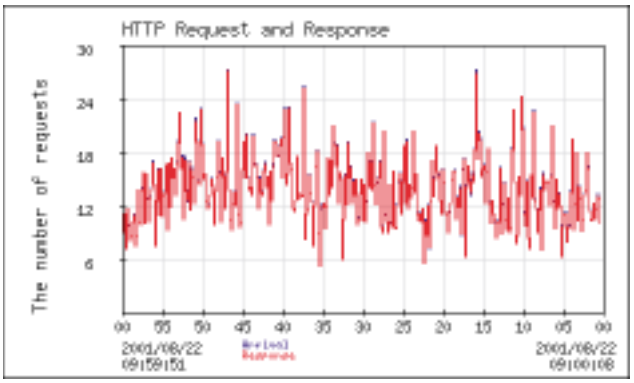

図 4: HTTP リクエストの経時変化 (視覚化モジュールの例)

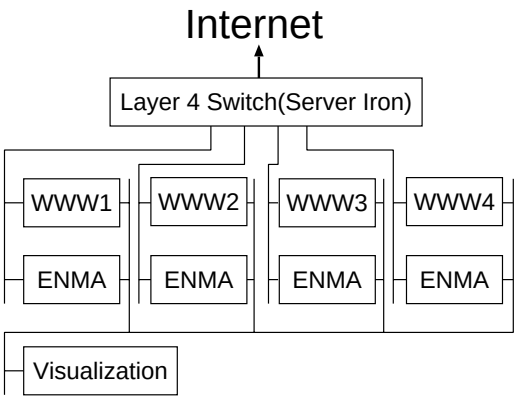

図 6: ネットワーク構成図

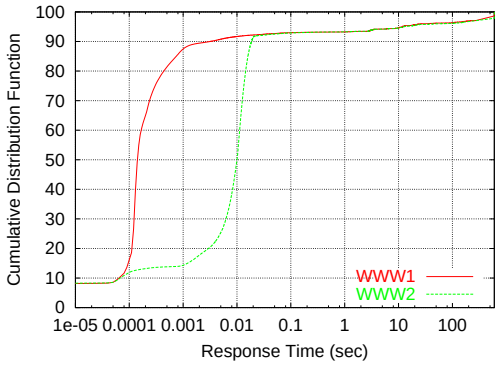

図 5: 応答時間の例

- 状態抽出モジュール

  外部状態抽出モジュールおよび内部状態抽出モジュールは 10 秒間隔で ENMA (プロトコル解析モジュール) および rep2(カーネルモニタモジュール) に対してデータを要求する．得られたデータはテキストファイル形式として出力され，視覚化モジュールによってグラフを生成する．

- 視覚化モジュール

  視覚化モジュールは様々なデータを表示するためのプログラムである．データは状態抽出モジュールによって得られる．図 4 のような性能指標に関するグラフをリアルタイムに生成する．

- 統計解析モジュール

  統計解析モジュールは，コネクション解析モジュールが出力したログファイルの統計解析を行い，コネクション継続時間や応答時間などの統計グラフを出力する．図 5 に応答時間の累積頻度分布の例を示す．

6 適用例

ENMA および rep2 の有効性を確認するために，実際に運用されている WWW サーバに対して適用した．第 82 回全国高校野球選手権大会のインターネット中継に用いられたサーバを観測した．このサーバでは 1 日単位で最大約 4600 万アクセスが得られた．

6.1 システム構成

図 6 にネットワーク構成図を示す．図 6 に示すように，4 台のサーバを用意した．上流には Foundary 製 Layer 4 Switch を用いラウンドロビン方式で負荷を分散した．

WWW サーバや観測に用いた計算機のスペックおよびアプリケーションを表 ?? に示す．表 1 に示すように，WWW1 は apache を用い，WWW2 〜WWW4 では，chamomile [3] を用いた．

chamomile では，シングルプロセス，マルチスレッドアーキテクチャが採用されている．これは主にマルチスレッドのコンテキストスイッチによるオーバヘッドを軽減するためである．

また，図 6 に示すように，ENMA を用いて外部より観測した．観測された全てのデータは制御線を用いて視覚化ホストへ転送され，リアルタイムにデータの視覚化が行われた．

