NAIST-IS-DT9861015

博士論文

WWW サーバ管理支援システムの構築

中村 豊

2001 年 2 月 5 日

奈良先端科学技術大学院大学
情報科学研究科 情報システム学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
博士 (工学) 授与の要件として提出した博士論文である。

提出者：　中村 豊

審査委員：　湊 小太郎 教授
山口 英 教授
福田 晃 教授
砂原 秀樹 助教授

WWW サーバ管理支援システムの構築*

中村 豊

内容梗概

インターネットは急速に普及し，社会基盤の一つとして重要な地位を占めるようになった．中でも，World-Wide Web(WWW) は最も利用されている．ある報告ではインターネットの約 8 割は WWW のトラヒックであるとされている．これから，WWW サービスの重要性が増していると言える．WWW 利用者の立場では，WWW サービスの品質向上を望むようになっている．したがって，サービス提供者 (サーバ管理者) は，WWW サービスの品質の向上を努力する必要が生じている．このサービス品質には，コンテンツの質，量の他に情報を滞る事なく提供するといった事があげられる．サービス品質のコンテンツの質，量に関しては，そのサイト固有の特徴が現れる．そして，WWW サービスを滞る事なく提供するためには，常にサーバの状態を把握する必要がある．

従来行われてきた，ログ解析手法やベンチマークテストなどのサーバシステムの性能計測手法では，サービス品質を十分に考慮していない．そこで，本研究では WWW サービスが WWW サーバシステムが送受信するパケットによって構成されることに着目し，パケットモニタによる WWW サーバシステムの性能計測手法を提案した．第 2 章では提案した手法を用いたシステム Enhanced Network Measurement Agent(ENMA) の設計および実装について述べる．ENMA は WWW サーバシステムを外部から観測するシステムである．外部から観測する事により，サーバに負荷を与える事なく計測できる．実際に運用されている WWW サーバを計測し ENMA の有効性を示す．

第 3 章ではサーバシステムの内部状態の解析について述べる．これまでは，サーバシステムの観測が容易ではなかったために，管理者は自身の勘と経験を基にしてサーバを管理してきた．サーバ管理者はリアルタイムに WWW サーバを観測し，常に正常な状態を維持する義務がある．ENMA を用いる事で，サーバシステムをリアルタイムで監視する事が可能となる．しかしながら，管理者はサーバシステムの計測が目的ではない．管理者の目的は観測を通してサーバシステムの状態を知り，その状態に対応する処置を施

*奈良先端科学技術大学院大学 情報科学研究科 情報システム学専攻 博士論文, NAIST-IS-DT9861015, 2001 年 2 月 5 日.

す事である．このようなサーバの状態を知るためには，サーバシステムの状態についてモデル化を行う必要がある．サーバ状態のモデル化のために，サーバの状態とその遷移条件について考慮しなければならない．そこで本研究ではサーバの状態について定義し，それらの間の状態遷移に関して実験した．そして，性能指標となるパラメータを様々に変化させ，サーバの状態がどのように変化するかを観測した．その結果から，サーバの状態を推測し，状態遷移モデルを示した．我々は，この状態遷移モデルに対応する管理者の行動について議論した．しかし，サーバの状態遷移を適用するためには，サーバシステムの内部情報を抽出する必要性が明らかになった．

第 4 章ではサーバシステムの内部情報を抽出する手法について述べる．サーバの状態遷移を考慮する場合，サーバの内部情報を知る必要がある．サーバの内部情報を得る事でサーバシステム内の様々な挙動を知る事が出来る．サーバの内部情報を得るには，カーネルモニタリングを行う．従来，カーネルモニタリングは負荷が高く実用的でないと考えられてきた．そこで本研究では，低負荷で実用的なカーネルモニタリングシステム (Rep2) を設計，実装した．そして実際に運用されているサーバシステムに適用し，その有効性を示した．そして，サーバ内部，外部による観測の結果から管理者がとるべき行動について議論した．

キーワード

ENMA, WWW サーバ，性能計測，パケットモニタ，カーネルモニタ，管理手法

WWW Server Measurement Support System*

Yutaka Nakamura

Abstract

The Internet is rapidly growing and has become a significant fundamental infrastructure. World-Wide Web (WWW) is using most in the Internet. The WWW traffic occupies approximately 80% of the total traffic carried on the Internet. The WWW is the most important service on the Internet. Regard to a WWW users point of view, improvement of WWW service quality is important. Service providers or server administrators should make effort to improve the WWW service quality. The service quality is quality and quantity of contents and providing information without being delayed. To supply the WWW service without being retarded, the WWW server state is observed.

Log analysis method and benchmark of server performance measurement method do not consider enough to the service quality. The packets that the WWW service construct transmitting and receiving on the WWW server is focused. This thesis proposes the performance measurement method on the WWW server using packet monitoring. Chapter 2, The design and implementation of the Enhanced Network Measurement Agent (ENMA) using our proposal method is described. ENMA can measure server performance without overloading at WWW server. This thesis shows available of the ENMA to measure in actual server.

Chapter 3, an analysis of internal state of server system is explained. The measurement of the server system is difficult in so far. Therefore, server administrators have to manage their servers only with their intuitions and experiences. They must maintain good state on the WWW server to observe server system in real time. The server could be measured using the ENMA in real time. However, objective of the administrator is not measurement of the system. Their object is what knows the server state in system

*Doctor's Thesis, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DT9861015, February 5, 2001.

through the observation and treats the server system against its state. To understand the server state, this thesis models state transition on the server system. condition to model the server state transition should be considered. This thesis defines the server state, and experiment the state transitions. In this test, the performance indices about server system is exchanged, and is observed how exchanging server state. As the result, this thesis guesses the server state and propose the server state transition model. The action of server administrator corresponding to state transition model is discussed. However, to apply this action on state transition, server administrators need extract internal information on the server system.

In chapter 4, the method of internal information of the server system is described. In case to consider of the state transition on the server system, information in the server system should be known. Administrators can know some behavior in the server system to obtain information in the server. Kernel monitoring can get internal information of the server. Kernel monitoring overloads the server system and it is not practical. The design and implementation low load and practical kernel monitoring system is described. This thesis applies this system in actual server and show its avail. The actions of server administrators from observation inside and outside of the server systemis discussed.

Keywords:

ENMA, WWW server, performance measurement, packet monitor, kernel monitor, measurement method

目 次

目 次

図 目 次

図目次

表 目 次

第 1 章　序論

1969 年に ARPANET として始まったインターネットは現在に至るまで休むことなく成長し続けている．インターネットは当初，軍事目的で開発が進められていた．しかし，インターネットが広まるに連れて，軍事以外の様々な方面で利用されるようになった．1990 年代に登場した World-Wide Web(WWW) はインターネットの成長の 1 つの起点である．1993 年イリノイ大学の学生グループによって開発された Mosaic [1] はインターネットの成長に大きな影響を与えた．文献 [2] によると 1993 年において WWW によるインターネットのトラヒックは約 3400 倍の伸びを示したと報告されている．現在，WWW はその利便性から多くのインターネットユーザに利用されている．WWW を利用して，ホテルや航空チケットの予約，オリンピックやワールドカップの中継，ショッピングなどが可能となっている．文献 [2] では，西暦 2000 年 7 月で約 9 億ホストが接続されており，2000 年 10 月で約 2000 万サイト存在すると報告している．この様に，WWW はすでに社会基盤の 1 つとして必要不可欠なものとなっている．

インターネットの発展により，WWW を用いた様々なサービスが提供されている．インターネット利用者はサービスの品質向上を求めている．サーバ管理者の立場では，この様なユーザの要求を満足させる必要がある．サーバ管理者はサービス品質の維持およびサービス品質向上に努めなくてはならない．しかし，WWW を用いた新たなサービスを提供する事によって，サーバ管理者の負担は大きくなっている．さらに現状では，サーバ管理の手法が確立されていないため，管理者の経験やその経験に基づく勘を頼りに管理業務が行われている．このようなサーバ管理者の負担を軽減するために，本論文では WWW サーバ管理者のための管理支援システムの構築を目指している．

インターネットが急速に普及した 1990 年代からルーティングやトラヒックの管理は盛んに行われている．1994 年には世界で始めて WWW を用いて「カリフォルニア州議会選挙」がインターネットにより中継された [3]．WWW が普及するに連れて，その問題点を解決するための様々な研究が行われるようになった．

研究の初期では WWW サーバ (CERN [4],NCSA [5]) の出力するログ解析やルータ, スイッチなどのトラヒック解析を基に WWW サーバの内部構造解析やモデリングなどが行われた [6]．その後，HTTP の欠点を指摘した研究がなされ，その解析や改良がなされた [7] [8] [9]．また，初期の結果を参照し，WWW のアクセスパターンやトラヒックパターンを考慮したベンチマークが開発された [10] [11] [12]．さらに，より詳細な解析のためにカーネルモニタを用いた性能計測 [13] といった研究がなされている．

この他に様々なツールが開発されている．パケットモニタリングツールとして tcpdump [14], sniffit [15], ethereal [16], snort [17]．帯域測定ツールとして netperf [18], pchar [19]．帯域観測ツールとして MRTG [20], OC3MON [21] などが開発されている．そして，WWW のベンチマークツールとして，SPECweb99[22], httperf [23], WebStone [24] など，様々な状況に合わせたツールが開発，利用されている．

このように現在，解析，モデリング，計測，性能評価といった分野で様々な研究が行われている．そして，計測のための様々なツールが開発されている．しかし，これまでの研究や開発されたツールはサーバ管理にフィードバックする事を念頭においているものは存在しない．この様な理由から，サーバ管理者は自身の勘と経験を基にサーバを管理してきた．IETF [25] では，WWW サーバの管理のための MIB を定義 [26] している．しかし，実際に用いられている例はない．ネットワークトラヒックや経路表に関する管理はこれまで盛んに行われてきている．しかし，ネットワークサーバの状態を管理するツールは少なく，研究もなされていない．

我々は，サーバの状態を観測するために新たな計測ツール (ENMA) を開発した．従来のログ解析手法では，サーバに対するアクセスパターンやクライアントの分布を得る事が出来る．しかし，サーバシステムの性能を十分に示す事は出来ない．ベンチマーク手法では，ある特定の条件の下でのみ性能を評価できる．したがって，ベンチマークによる結果と実際に運用されているサーバシステムの性能とは大きくかけ離れている事が多い．また，ログ解析，ベンチマークともに，リアルタイムにサーバシステムを観測する事が出来ない．そこで我々は WWW サーバがパケットの送受信によってデータ交換している事に着目し，パケットモニタリングによってサーバの状態を観測するツールを開発した．この ENMA を用いる事で，サーバ管理者はリアルタイムにサーバの状態を把握する事が可能となる．

しかし，サーバの状態について我々は十分な定義をしていなかった．そのため，たとえサーバの処理が飽和していたとしても，どの部分にボトルネックがあるのか把握する

事が出来なかった．そこで我々は独立した環境でサーバシステムに対して実験し，サーバシステムの状態について定義した．そしてサーバシステムの状態遷移の条件について考察した．その実験の結果，サーバシステムの状態遷移を把握するためには，サーバの内部情報が不可欠である事が明らかとなった．

そこで，我々はサーバの内部情報を抽出するプログラム (rep2) を設計，実装し評価した．従来，カーネルモニタリングによるサーバ観測は，負荷が高く実用的でないと考えられていた．文献 [11] ではカーネルモニタリングシステムを用いたサーバ計測を行い，そのオーバヘッドは約 4%であったと報告している．我々は，このシステムよりも負荷の低いカーネルモニタリングシステムの実装を目指した．

また，文献 [11] では，サーバシステム内でカーネル時間が約 90% 程度になった時に性能が飽和すると報告している．しかし本論文ではカーネル時間が 90%程度という状況をつくり出す事が出来なかった．そのため，状態遷移について確認できなかった．

1.1. サーバ管理

サーバ管理者はサーバを管理することが業務である．本節では，具体的にサーバの管理業務について述べる．

まず始めに，障害回避について述べる．サーバの運営者にとって最も避けたい事はサーバがダウンすることである．もし，サーバがダウンしたならサーバ管理者は障害の場所 (ネットワーク，ハードウェア，アプリケーション，OS) を特定し，対処しなくてはならない．この様な対処作業は管理者がそれまで経験してきた知識によって作業されてきた．

次の管理業務としては，障害の発生しないシステムを構築もしくは設計する事である．上のように障害が発生すると，その対処は迅速に行わなくてはならない．したがって，サーバ管理者には出来るだけ障害を起こさないようにシステムを構築する事が求められる．この様なサーバシステム構築に関しても，上と同様にサーバ管理者のこれまでの経験や知識に基づいて行われてきた．

またこれら以外としては，サーバシステムのパフォーマンスチューニングが挙げられる．サーバシステムが飽和していると判断されるなら，それよりも高性能のシステムを導入する事が最も容易な解である．しかし，これは資金が必要となる．資金を用いずにサーバシステムの性能を少しでも高める努力をサーバ管理者は行わなくてはならない．

1.2.　サーバ管理の支援

本論文でのサーバ管理の支援とは，具体的には 1.1 節で述べた，障害回避，障害回避のためのシステムの設計およびパフォーマンスチューニングについて，何らかの指標を与える事を目指している．例えば,「サーバシステムはもうこれ以上のリクエストを処理できないので，何らかの方策を考えないといけない」という状況になった時,

- 資金が潤沢である

  より高性能のシステムを導入する

- 資金がない

  サーバシステムのパフォーマンスチューニングを行う．もしくは，トラヒックを制限し，サーバシステムの負荷を下げる．

と言った事が考えられる．ここで述べたのは 1 例であるが，サーバ管理にはこれ以外にも様々な存在する．詳細については 6章で述べる．

1.3.　サーバ観測

本論文で述べる観測 (パケットモニタリング，カーネルモニタリング) は PC を用いた観測である．本節では，本論文における観測の必要条件について説明する．

現在，多くのメーカがネットワークにおけるパケットのエラーやネットワークインタフェースの故障などを検出するために，高精度の計測機器を開発している．しかし，本研究では，ネットワークサーバの計測であるため，パケットエラーの検出といった高精度な計測は必要ない．

パケット単位のマイクロ秒の粒度の計測よりも，コネクション情報などの荒い精度での計測が重要になる．ユーザに対する応答などを計測する場合，パケット 1 つ 1 つの正確性よりも，コネクション全体に渡る計測が重要である．この様な計測を行う場合，専用ハードウェアを用いるよりも PC を用いて計測する方が安価かつ容易に計測システムを構築する事が可能である．したがって，本論文では計測システムを PC で構築している．

1.4. 計測の精度および粒度

本論文では，パケットモニタリングを用いた観測システムを設計，実装している．本節では，観測における精度と観測粒度について述べる．

1.4.1 観測精度

本論文で述べている ENMA では，パケット到着時刻をマイクロ秒単位で記録できる．しかし，オペレーティングシステムやモニタリングライブラリに依存している時刻であるので，下一桁は有効数字でないと考えられる．WWW による通信では，コネクション開始から終了まで数ミリ秒から数秒かかる．また，サーバアプリケーションの応答時間などについて考えると，その時間は数十マイクロ秒から数ミリ秒である．したがって，観測の精度は，計測する性能指標により，ミリ秒から数十マイクロ秒までの範囲が適当であると考えられる．

1.4.2 観測粒度

本論文では，1 本のコネクションに関する情報と，サーバシステム全体に関する情報を観測する．本節では，それぞれの観測粒度について述べる．

ENMA はパケットモニタリングによる観測を行っているので，パケットを粒度にして観測するのが理想である．しかし，ネットワーク上のパケットを全てモニタリングし，なおかつディスクに保存する事は非常に困難である．その理由は，ディスクへの保存の速度がネットワークの速度よりも低速であるためである．したがって，パケットを粒度として観測するには，高速なハードウェアを用いる，もしくはディスクではなく，メモリに保存するという方法が考えられる．本論文では，メモリに一旦パケットの情報を保存し，コネクション単位に情報を縮退してディスクに記録している．こうする事で，メモリの消費量を少なくし，かつディスクへの出力も可能にしている．

一方，リアルタイム性が重要な情報に関しては，10 秒毎に計測している．これは，計測の間隔が短くなると，計測システム自身が負荷の原因となることが考えられるためである．しかし，10 秒という単位が妥当なものであるかどうかは議論が必要である．

1.5.　本論文の構成

本章では，インターネットの発展および WWW の普及，発展の経緯について述べ，これまでの過去の研究について述べた．その上で現在の問題点に触れている．

2 章では WWW サーバ性能計測システム ENMA の設計と実装について述べる．ENMA は従来の性能計測手法であるログ解析やベンチマーク手法とは異なりパケットモニタリングを元にした性能計測システムである．2 章では，ENMA の設計および実装手法について述べ，実際の運用例について述べる．

3 章では，2 章で述べた ENMA を用いたサーバの状態解析について述べる．サーバシステムをモデリングし解析した研究は過去にも行われている．しかし，管理に結びつけて考えられものはこれまでない．そこで，3 章では，サーバの状態を定義し，その状態に対してサーバ管理者が行うべき行動について議論する．

4 章では，我々の提案する新たなカーネルモニタリングシステム (rep2) について述べる．3 章で，サーバの状態を定義し，その状態遷移について議論する．しかし，正確な状態遷移の把握にはカーネルモニタリングが不可欠である．そこで 4 章では，これまで高負荷で実用的でないと思われていたカーネルモニタリングシステムを新たに設計，実装する．そして実際の運用例について述べる．

5 章では，ENMA および rep2 を用いた性能計測システムの応用として，代理サーバを計測した場合について述べる．

6 章では，ENMA および rep2 を用いた性能計測の結果を踏まえた，サーバ管理に関する議論を行う．

7 章は結論であり，本論文で得られた知見および，今後の課題について述べる．

第 2 章　パケットモニタによる WWW サーバの性能計測システムの設計と実装

WWW による情報提供サービスの充実にともない，WWW サービスの品質に対する要求が高まってきた．従来行われてきた，ログ解析手法やベンチマークテストなどのサーバシステムの性能計測手法では，サービス品質を十分に考慮していない．WWW サービスは WWW サーバシステムが送受信するパケットによって構成される．これに着目し，我々はパケットモニタによる WWW サーバシステムの性能計測手法を提案した．本章では，パケットモニタによる WWW サーバシステム性能計測手法を提案し，その手法を用いたシステム Enhanced Network Measurement Agent(ENMA) の設計および実装について述べる．そして，実際に運用されている WWW サーバの性能計測を行い ENMA の有効性を示す.

2.1.　まえがき

WWW は社会に急速に広まっている．現在，3,500 万を超える WWW サーバが存在しており，WWW トラヒックは Internet 上のトラヒックのおおよそ 80%を占めるようになった.

この急速な普及にともない，様々な種類の WWW サービスが提供されるようになった．例えば，WWW を用いて航空チケットや宿泊施設の予約が可能となった．また，イベントの中継にも WWW が利用されるようになった．例えば，1998 年の長野オリンピックやサッカーのワールドカップなどがあげられる.

WWW サービスの充実にともない，サービス利用者はサービス品質を意識するようになった．サービス品質とは，WWW コンテンツの内容やコンテンツの表示が開始されるまでの時間もしくは表示が終了するまでの時間である．特に，サービス利用者は「より

速くコンテンツが表示される」ということを要求するようになった．

一方，WWW サーバ管理者の立場では，利用者の要求を満たすより良いサービス品質を提供するために，サーバシステムの性能を知る必要がある．サービス品質を正確に測定するには，1) サーバシステムの全体の挙動を測定可能，2) 運用中のシステムの性能を計測可能，3) 計測がサーバシステムに影響を与えない，といった要件が必要である．しかし，ログ解析手法やベンチマークテストといった，これまでの性能解析手法ではサービス品質について考慮されていない．そこで，我々はパケットモニタによる WWW サーバ性能計測手法を提案する．パケットモニタによる計測では，外部からサーバシステムの性能計測が可能であるため，上記の要件を満足することができる．

第 2 章では，パケットモニタによる WWW サーバシステム性能計測手法を提案し，その手法を用いたシステム Enhanced Network Measurement Agent(ENMA) の設計および実装を行う．そして，実際に運用されているサーバシステムの性能計測を行い ENMA の有効性を明らかにする．

2.2.　パケットモニタによる WWW サーバ性能計測手法の提案

本節では，サービス品質と WWW サーバシステムの性能との関係について述べ，従来の性能計測手法の問題点および我々が新たに提案する手法について説明する．

2.2.1　サービス品質

WWW サービスの品質には以下に示す 4 つの項目が考えられる．

1. コンテンツの質と量

   コンテンツの質と量には，各サイトの特徴が現れる．コンテンツ (データファイル) の大きさや数量はサーバの性能に大きく依存する．本研究では，この様なコンテンツの配置や大きさに関しては考慮しない方針である．

2. クライアントが WWW コンテンツを要求してから表示が開始されるまでの時間

   クライアントが WWW コンテンツを要求してから表示が開始されるまでの時間は，様々な要因によって変動する．WWW サーバの性能について考えた場合，この時間が短いということはサーバの性能が高いことを示している．

3. コンテンツが表示され始めてから表示が終了するまでの時間

コンテンツが表示され始めてから表示が終了するまでの時間も，先の項目と同様のことが言える．WWW サーバの性能について考えた場合，この時間が短いということはサーバの性能が高いことを示している．

4. どれだけ多くの利用者に対して同時にサービスを提供することができるか

   より多くの利用者にサービスを提供する事が，サーバとしての性能が高い事を示している．

以上の WWW サービスの品質についての検討の結果，WWW サーバ管理者に必要な項目は，2,3,4 である．これは，利用者は「より速くコンテンツが表示される」ということを最も期待しているからである．

2.2.2 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．

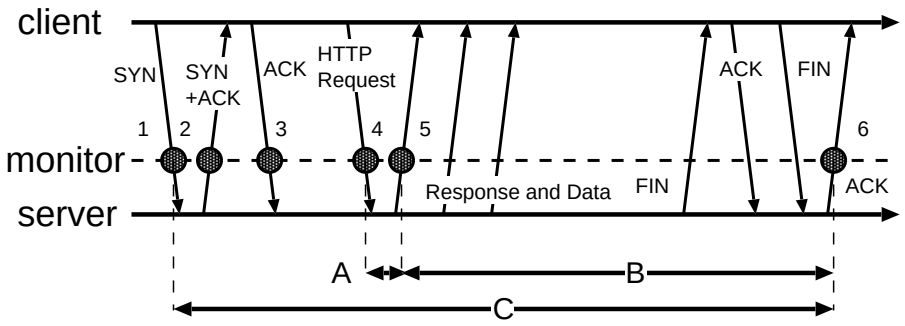

図 2.1 性能指標

2.2.2.1 同時コネクション数 (N_c)

同時コネクション数は WWW サーバシステムが同時に処理している HTTP コネクションの数である．この項目は，サービス品質の「どれだけの利用者に対して同時にサービスを提供できるか」に関連する．

2.2.2.2　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 2.1 の 1) を観測してから，最後の ACK パケット (図 2.1 の 6) を観測するまでの時間である (図 2.1 の区間 C)．この項目は「コンテンツが表示され始めてから表示が終了するまでの時間」に関連する．

2.2.2.3　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 2.1 の区間 A である．この項目は「クライアントが WWW コンテンツを要求してから表示が開始されるまでの時間」に関連する．

2.2.2.4　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 2.1 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．この項目は「コンテンツが表示され始めてから表示が終了するまでの時間」に関連する．

2.2.3　従来の性能計測手法の比較

WWW サーバの性能計測方法は図 2.2 に示すように以下の 3 手法が考えられる．

1. ログ解析
2. カーネルレベルモニタリング
3. ベンチマークテスト

本節では 2.2.2 における性能指標に着目し，これらの手法の利点および欠点について述べる．

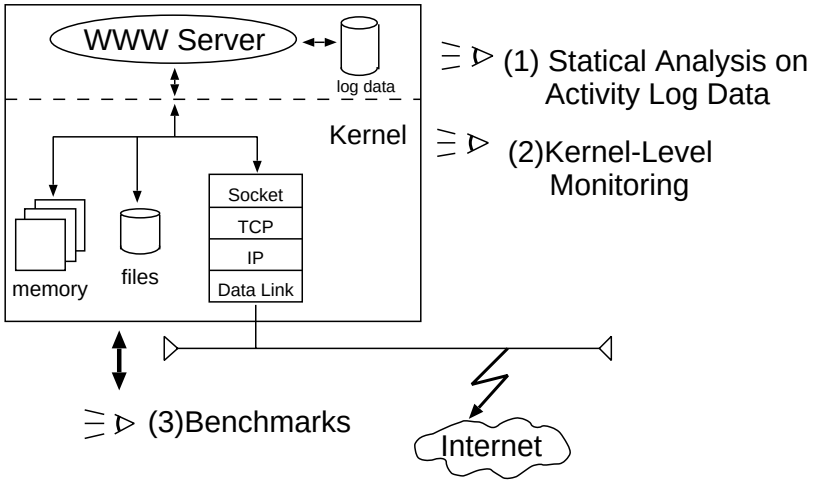

図 2.2 従来の WWW サーバ性能計測方法

2.2.3.1 ログ解析手法

この手法は，WWW サーバが出力したログの統計解析を行うもので，WWW サーバの性能解析手法として一般的に用いられている．この手法では，クライアントのアクセスパターンやリクエスト処理時間などを知ることができる．しかし，アプリケーション層で測定しているために，サーバ管理者に必要な，応答時間，データ転送時間，同時コネクション数は測定することができない．

2.2.3.2 カーネルレベルモニタリング

この手法では，システムコール内での I/O の数，シグナル処理の数，ファイルシステムオペレーションの数などを明らかにできる．しかし，この手法でサーバ管理者に必要な，応答時間，データ転送時間，同時コネクション数を測定するには，カーネルへの変更が必要になる．商用 OS や専用ハードウェアを用いたサーバなどを用いた場合，カーネルの変更は非常に困難となる．また，これは計測がサーバシステムに影響を与えるので WWW サーバシステム自体の性能劣化を引き起こす可能性がある．