6.2 結果および考察

本節では，大会期間中最もアクセスの集中した時間帯 (8/21 の 14:45〜15:45) の解析結果について述べる．

6.2.1 ENMA による観測結果

図 7 に示すように，4 台合計のセッション数は約 6 万本である．コネクション到着数は 4 台合計

表 1: システム構成

	WWW server 1	WWW server 2-4	ENMA host
OS	Solaris 2.7	Solaris 2.7	FreeBSD 4.0
CPU	Ultra SPARC II 400MHz	Ultra SPARC II 400 MHz	PentiumIII 700MHz
Memory	2 GBytes	2 GBytes	256 MBytes
Application	Apache 1.3.12	chamomile	enma

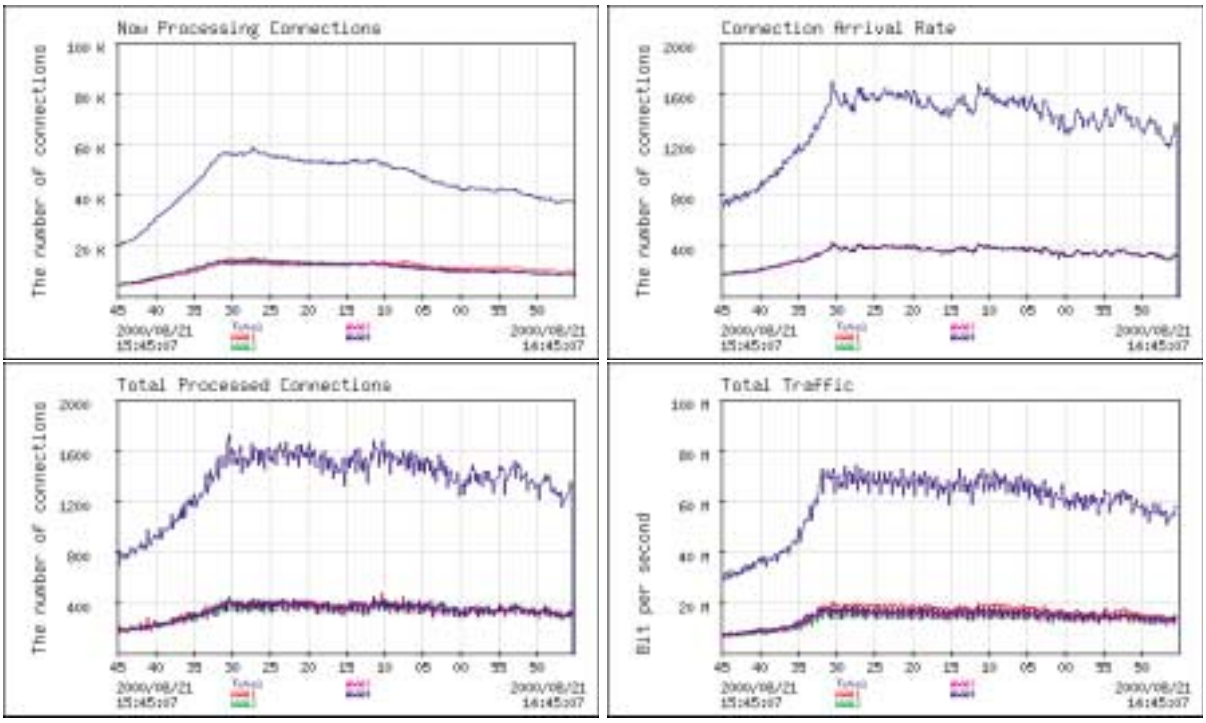

図 7: 4 台合計の結果

で約 1800/秒である．同様に，コネクション処理数も約 1800/秒である．そして，トラヒックは 4 台合計で約 70MBytes/秒である．

コネクション到着数とコネクション処理数の差はほとんど見られないので，サーバシステムは通常状態であると言える．

6.2.2 rep2 による内部観測

- プロセス数

  図 8 に apache と chamomile のプロセス数の遷移の結果を示す．apache はマルチプロセスによる実装のため，プロセス数が上下に揺らいでいる．しかし，chamomile はシングルプロセスな実装のため，プロセス数の揺らぎはない．