2.2.3.3 ベンチマークテスト

WWW サーバ計測の代表的なベンチマークプログラムとして，SPECWeb99 [22] や WebStone [24] がある．ベンチマークテストでは，サーバ管理者の必要とする性能指標を

測定することは可能である．しかし，ベンチマークテストでは特別な環境でテストを行うので，実際の運用時のデータとは同じ結果が得られない．また，運用中の大規模 WWW サーバにおいてベンチマークテストのためにサービスを中断することは困難である．

2.2.4　パケットモニタによる WWW サーバ性能計測手法

前節で述べた方法では，サーバ管理者が必要とする性能指標を測定することは困難であった．そこで我々は，WWW サービスが WWW サーバシステムにおいて送受信されるパケットによって構成されることに着目し，パケットモニタによる WWW サーバシステムの性能計測手法を提案する．

WWW サービスは HTTP 上で行われており，HTTP は TCP を用いている．TCP はパケットを観測することで TCP のコネクション数や，その状態遷移を監視することができる．

本手法ではサーバシステムと同一のネットワークセグメント上に計測ホストを設置し，パケットモニタリングを行い，対象となるサーバシステムの外部から性能計測を行う．

新たに提案する手法には以下に示す特徴がある．

A. トランスポート層での挙動を観測できる

パケットモニタリングを行い，TCP コネクションを解析することによって，トランスポート層での挙動をカーネルレベルモニタリングと同等，あるいはそれに近い精度で観測することができる．

B. WWW サーバシステムに影響を与えない

本手法はパケットモニタリングによる外部からの性能計測である．このため，本手法の計測によって，WWW サーバシステムに影響を与えない．

C. 運用中のサーバシステムを測定できる

本手法は運用中の WWW サーバシステムに対するコネクションの解析を行うことを想定している．このため，性能計測のために運用の停止や，運用状態とは別の特別な環境を用意する必要がない．さらに，運用状態のデータを得ることができるため，より正確なサービス品質の分析を行うことができる．

D. 特定のサーバシステムに依存しない

本手法では WWW サーバシステム自体に変更を加える必要がない．このため，WWW サーバシステムが商用 OS で運用されているといった場合にも計測を行うことができる．

表 2.1 性能評価手法の比較

	Log Analysis	Kernel Level Monitoring	Benchmark Test	Packet Monitoring
A	×	○	△	○
B	○	×	×	○
C	○	△	×	○
D	△	×	○	○

従来の手法および我々の提案した手法の比較を表 2.1 にまとめた．

2.3. システム設計

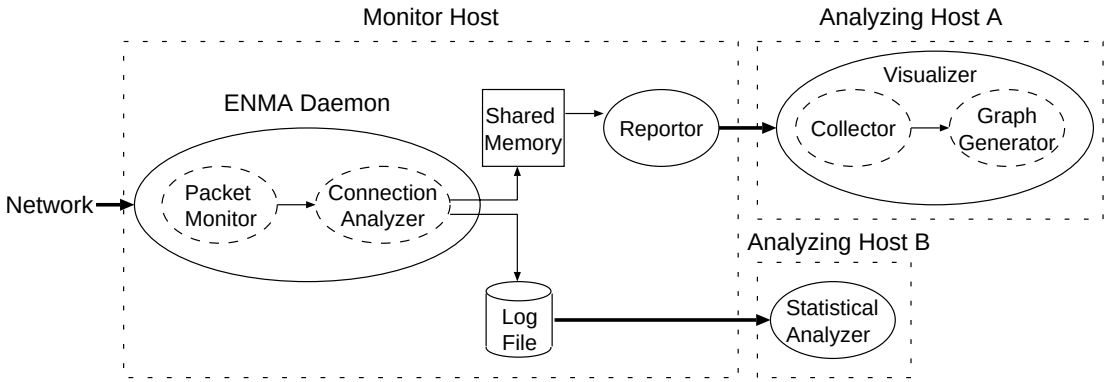

図 2.3 システム構成

本節では，提案した計測手法を用いた WWW サーバ計測システムの設計について述べる．また，本システムを Enhanced Network Measurement Agent(ENMA) と名付けた．

ENMA は，ENMA デーモン (ENMA Daemon) と性能解析プログラム群 (Performance Analysis Workbench) の二つの要素から構成されている (図 2.3).

2.3.1　ENMA デーモン

ENMA デーモンはリアルタイムにパケットモニタリングを行う．この ENMA デーモンはパケットモニタモジュールとコネクションアナライザモジュールの 2 つのモジュールから構成されている．

パケットモニタモジュールは ENMA が設置されているネットワーク上の HTTP コネクションを構成するパケットのモニタを行う．モニタされたパケットから，IP データグラムを抽出し，コネクションアナライザへ渡す．

コネクションアナライザ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．

2.3.2　性能解析プログラム群

性能解析プログラム群は ENMA デーモンが提供するデータの様々な解析を行うプログラムの集合であり，以下に示す 2 種類に分類される．

- 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．

- これまでに処理したコネクション数や，現在処理しているコネクション数といった，現在の WWW サーバの処理状態を知るために，リアルタイムな性能解析を行うプログラムを用意した．このプログラムは ENMA デーモンが用意している共有メモリ領域からデータを読み込む．

2.4. 実装

本論文では，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 2.2.7 や IRIX6.3 である．本節では ENMA の実装について述べる．

2.4.1 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 [27]) を用いている．このライブラリは BSD 上の Berkeley Packet Filter(BPF [28]) 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．

2.4.2 コネクションアナライザ

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 2.4 に示す．

図 2.4 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に処理する．

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．

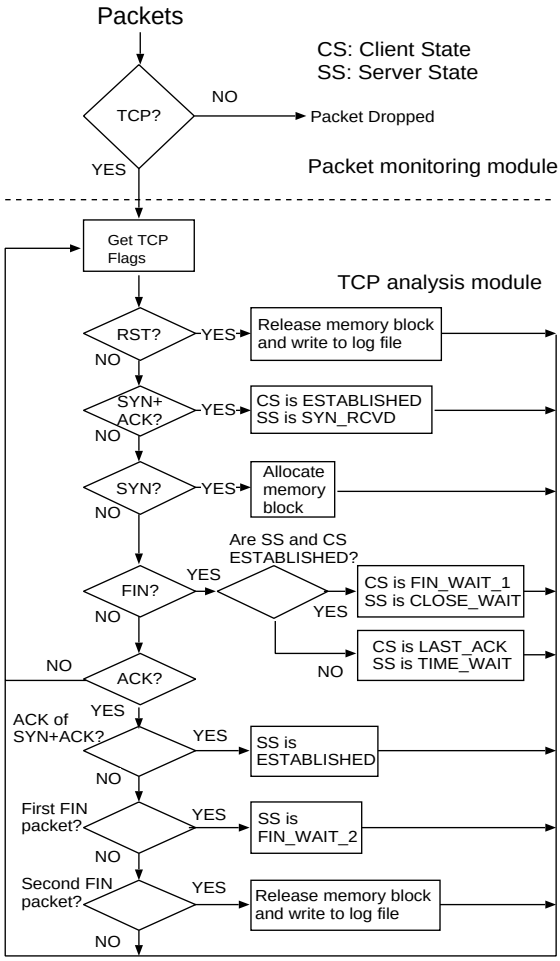

図 2.4　コネクション解析アルゴリズム

3. SYN の場合，メモリブロックを割り当て，状態を初期化する．

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．

2.4.3 性能解析プログラム群

図 2.3 に示すように，性能解析プログラム群には 3 つのプログラムがある．統計データ報告プログラム (Reporter) および視覚化プログラム (Visualizer) は，リアルタイムな解析のために用いられる．ENMA デーモンは ENMA システム内の多くの計算機資源を用いるので，ENMA デーモンと視覚化プログラムの両方を同一の計算機で実行することは，ENMA システムの性能劣化を引き起こす可能性がある．最悪の場合，ENMA デーモンがネットワーク上のパケットを喪失する可能性がある．そのような状況を回避するために，リアルタイム解析を行うプログラムを統計データ報告モジュールと視覚化モジュールの 2 つのモジュールに分割し，視覚化モジュールは異なる計算機上で実行することを可能とした．

統計解析プログラム (Statistical Analyzer) は ENMA デーモンが出力したログファイルの統計解析を行うプログラムである．以下に各プログラムの詳細を述べる．

2.4.3.1 統計データ報告プログラム

統計データ報告モジュールは図 2.3 に示すように，共有メモリから統計データを得る．後で述べる視覚化プログラムの要求を受け付ける．

2.4.3.2　視覚化プログラム

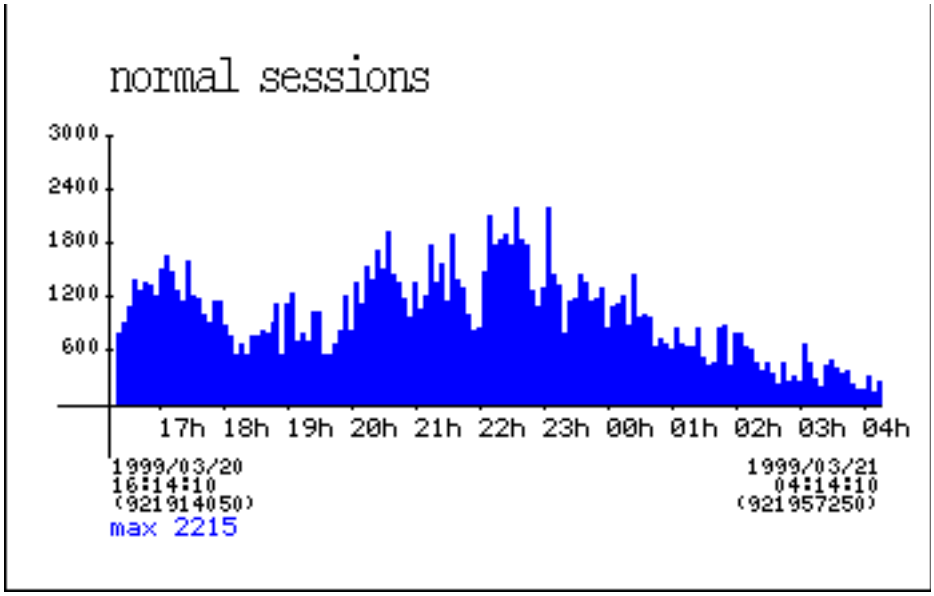

図 2.5　正常セッションの経時変化 (視覚化モジュールの例)

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．データは統計データ報告モジュールに対して要求を発行することで総計データを入手する．その入手した統計データより，図 2.5 のような性能指標に関するグラフを生成する．

2.4.3.3　統計解析プログラム

統計解析プログラムは，ENMA デーモン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同時コネクション数などの頻度分布を集計する．

2.5.　ENMA の有効性の検証

ENMA の有効性を確かめるために，ENMA を実際に動作している WWW サーバに対して適用し，有効性の検証を行った．

2.5.1 WWW サーバの処理能力差の計測

我々は，ENMA を用いることで WWW サーバの処理能力差を計測することが可能かどうかの検証を行った．2.2 節で述べたように，応答時間 (T_r) は WWW サーバの性能を表している指標の一つと考えられる．そこで，我々は処理能力の異なる 2 つのサーバシステムに対して ENMA を適用し，これらの値を測定した．

2.5.1.1 システム構成

この実験では，PentiumII 200MHz のホスト A と 80486DX2 66MHz のホスト B，2 台の WWW サーバを用意した．両ホストとも OS は FreeBSD 2.2.7 で WWW サーバアプリケーションとして Apache 1.3.1 を用いた．サーバ，クライアント，モニタホストを同一のセグメントに接続し実験を行った．

この実験では，ランダムに WWW サーバ上のオブジェクトにアクセスした．そして，サーバに約 200 のオブジェクトを置き，並行してすべてのオブジェクトへアクセスを行った．1 回の測定で約 10,000 アクセスを実行し，T_c, T_r を測定した．

2.5.1.2 結果

図 2.6,2.7 に測定の結果を示す．図 2.6 は応答時間 (T_r) の頻度分布である．

図 2.6 より，ホスト A では T_r が 20 msec 付近に最も多く分布している．しかしホスト B ではピークは見当たらず，T_r は 50 msec よりも長くなっている．したがって，図 2.6 からはホスト A の方が高速に処理していることがわかる．

図 2.7 はコネクション継続時間の頻度分布である．図 2.7 より，ホスト A ではコネクション継続時間 (T_c) が 3 msec 付近に最も多く分布しており，ホスト B の T_c は 200 msec 付近に分布している．これから，ホスト A の方がホスト B よりもコネクションに関する処理を高速に行っていると言える．