  プロセス数が増加すると，OS 内部でのプロセス管理表が増大するためシステムが利用できるメモリ数が減少すると考えられる．

- メモリとページフォルトの関係

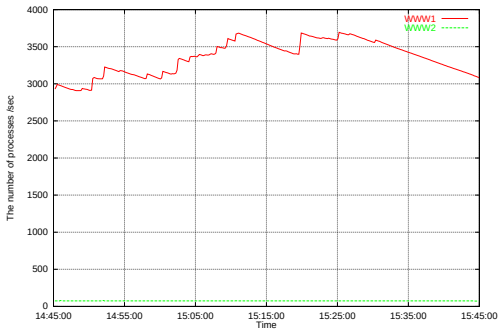

図 8: プロセス数の遷移

図 9 に apache のメモリ量とページフォルト数を示す．図 9 に示すように free memory が減少してくると，ある閾値から突然ページフォルトが発生している．

このページフォルトはシステム内部の再利用

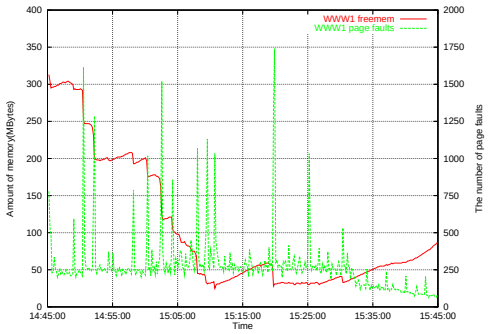

図 9: メモリ量とページフォルト数の遷移

されていないメモリを検索するために発生していると考えられる．そのため，ページフォルトによるメモリの確保とリクエストの到着とのバランスが崩れる時，サーバは飽和状態になると考えられる．

- メモリ量とプロセス数の関係

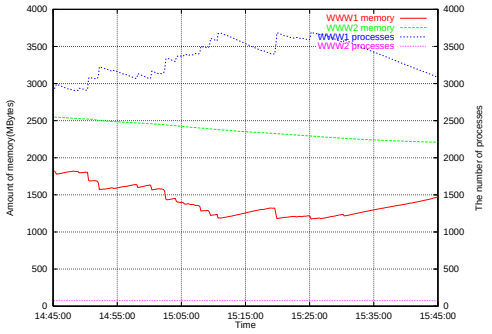

図 10: メモリ量とプロセス数の遷移

図 10 にメモリ量とプロセス数を示す．図 10 に示すように，WWW1 では，プロセス数が増加するとメモリ量が減少している．これは，プロセス数が増えると，OS 内部でのプロセス管理テーブルが大きくなり，ユーザプロセスの使用できるメモリが減少するからであると考えられる．

一方，WWW2 ではプロセス数がほとんど変化していないので，メモリ量もあまり変化していない．

- コンテキストスイッチについて

図 11 に WWW1 および WWW2 のコンテキストスイッチの遷移を示す．図 11 に示すように，WWW1 の方が WWW2 よりも明らかにコンテキストスイッチの数が多い．これは WWW1 の方が多くのプロセスを起動しているからであると考えられる．

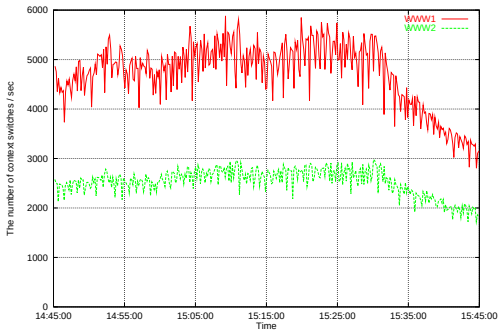

図 11: コンテキストスイッチの遷移

- システムコールについて

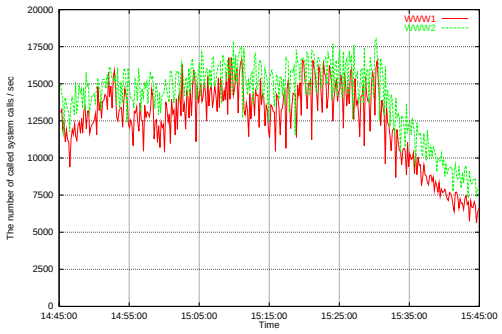

図 12: システムコール数の遷移

図 12 に WWW1 および WWW2 のシステムコール発行数の遷移を示す．図 12 に示すように WWW1 と WWW2 では WWW2 の方が若干システムコールの発行数が多くなっている．apache は select システムコールを用いてセッションの検索を行っている．一方，chamomile は select システムコールではなく poll システムコールを用いてセッション管理を行っている．poll は select とは異なり，検索するセッションの数を指定する事が出来る．chamomile の実装では，poll の検索個数の最大値を制限していたため，セッション検索のために select よりも多くの poll が発行されたと考えられる．このため，chamomile の方が apache よりもシステムコールの発行回数が多くなったと考えられる．