これら 2 つのグラフから，ホスト A とホスト B との性能の違いを見ることができる．ENMA により，容易に WWW サーバの性能の違いを示すことができた．

2.5.2 大規模 WWW サーバの計測

ENMA が実際に運用されている WWW サーバの性能計測を行うことができるかどうか確かめるために，第 80 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 WWW サーバを計測した．この WWW サーバでは，1 日に数千万アクセスが得られた．

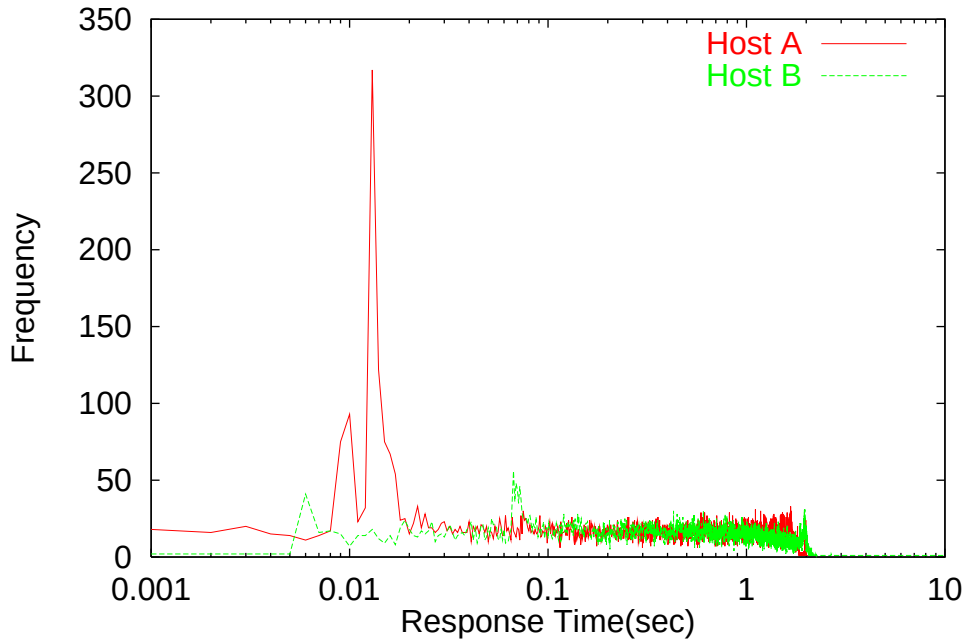

図 2.6　応答時間の頻度分布

2.5.2.1　システム構成

ターゲットとなる WWW サーバホストは Sun Enterprise 450(CPU は Ultra SPARC II 300MHz 2 個，512MB のメモリ) である．OS は Solaris 2.6 で WWW サーバプログラムは Apache [29] 1.3.1 である．ENMA デーモンが動作しているホストは IBM-PC(PentiumII 300MHz，64MB メモリ)，OS は FreeBSD 2.2.7-RELEASE を用いた．WWW サーバホストと ENMA を同一セグメントに接続し，パケットのモニタを行った．

2.5.2.2　結果

図 2.8 はコネクション継続時間 (T_c) の頻度分布を示している．WWW サーバによって得られたログの解析では T_c が 1 msec 付近に最も多く分布している．しかし ENMA によって得られたログの解析では，20 msec 付近に最も多く分布している．

WWW サーバによって生成されたログは，そのシステムのアプリケーション層における記録である．そのため，ログにおける T_c は図 2.9 に示したように，accept システム

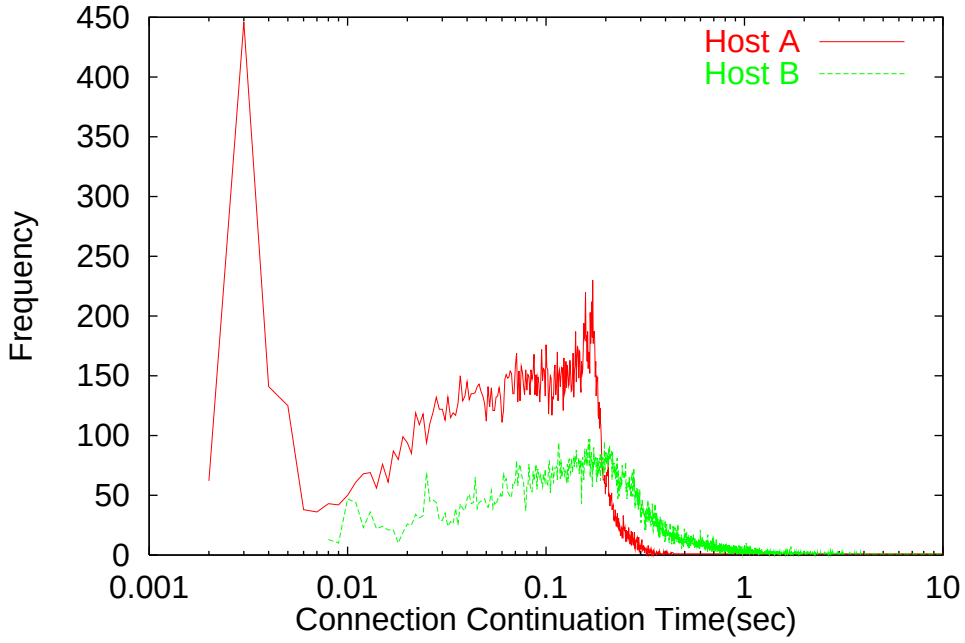

図 2.7 コネクション継続時間の頻度分布

コールが終了してから close システムコールが終了するまでの時間である．この多くが 1 msec 付近に分布しているということは，図 2.9 のこれらのシステムコールが直ちに終了することを示している．例えば，write システムコールに関して言えば，今回測定で用いた Solaris 2.6 の送出ソケットバッファサイズは 8KByte であり，サーバが処理したオブジェクトの大半が 8KByte に収まる (図 2.10) ことから，write システムコールはオブジェクトをソケットバッファに格納した直後に終了することを示している．このようにアプリケーション層におけるサーバログでは正確な T_c は得られない．

一方，ENMA による T_c は，サーバのログによる T_c に加え，TCP 層でのコネクション確立と切断の手続き時間を含んでいる．さらに，close システムコールが呼ばれた後の，OS によるパケット送出処理時間も含んでいる．したがって，ENMA によって測定された T_c は TCP 層でのコネクション継続時間である．よって，ENMA のログファイルの解析結果は WWW サーバのログファイルより正確な性能指標を示したと考えられる．

同時コネクション数 (N_c) を図 2.11,2.12 に示す．WWW サーバおよび ENMA によるロ

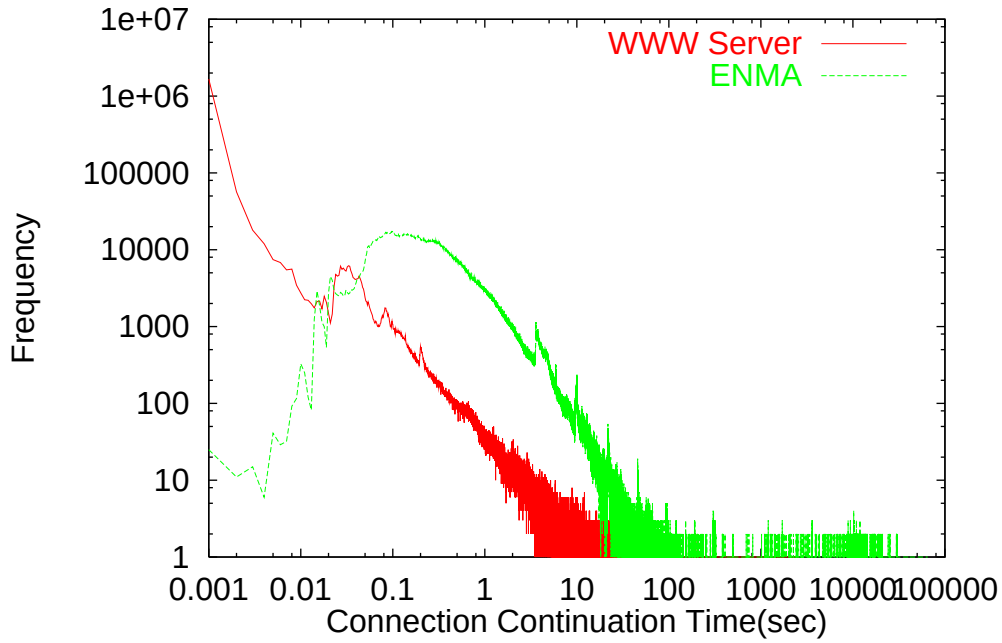

図 2.8　コネクション継続時間の頻度分布

グのそれぞれの T_c は異なっているので，T_c から算出する N_c の解析結果も異なる．ENMA が報告する T_c と WWW サーバのログによる T_c では，ENMA による T_c の方が長くなっている．T_c が長い場合には時間の重なりが多くなるので，N_c も大きな値となる．

図 2.11,2.12 より WWW サーバのログからの解析ではピーク時の N_c は 550 である．しかし，ENMA ではピーク時の N_c は 13,000 である．

ENMA で十分にパケットを受信できたかを検証するために，単位時間あたりに受信したコネクション数を図 2.13 に示す．図 2.13 では，WWW サーバのログと ENMA のログでは，大部分が一致している．したがって，ENMA ではパケット喪失がほとんど発生しなかったと考えられる．

これらの結果より，ENMA を用いることで WWW サーバのログ解析ではわからなかった WWW サーバの性能指標を計測することができた．これより ENMA は実際に運用されている WWW サーバに対して，性能計測を行うことができることが明らかとなった．

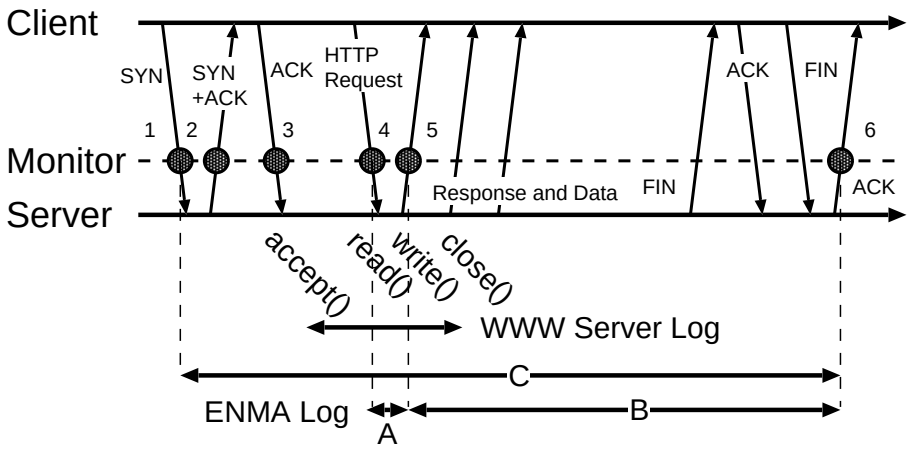

図 2.9 コネクションとシステムコール

2.6. 問題点

本節では，これまで述べてきた ENMA の問題点について述べる．

2.6.1 パケット喪失

モニタリングによるパケット喪失は，正しい性能計測を行うことができなくなるので，パケット喪失は重要な問題である．ENMA における，パケット喪失の解決方法を以下に述べる．

2.6.1.1 サーバホストと同等な高速な計算機を用いる

WWW サーバアプリケーションと ENMA デーモンの計算機に与える負荷を考慮すると，WWW サーバアプリケーションの方が高負荷であると考えられる．したがって同等の計算機を用いた場合には，ENMA デーモンでのパケット喪失を抑制することができると考えられる．

2.6.1.2 オーバヘッドの小さいパケットモニタ機構で動作させる

一般的に UNIX を用いるパケットモニタリングは数百 Mbps や高帯域ネットワークに適用する場合，OS のオーバヘッドに大きく影響される．したがって，Gigabit Ethernet

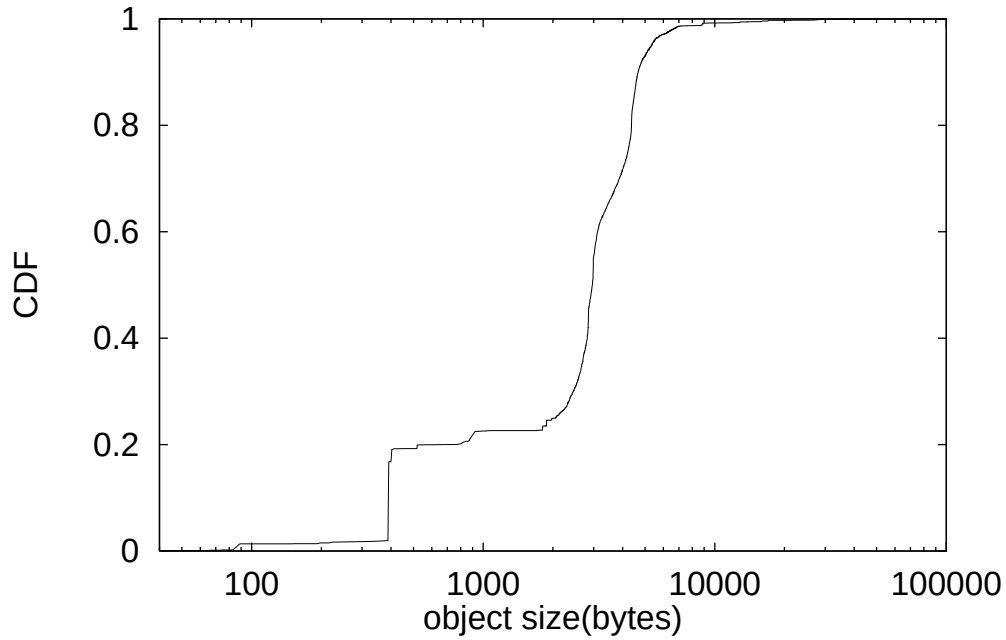

図 2.10　オブジェクトサイズの累積分布

や他の高速ネットワークに ENMA を適用した場合には，オーバヘッドの小さい OS 上で動作させる必要がある．

2.6.2　パケット順序の入れ替わり

別の問題としてパケット到着の順序が入れ替わることがあげられる．現在の実装では，ENMA はシーケンス番号の追跡を行っていない．したがって，パケット順序の入れ替わりを検出することができない．これにより WWW サーバの状態の検出を失敗する可能性があるので，性能計測に問題が生じる恐れがある．

2.7.　他のパケット モニタリングシステム

従来のパケットモニタリングシステムとして tcpdump [14] がある．tcpdump はネットワーク上のトラヒックをモニタリングしパケットを観測するための汎用のプログラム

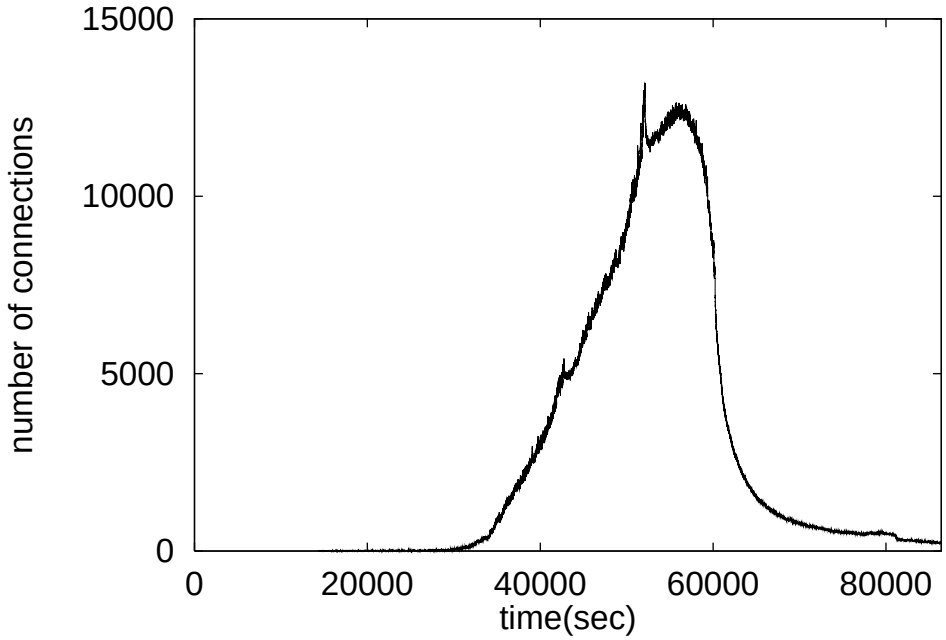

図 2.11 ENMA のログ解析による同時コネクション数

である．しかし，汎用のプログラムであるため HTTP に特化した情報を取り出すことができない．また，通信状態をリアルタイムでモニタリングする用途には向かない．そして，汎用であるためプログラムのホストに与える負荷が大きく，高負荷な WWW サーバの計測を行う場合には，パケット喪失の可能性がある．

それに対して，ENMA ではリアルタイムに HTTP に特化したパケットモニタリングを行う．また，コネクション単位でログへの出力を行っているため，ログ出力のオーバヘッドが tcpdump を比べて小さい．さらに，ログファイルのサイズを抑制することができる．

2.8. おわりに

WWW サービスではサービスの品質がもっとも重要である．WWW サーバ管理者はサービスの品質を維持するために，サーバシステムの性能を知る必要がある．従来の手法

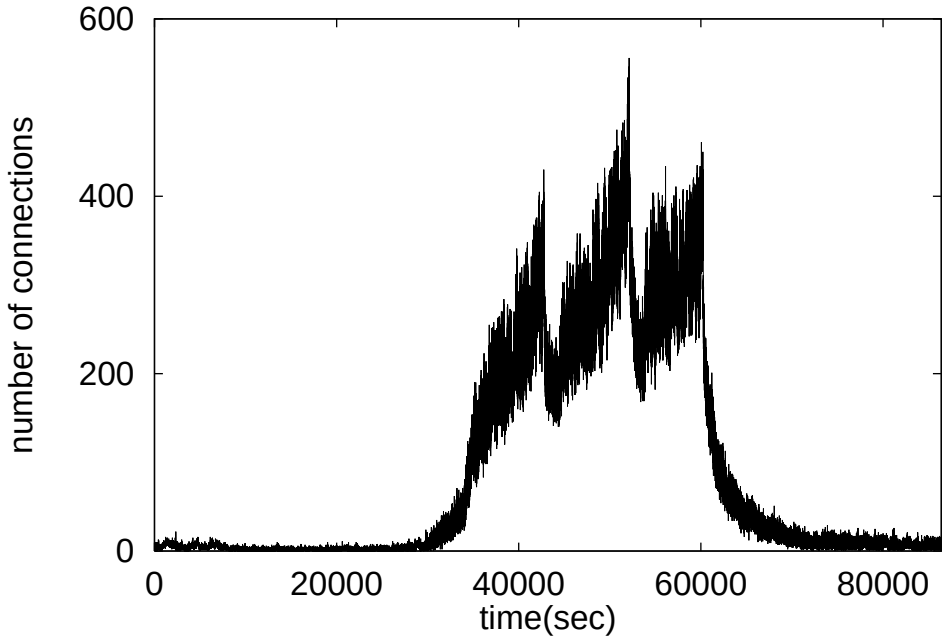

図 2.12　WWW サーバのログ解析による同時コネクション数

では，サービスの品質に関連する性能計測を行うことは困難であった．例えば，WWW サーバのログ解析では，アプリケーション層での解析であるため，サーバ管理者の要求する性能指標を得ることができなかった．

そこで我々は，WWW サービスが WWW サーバに対するパケットの送受信で構成されていることに着目し，パケットモニタによる性能計測手法を提案した．提案した手法には以下に示すような特徴がある．

- WWW サーバシステム全体の挙動を測定できる．
- WWW サーバシステムに影響を与えない．
- WWW サーバシステムを変更する必要がない．
- 運用中のシステムを測定できる．

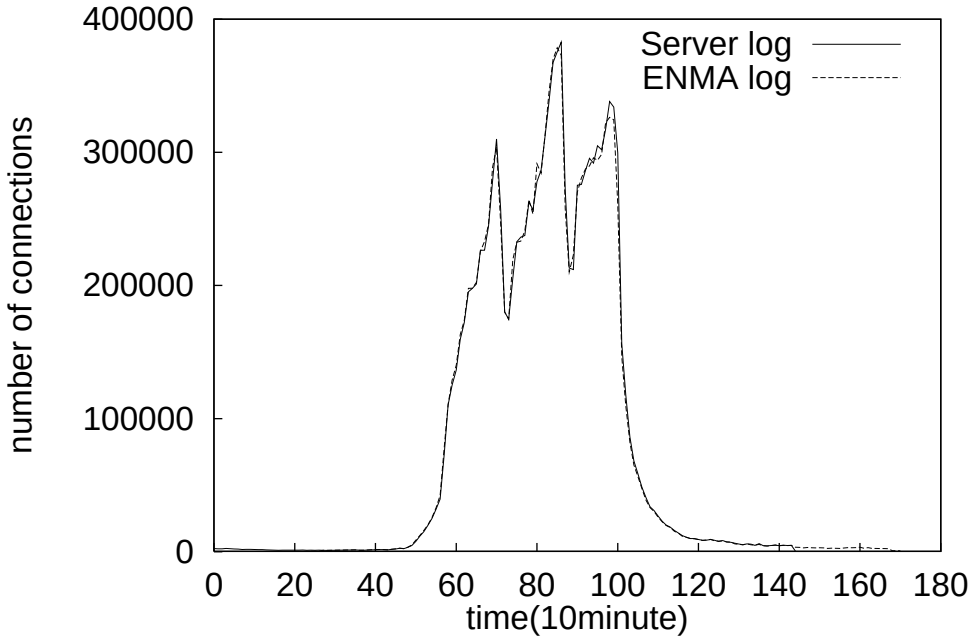

図 2.13 単位時間あたりに受信したコネクション数

我々は，提案した手法を用いたシステム ENMA の設計と実装を行った．ENMA を用いることで，同時コネクション数，応答時間，データ転送時間などの性能指標を測定することができた．そして，ENMA を運用されている WWW サーバに適用し，実際に運用されている WWW サーバシステムの性能計測を行うシステムとしての有効性を示した．

第 3 章　パケットモニタを用いた WWW サーバの内部状態解析

サーバシステムの管理は管理者の勘と経験で行われてきた．サーバ管理者はリアルタイムにサーバを観測し，正常状態に保つ義務がある．ENMA を用いる事でサーバシステムのリアルタイムな観測は可能である．管理者の目的は，サーバを計測する事でシステムに異常が発生した場合に，直ちに復旧する事である．サーバに対して正しい処置を施すにはサーバシステムの状態をモデル化する必要がある．本章では，サーバシステムの状態について定義し，状態遷移をモデル化した．また，実際に運用されているサーバに対してモデルを適用し，管理者に対するサーバの管理方法について議論する．

3.1.　まえがき

インターネットは急速に成長し，社会基盤となっている．特に World-Wide Web(WWW) はインターネットの中でも最も普及しているサービスの一つである．したがって，サーバ管理者にとって常に安定したサービスを提供する事が重要である．サーバシステムを停止させずに管理するために，これまで多くのツールやシステムが開発されてきた．しかし，それらのツールを用いたとしてもサーバの管理には管理者の勘と経験を必要としている．

そこで我々は，サーバ管理者のために WWW サーバの性能をリアルタイムに計測するためのツール (ENMA [30]) を開発した．ENMA を用いる事で WWW サーバのいくつかの性能指標を測定できる．そして，得られた性能指標を視覚的に表示できる．しかし，サーバ管理者は WWW サーバの性能計測を目的としているわけではない．サーバ管理者の目的は観測を通じてサーバシステムの状態を知り，その状態に対して処置を施す事である．

この様なサーバの状態を定義するために，サーバの性能指標となる項目を変化させ性能評価実験を行った．その結果から，サーバの状態がどのように変化するかを推測し，サーバの状態遷移モデルを提案した．そして，提案した状態遷移モデルに対応する管理者の行動について議論した．実験の結果から，より詳細な状態遷移を知るためには内部情報の抽出が不可欠である事が明らかとなった．

3.2 節では WWW サーバの性能計測方法を述べる．そして我々の戦略を述べる．3.3 節では，WWW サーバの性能指標について議論する．3.4,3.5,3.6 節では性能指標を用いたサーバの状態の定義，その状態を用いた管理の方法を述べる．3.7,3.8 節で今後の課題とまとめを述べる．

3.2. 性能計測手法

2.2 節において，すでに我々は WWW サーバの性能計測手法についていくつか述べた．本節では ENMA システムと従来のネットワークモニタリング手法について比較・検討する．

3.2.1 ネットワークモニタリング

WWW サーバが接続されているネットワークセグメントのパケットを観測する事で，WWW サーバの性能を知る事ができる．ネットワークモニタリングは WWW サーバシステムに影響を与えない．しかし，観測したパケットを解析するアプリケーションが必要である．一般的にパケット観測では `tcpdump` [14] が用いられている．tcpdump は汎用モニタリングプログラムであるので，観測結果を得るためには tcpdump によって生成された出力を加工するプログラムを開発する必要がある．

また，MRTG [20] はルータやスイッチに対して SNMP を用いるトラヒック観測ツールである．サーバ管理者はサーバが接続されているネットワーク上のスイッチやルータに対して MRTG を用いる事でトラヒックを観測する事ができる．MRTG は観測粒度が大きいため，1 週間や 1ヶ月といった長期間の観測に向いている．

3.2.2 ENMA システム

ENMA は WWW サーバの性能計測のために開発されたアプリケーションである．その特徴はパケットモニタリングを用いている事である．ENMA は WWW サーバの様々な性能指標を観測できる．例えば，それは同時コネクション数，1 秒間のコネクション処