- CPU 利用率について

図 13 に WWW1 および WWW2 の CPU 利用率を示す．図 13 に示すようにカーネル利用率は WWW1 および WWW2 は同様の値で遷移している．しかし，ユーザ利用率は chamomile の方が高く，アイドル率は apache の方が高くなっている．文献 [4] によると，カーネル

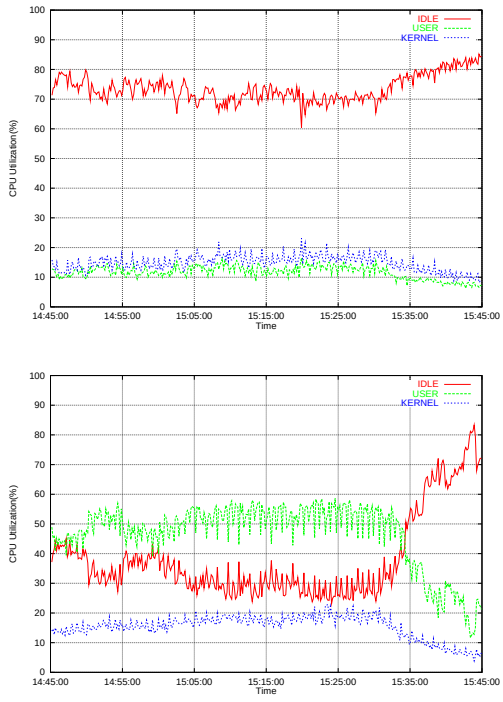

図 13: CPU 利用率の遷移

利用率が全体の 90%近くになったら飽和すると報告している．これから，WWW1 および WWW2 はまだ十分余力を残していると考えられる．

6.2.3 考察

ページフォルトと未使用メモリの残りに注意を払えば，メモリ不足による飽和状態を防止できる．

今回の計測では，WWW1 においてもう少しリクエストレートが上昇したら，飽和状態に移ると考えられる．これは，未使用メモリが少なくなり，OS がページフォルトを発行してメモリを確保していたからである．リクエストレートが高くなると，ページフォルトによるメモリの確保が間に合わなくなり，スラッシング状態になると考えられる．

この状態を防ぐには，ページフォルトの抑制が不可欠である．apache システムでは fork システムコールの発行を抑制する事でページフォルトの発行が抑制されると考えられる．

chamomile でこの現象が現れなかったのはシングルプロセスによる実装のため，システムが消費するメモリ量を抑制できたからであると考えられる．

WWW1 ではプロセスを生成し過ぎでシステムの領域が多く確保されてしまって，未使用メモリが少なくなった．これは非効率的であり，apache の設定の問題点であると言える．

7 まとめ

従来，多くの管理者は上で述べた障害に対して，勘と経験を頼りに管理業務を行ってきた．なぜなら，サーバの性能や管理の指標となるものが，これまでは存在しなかったからである．ベンチマークシステムを用いる事で，サーバの定量的評価は可能である．しかし，運用されているシステムとベンチマークでの結果は大きく事なる．

そこで我々は,「サーバの不具合の検知」を容易に行い,「障害発生時の対応」に何らかのヒントを与える事ができ，広域に分散したサーバ群を集中管理できるシステムを開発した．本システムは，実環境で分散サーバシステムの稼働状況を把握し，サーバ管理者にその状況を伝えるシステムである．本システムを実際に運用しているサーバに適用し，その有効性を明らかにした．

8 参加企業及び機関

本開発は情報処理振興事業協会の未踏ソフトウェア育成事業プロジェクトおよび株式会社アックスの協力を得て開発を行った．

参考文献

[1] LBNL's Network Research Group. http://ee.lbl.gov/.

[2] Steven McCanne, and Van Jacobson. The BSD Packet Filter: A New Architecture for User-level Packet Capture. In USENIX conference, pages 25–29, San Diego,CA, January 1993.

[3] Chamomile Home. http://minatow3.aist-nara.ac.jp/ eiji-ka/chamomile/index.html.

[4] J. Almeida, V. Almeida and D. Yates. Measuring the Behavior of a World-Wide Web Server. Technical report, Computer Science Department, Boston University, October 29 1996.