理数，トラヒック量などである．また，ENMA はリアルタイムに視覚化する機能を持っている．よって上で述べたような性能指標をリアルタイムに視覚化できる．本論文では，観測粒度を小さくする事が求められている．それは，サーバの処理が滞る時に直ちにトラブルを管理者に報告するという要求があるからである．また，ENMA は HTTP に特化しているため，HTTP 特有のパケットを識別できる．例えは，ENMA はクライアントの HTTP リクエストに対するサーバの応答時間を知る事ができる．さらに，ENMA は観測粒度を容易に変更できる．

3.3. サーバシステムにおける性能指標

本節では，サーバの状態を定義するための性能指標について述べる．

3.3.1 節で，サーバの HTTP 通信での TCP コネクション処理モデルについて述べる．3.3.2 節で，サーバを外部から観測する事による性能指標について述べる．

3.3.1 サーバ内部でのコネクション処理の流れ

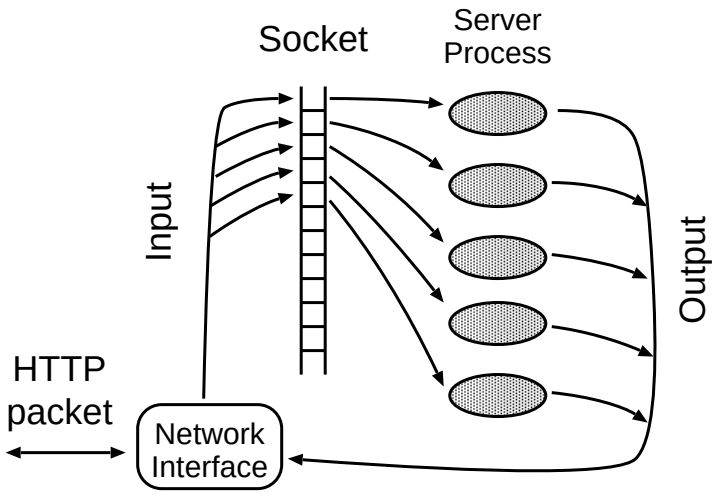

図 3.1 サーバ内部の処理の流れ

図 3.1 はサーバ内での TCP コネクションの処理の流れを示している．第 1 段階では，SYN パケットがクライアントから到着する時，コネクション確立のためにソケットが確保される．第 2 段階では HTTP リクエストに対して応答するためにコネクションからの

データをサーバアプリケーションが処理する．これから，ソケット数とサーバプロセス数がサーバの性能指標として重要である．

3.3.2　サーバ外部からの計測

サーバ外部から計測を行う場合，コネクション処理率，コネクション到着率，現在処理中のコネクション数など，いくつかの性能指標が要求される．

これらの性能指標について以下で述べる．

- 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．

3.4.　サーバ状態定義のための実験

本節では，ベンチマークを用いて性能指標に基づいたサーバの状態を決定するための実験について述べる．この実験では，ディスク I/O のオーバヘッドを取り除くために，固定 URL で HTTP リクエストを生成する設定で実験した．先に述べた性能指標に違いを持たせ，サーバを設定し，ベンチマークシステムによって実験した．本実験の目的は，サーバの飽和状態を意図的につくり出し，その時のサーバ状態を計測し解析する事である．

3.4.1　実験

図 3.2 に本実験のネットワーク構成図を示す．図 3.2 に示すように実験には，サーバ，クライアント，モニタ，ダミーネット，視覚化の 5 台の計算機を用いた．クライアントは

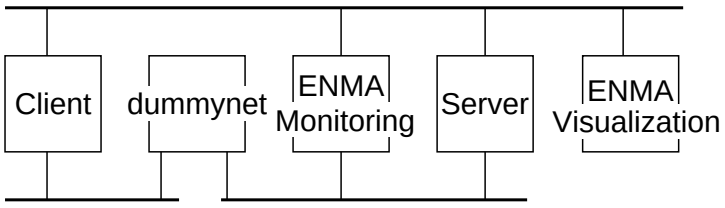

図 3.2 実験ネットワーク構成図

サーバに対して HTTP リクエストを生成する．モニタはサーバへ流れるパケットを観測する．ダミーネットは WAN 環境をエミュレートするために 100 ミリ秒の遅延を挿入する．この遅延の値はサーバ管理者の経験から導かれた値である．サーバ，クライアント，モニタ，視覚化は FreeBSD3.2 上で動作している．ダミーネットは picoBSD [31] 上で動作している．サーバプログラムは apache 1.3.9 [29] でクライアントプログラムは httperf 0.6 [23] である．

3.4.2 結果

表 3.1 Configuration Parameter

	Part	No. of sockets	No. of processes (start/min/max)	request rate
1	1	1064	5/10/150	80
	2	16424	5/10/150	400
2	1	16424	5/10/150	700
	2	16424	5/100/150	700
3	1	16424	5/100/256	1000
	2	16424	128/100/256	1000
	3	16424	256/100/384	1000

表 3.1 にカーネルパラメータと apache の設定値を示す．実験 1 では，WWW サーバシステムの最大ソケット数に着目して実験した．実験 2 では，WWW サーバシステムの最小および最大プロセス数を変化させて実験した．実験 3 では，サーバプロセス数と CPU アイドル時間との間のトレードオフについて議論した．

3.4.2.1　実験 1

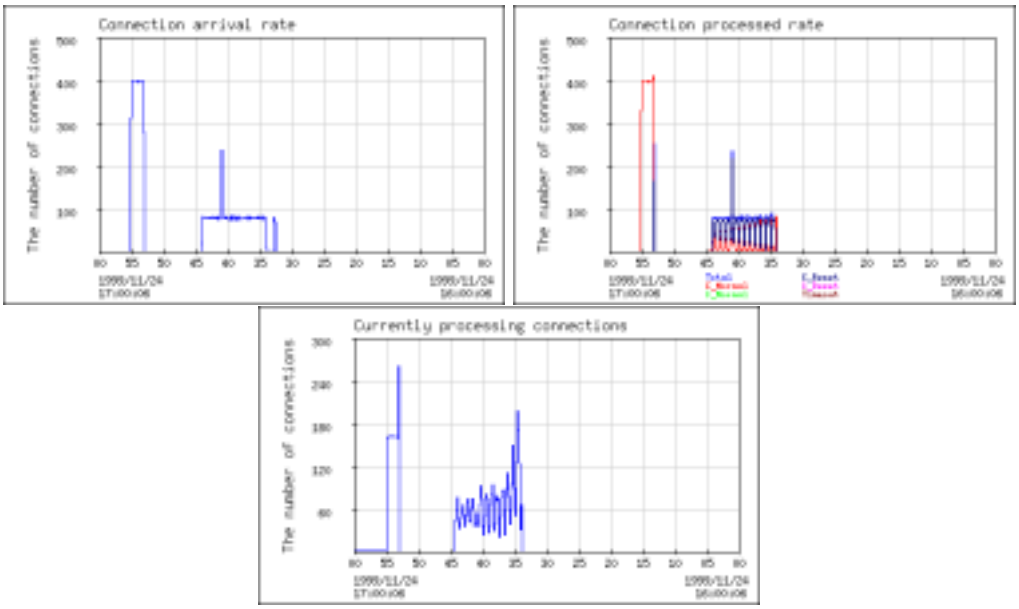

図 3.3　最大ソケット数を変化させた結果

実験 1 は表 3.1 に示すように実験 1-1 と実験 1-2 から構成されている．実験 1-1 は 16:34〜16:44 の間で，OS の設定は FreeBSD の GENERIC kernel を用い，WWW サーバの設定は apache の default の設定ファイルを用いている．HTTP リクエストレートは表 3.1 に示すように 80 リクエスト/秒である．実験 1-2 は 16:53〜16:55 の間で，カーネル内のソケット数を 1064 から 16424 に増加させた．apache の設定ファイルは実験 1-1 と同様のものを用いた．実験 1-2 におけるリクエストレートは 400 リクエスト/秒である．実験 1-1 では，図 3.3 コネクション到着率は安定している．しかし，コネクション処理率と現在処理中のコネクション数は図 3.3 に示すように安定していない．しかし，実験 1-2 ではコネクション到着率，コネクション処理率，現在処理中のコネクション数，すべて安定している．

この 2 つの結果から，カーネル内部のソケット数はサーバシステムの性能指標として重要であることを示した．

3.4.2.2　実験 2

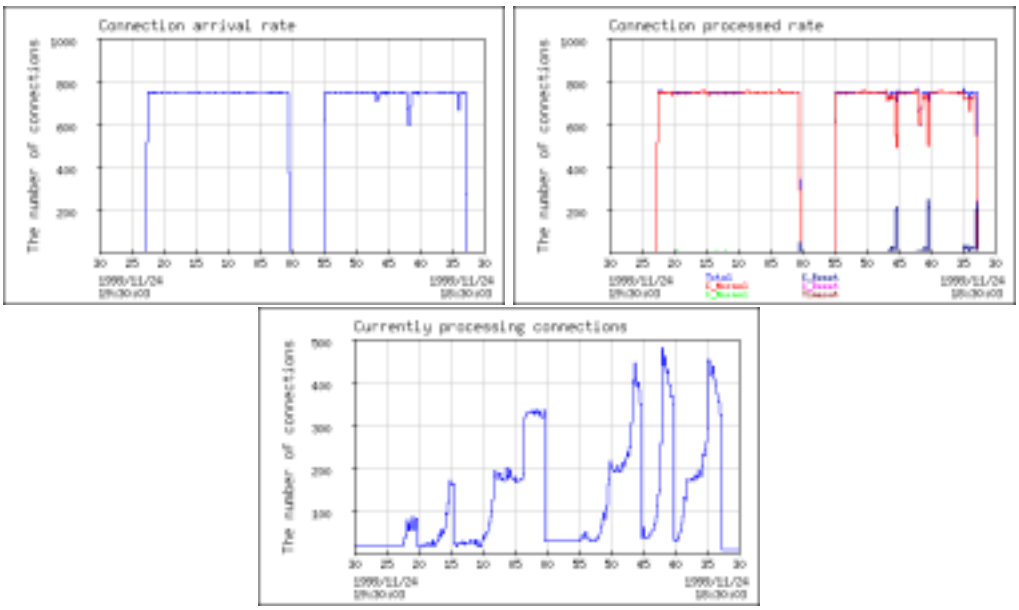

図 3.4 プロセス数を変化させた結果

実験 2 は表 3.1 に示すように実験 2-1 と実験 2-2 から構成されている．実験 2-1 は 18:33〜18:55 で，カーネルパラメータとして，16424 ソケットとしている．apache の設定は，最大サーバプロセス数 150 としている．リクエストレートは 700 リクエスト/秒である．実験 2-2 は 19:00〜19:23 で，表 3.1 に示すように最小サーバプロセス数を 100 としている．

図 3.4 の実験 2-1 に示される 18:40〜18:45 の間でコネクション処理率にスパイクが現れている．これから，この時間においてサーバが飽和したと考えられる．

さらに，現在処理中のコネクション数は飽和状態の時，急速に上昇している．これは，apache が 18:40〜18:45 の間に fork システムコールを発行し，その実行の影響からサーバ内部でのコネクション処理が滞ったためであると考えられる．

実験 2-2 では，apache の設定において最小サーバプロセス数が変更されている．図 3.4 において，コネクション到着率，コネクション処理率が安定している．図 3.5 に示すように，実験 2-1，2-2 において実験中の CPU アイドル時間は約 55%である．この値はまだ余力があると言える．したがって，このシステムが更に多くの HTTP リクエストを処理できる可能性がある．

この実験 2 から，最大および最小サーバプロセス数はサーバシステムの性能指標とし

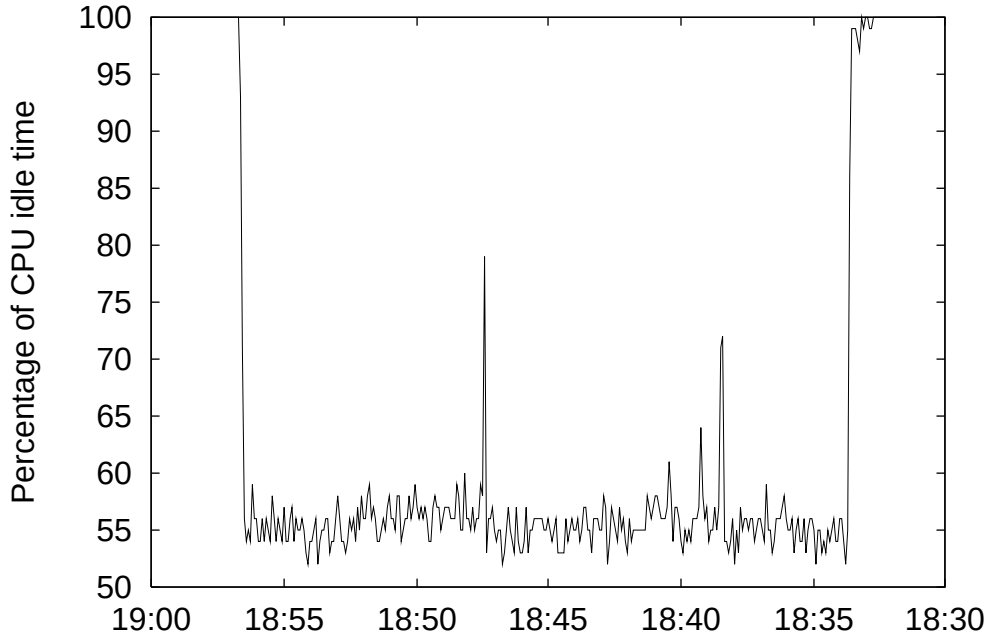

図 3.5　CPU アイドル時間

て重要な項目であると言える．

3.4.2.3　実験 3

この実験 3 は表 3.1 に示すように実験 3-1, 実験 3-2, 実験 3-3 から構成されている．実験 3-1 は 20:32〜20:48 の間で表 3.1 に示すように，カーネルパラメータとして 16424 ソケット，apache の設定として最大 256 プロセス，最小 100 プロセスである．リクエストレートは 1000 リクエスト/秒である．実験 3-2 は 20:52〜21:07 の間で，16424 ソケット，開始サーバ数 128 プロセス，1000 リクエスト/秒である．それ以外の設定は実験 3-1 と同様である．実験 3-3 は 21:15〜21:21 の間で，16424 ソケット，最大 384 プロセス，最小 100 プロセス，開始サーバ数が 256 プロセスである．リクエストレートは 1000 リクエスト/秒である．

図 3.6 に示すように，サーバは実験 3-1 において 20:32〜20:33，20:37〜20:40，20:44〜20:47 の時間帯において飽和状態となっている．この飽和状態の理由は図 3.7 に示さ

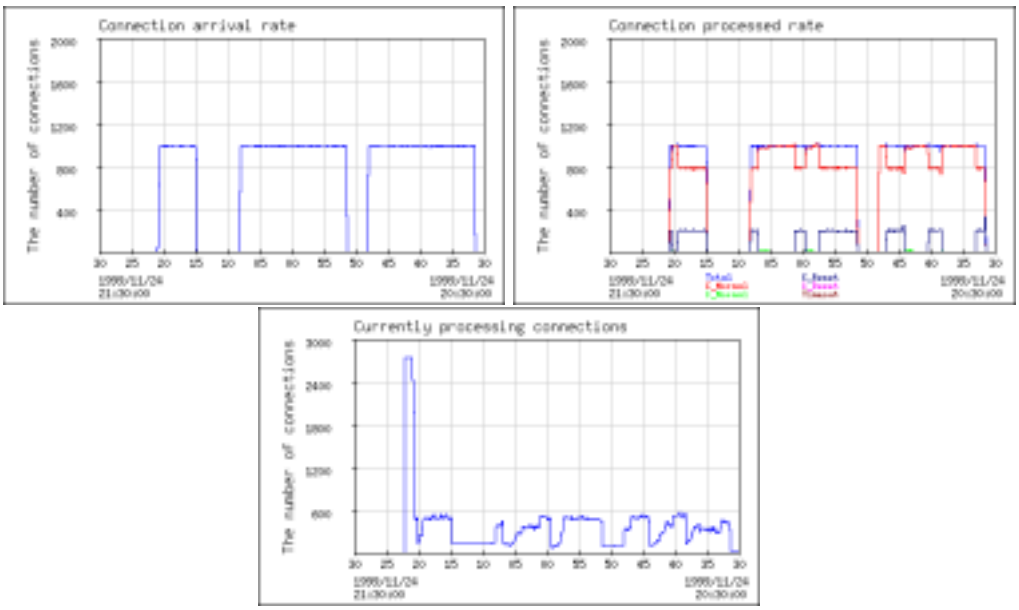

図 3.6 システム能力不足の結果

れる CPU アイドル時間が不十分である事が考えられる．図 3.7 に示すように，20:32〜20:33，20:37〜20:40，20:44〜20:47 の時間帯では CPU アイドル率が約 30%である．この値ではシステムの余力は不十分であると考えられる．

実験 3-2 においても同様に飽和状態が確認できる．これらの二つの実験よりサーバシステムの設定を変更したとしても性能に変化が見られなかった場合，その時のリクエストレートがサーバシステムの性能限界であると考えられる．この様な性能限界が見られる場合，サーバ管理者はシステムの置換えを考えるべきである．

図 3.6 の 21:21 において現在処理中のコネクションが急上昇している．これは実験中にシステムパニックが生じたためである．

3.4.3 考察

3.4.2 節の実験結果から，apache サーバシステムを運用する場合，以下の事が明らかとなった．

- 実験 1 の結果から，カーネルパラメータである，ソケット数を変更する事で，サーバシステムの性能向上が見られる．

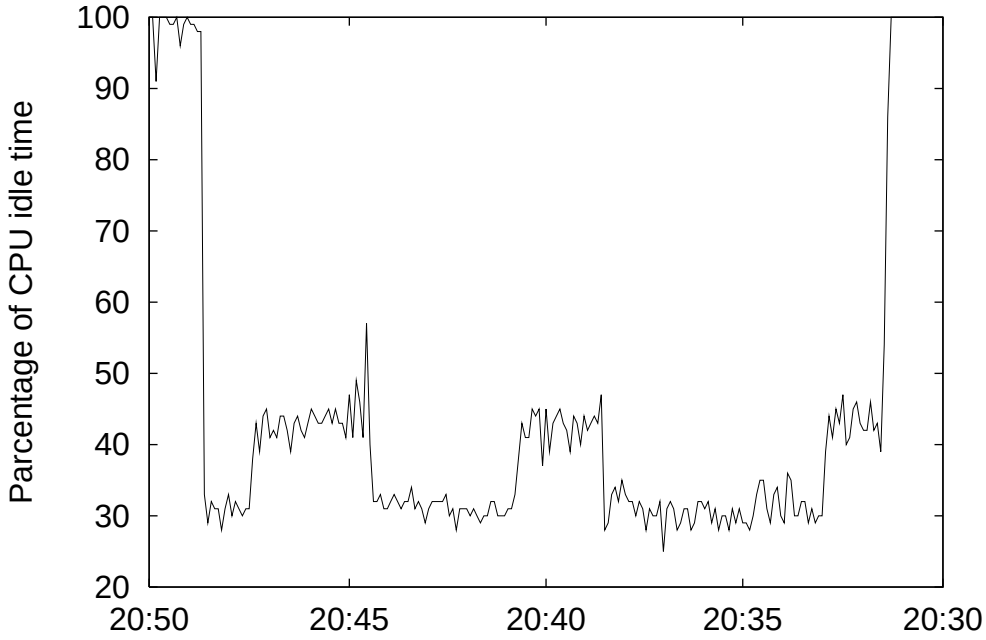

図 3.7　Percentage of CPU idle time

- 実験 2 の結果から，apache のパラメータである最大および最小サーバプロセス数を変更する事で，サーバシステムの性能向上が見られる．
- 実験 3 の結果から，サーバの最大性能を維持する事と，サーバシステムの安定運用にはトレードオフが見られる．実験 3 に示されるように，リクエストレートが上昇する時 CPU の処理能力とサーバプロセス数の間ではトレードオフ関係が見られた．

以上より，我々はサーバの状態を推測するための重要な性能指標 (最大ソケット数，サーバプロセス数，CPU の処理能力) を得る事が出来た．

3.5.　サーバの状態定義とシステム管理手法

本節では，3.4.3 で述べた考察を基に，サーバの状態に対する管理方法について述べる．我々は，サーバの状態に対して，以下の 3 つの状態を定義した．

1. 通常状態

   我々は，トラブルなしにクライアントからの HTTP リクエストをサーバが応答することを通常状態と定義する．サーバ管理者はこの状態に WWW サーバを維持するべきである．

2. 飽和状態

   いくつかの TCP コネクションがサーバで正しく処理されていない状態を飽和状態と定義する．管理者はサーバシステムがこの状態に留まる事を避けるべきである．

   我々は以下に示すようにこの飽和状態をさらに 3 つの正確な状態に区別する．

   - ソケット不足
   - サーバプロセス不足
   - システム能力不足

3. 停止状態

   停止状態はサーバシステムが停止している状態と定義する．

   我々はこの状態を以下に示す更に 2 つの状態に区別する．

   - 再起動
   - クラッシュ

図 3.8 に我々の提案する状態遷移モデルを示す．図 3.8 は 3.4.2 節の実験に基づいている．3.4.2 節の実験 1 では，図 3.8 の通常状態からソケット不足への状態遷移 (1) が明らかとなった．もし WWW サーバがソケット不足状態になったとすると，サーバ管理者はサーバシステムの最大ソケット数を増加させる必要がある．例えば，サーバシステムが一般的な BSD Unix システムであるなら，カーネルパラメータの 1 つである *MAXUSER* を増加させることで，最大ソケット数を増加できる．3.4.2 節の実験 2 では，図 3.8 の通常状態からサーバプロセス数不足への状態遷移 (2) が明らかとなった．WWW サーバがサーバプロセス数不足になったとすると，サーバ管理者は最大および最小サーバプロセス数を増加させるべきである．例えば，apache サーバシステムなら *MaxClient* と *MaxSpareServers* のパラメータを増加させる事で，最小および最大サーバプロセス数を増加できる．3.4.2 節の実験 3 では，図 3.8 の通常状態からシステム能力不足への状態遷

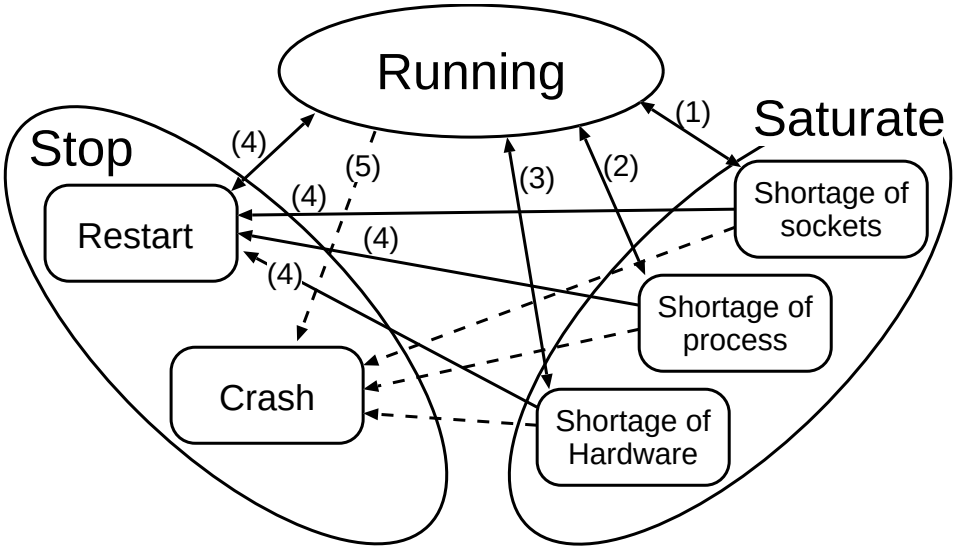

図 3.8　State Transition in the Server

移 (3) が明らかとなった．サーバシステムがシステム能力不足状態になったと判断したなら，管理者はシステムのボトルネックを調査しなくてはならない．もし，CPU 能力が不十分であるなら，より高速な CPU に交換するべきである．また，メモリ量が不十分であるなら，メモリを増強しなくてはならない．通常状態，ソケット不足状態，プロセス数不足状態，システム能力不足状態から再起動状態への状態遷移 (4) は，大抵サーバ管理者によって行われる．再起動後は，サーバシステムは通常状態となる．3.4.2 節の実験では，我々は通常状態からクラッシュ状態への遷移が 1 度だけ生じた．図 3.8 以外の状態遷移も存在するかも知れない．それらの遷移を発見する事は今後の課題である．

3.6.　実際のサーバ観測の例

我々は，実際に運用されている WWW サーバの状態を解析するために，ENMA を用い計測した．

計測した WWW サーバは「第 81 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である．

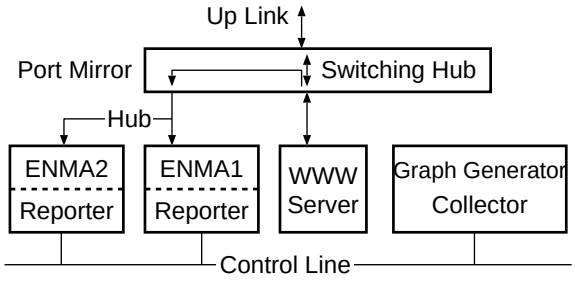

図 3.9 ネットワーク構成

表 3.2 計算機構成

	WWW server	ENMA host 1	ENMA host 2
OS	SunOS 5.6	SunOS 5.6	FreeBSD 3.2
CPU	Ultra SPARC II 300MHz (x2)	Ultra SPARC II 300MHz (x2)	PentiumII 400MHz
Memory	512MBytes	256MBytes	128MBytes
Application	Apache 1.3.6	ENMA (19990823)	ENMA (19990823)

3.6.1 Environment

図 3.9 は今回の計測でのネットワーク構成図である．WWW サーバや観測に用いた計算機のスペックおよび性能を表 3.2に示す．ENMA を用いて WWW サーバを計測するために，図 3.9 に示すようにスイッチのポートミラー機能を用いて，パケットをコピーした．我々は ENMA のトラブルによって計測不可能となる状態を避けるために，1 台の WWW サーバに対して 2 台の ENMA ホストを準備した．各々の ENMA ホストは制御用セグメントを用いて解析ホストへリアルタイムデータを送信した．制御用セグメントを用いる理由は，WWW サービス側のスイッチに対して負荷を増加させないためである．解析ホストは WWW サーバの現在の状態を表示するグラフを生成する．

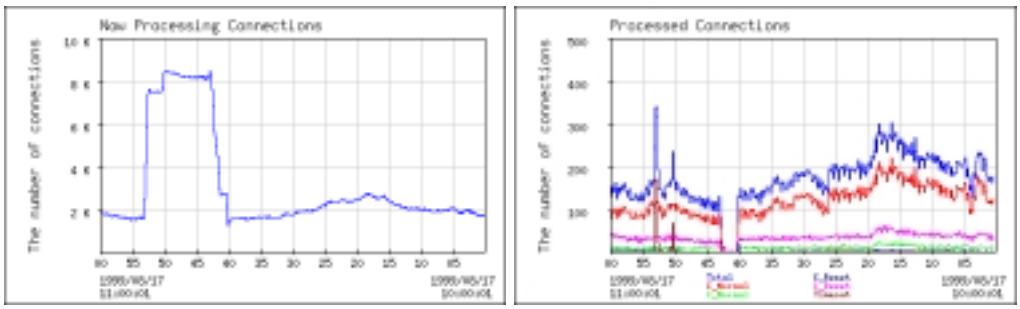

図 3.10　WWW サーバの再起動

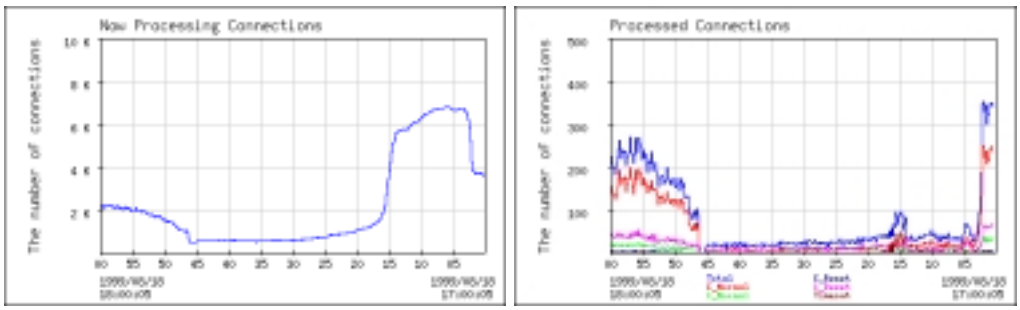

図 3.11　WWW サーバの飽和状態

3.6.2　事例研究

3.6.2.1　サーバの再起動

図 3.10 は再起動時のグラフである．図 3.10 の 8 月 17 日 10:40 において現在処理中のコネクションが急速に上昇し，処理したコネクション数が 0 まで減少している．この 10:40 において我々はサーバ管理者が何らかの理由によりシステムを再起動したと考える．

3.6.2.2　サーバの飽和状態

図 3.11 は WWW サーバの飽和状態を示している．図 3.11 の 8 月 18 日 17:02 において，現在処理中のコネクションが上昇し，処理したコネクション数が減少している．17:14 頃に処理したコネクションにおいてタイムアウトが観測されている．この時，現在処理中のコネクションが急速に減少している．17:02 から 17:45 において，処理したコネクションは 0 にはなっていない．これは，サーバ能力は低下しているがサービスはかろうじて維

持している事を示す．17:02 において，サーバ管理者はシステムメッセージによってシステムのメモリ量が不足していると判断した．そしてスワップ領域を増加した後，17:45 にシステムを再起動した．今回の飽和状態では，管理者はサーバシステムのムメッセージを通してシステム能力不足 (メモリ不足) と判断した．しかし，管理者はサーバシステムのパラメータを調査するべきであった．これは，最大サーバプロセス数は 1536 で CPU アイドル率は 40〜50 パーセントを示していたからである．この状態では，我々は最大プロセス数が多過ぎる事によってメモリ量の不足が起きたのではないかと判断する．したがって，サーバ管理者はシステム能力に一致する最大サーバプロセス数を設定しなくてはならない．

3.7. 今後の課題

本節では，ENMA を用いたサーバの状態解析における今後の課題について述べる．

3.7.1 飽和状態の検出

サーバ管理者の立場では，サーバシステムの処理が飽和する事を避けるのが最善である．しかし，サーバシステムが飽和した場合，その状態をすぐさま検出する必要がある．これまでの実験では ENMA の出力結果を観測する事でサーバの状態を判断してきた．しかし，その判断には明確な閾値が存在するわけではない．今後はこのサーバシステムが飽和する条件となる明確な閾値を調査する必要がある．

3.7.2 管理者への警告の方法

サーバにおいてトラブルが生じた時，管理者に警告する方法を考えなくてはならない．例えば，WWW サーバが飽和状態になった時，ENMA が管理者にトラブルをメールする事が考えられる．

3.7.3 サーバ内部状態を得るシステムの構築

WWW サーバの内部状態をえるためには，ソケット利用率 (最大数，現在使用している数)，サーバプロセス数，メモリ消費量，CPU アイドル時間などが観測できなくてはならない．今回の実験において，我々は `vmstat` を用いて計測した．しかし，このプログラムは上で述べた性能指標を得るには不十分である．したがって，我々は上で述べた性能指標を得るために新たなサーバ内部情報抽出システムを構築しなくてはならない．

3.8. おわりに

WWW はインターネットにおいて重要なサービスとなった．WWW サービスを維持し，サービスの中断期間をなくす事はシステム管理者にとって重要である．しかし，サーバ管理は管理者の勘と経験を必要とする．これは，WWW サーバの状態を知る方法が確立されていないからである．そこで我々は WWW サーバの状態をリアルタイムに観測するために ENMA を開発した．ENMA を用いて，WWW サーバの状態について定義し，状態遷移モデルを提案した．そして，実際に運用されているサーバに対し ENMA を用い，その状態を推測した．さらに，管理者の行動について考察した．

第 4 章　カーネルモニタリングシステムの実装

サーバシステムの計測手法はこれまでに様々な手法が提案，実装されている．これまで述べてきたパケットモニタリングによる手法は，サーバシステムに負荷を与えずに計測できる手法として有効である．しかし，サーバ管理者がサーバシステムのボトルネックを調査する場合，パケットモニタリングだけでは不十分である．サーバシステムのボトルネックを調査する場合，カーネルモニタリング手法を用いて調査する．これまで，カーネルモニタリング手法は，サーバシステムに負荷を与えるため有効でないと考えられてきた．そこで本章ではサーバシステムの性能に影響の少ないカーネルモニタリングシステムの提案，実装，評価について述べる．

4.1.　あらまし

情報化社会の急速な発展により，インターネットの利用者が急速に増加している．中でも World Wide Web(WWW) は携帯電話や PDA などからも利用できるほど，社会に広まっている．このようなインターネットユーザの急速な増加により，WWW サーバの数も同様に増加している．ユーザの増加により，より良いサービス品質が望まれる．サーバ数の増加，サービス品質の向上によりサーバ管理者の負担は増大している．

このような要求があるにもかかわらず，サーバ管理者は自身の経験とそれに基づいた勘を頼りにサーバを管理している．その理由は，サーバの性能を運用中に容易に知る事が出来ないからである．そこで我々は ENMA を開発し，運用中にサーバの性能を知るためのシステムを構築した．しかし，次に問題となった項目は，サーバシステムのボトルネックの調査である．ENMA はパケットモニタリングを基に計測するため，サーバシステム内部の出来事を観測できない．これまで，サーバシステムの内部情報を抽出するにはカーネルモニタリングを行ってきた．しかしカーネルモニタリングは負荷が高く実用的でないと考えられてきた．そこで我々は，サーバシステムの性能に影響をほとんど与

えないカーネルモニタリングシステム (rep2) を設計，実装した．そしてその有効性を示すために実際に運用されているサーバシステムに適用した．

本章では ENMA と rep2 を組み合わせ，サーバ外部および内部の観測結果から，サーバ管理者がとるべき行動について議論した．

4.2. 従来システム

サーバシステムの内部情報としては，計算機依存部分 (CPU や memory に関する使用量など) および，ネットワークに関する指標 (ファイルディスクリプタ，トラヒック，パケット数など) などがある．

4.2.1 vmstat

仮想メモリの統計情報を表示するプログラム．リアルタイムにディスプレイに表示する．CPU 利用率 (アイドル率，ユーザ使用率，カーネル使用率)，メモリ利用率，ページの状況などがわかる．欠点としては，タイムスタンプが残らない，計測と表示をサーバシステム内部で行うためログを残すことが困難であるなどが挙げられる．

4.2.2 netstat

ネットワークシステムの統計情報を表示するプログラム．オプションの指定により，様々な情報を得ることが出来る．ルーティング情報，ネットワークインタフェース情報，プロトコル別統計情報，ネットワークメモリバッファの使用量，入力パケット数および出力パケット数，エラーパケット数などを得ることが出来る．netstat も vmstat 同様，タイムスタンプが残らず，ログを残すことが困難である．

4.2.3 ps

プロセスに関する情報を表示するプログラム．オプションの指定により，プロセス ID，親プロセス ID，CPU 利用率，実行開始時間など，様々な情報を得ることが出来る．ps は，システムで動作しているプロセスの状態のスナップショットを得るプログラムである．したがって，時間遷移のログを残すことは出来ない．

4.3. カーネルモニタリングシステムの設計

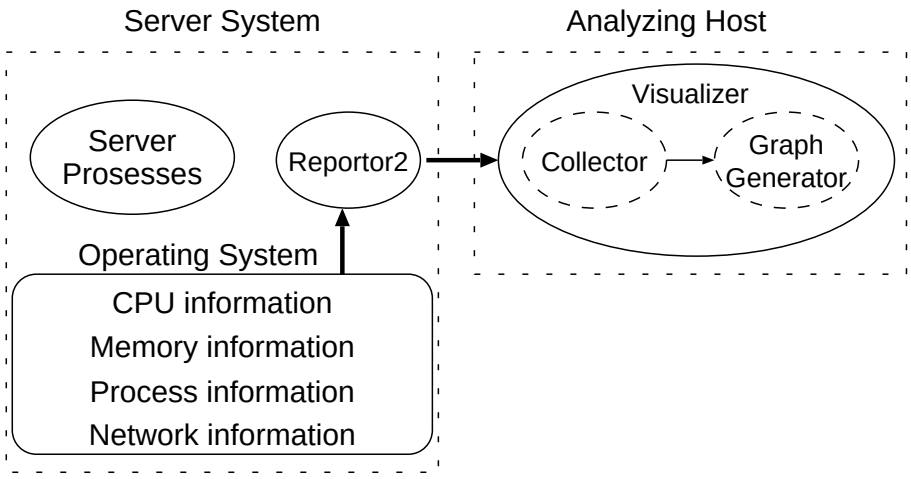

図 4.1 カーネルモニタリングシステム

本節ではカーネルモニタリングシステムの設計について述べる．本システムを rep2 とした．カーネルモニタリングシステムは OS 依存の実装である．可能な限り OS 依存部分を減らすために，我々は kvm インタフェースを用いて実装した．rep2 は二つの要素から構成されている．カーネルモニタデーモン (rep2) とデータ収集プログラム (collector) である．(図 4.1)

従来システムの問題点は，ログ出力とモニタリングを同一ホストで行っていたことが挙げられる．これにより，ログ出力のオーバヘッドが大きくなり計測自体が高負荷なものとなっていた．そこで本論文ではログを別ホストで記録することによりこの問題点の解決を図る．

4.3.1 カーネルモニタデーモン (rep2)

rep2 は collector からのリクエストに応じて，カーネル情報をネットワークに出力する．rep2 では以下の情報について観測する．

- 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．

- 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．

4.3.2 データ収集プログラム (collector)

データ収集プログラム (collector) は 10 秒間隔で rep2 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．サーバシステムの負荷を軽減しリアルタイムに解析するためにはサーバ外部での解析が不可欠である．rep2 は collector を用いることでデータを外部ホストへ出力し，外部ホストによる解析を可能とした．

4.4. カーネルモニタリングシステムの実装

UNIX プラットホームで動作するカーネルモニタリングシステムを目標としてシステムを実装した．rep2 および collector は C 言語で実装した．

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．

4.5. 実証実験

本節では，rep2 を起動することによるサーバへの負荷を調査する．そして，実際に運用されているシステムに適用した例について述べる．

4.5.1 rep2 による負荷の影響

本節では，我々の実装した rep2 がサーバの性能にどの程度の影響を与えるのか確認する．rep2 を使用した場合と使用しない場合におけるサーバシステムの応答時間を計測する．応答時間がどの程度劣化するかで rep2 のサーバに与える影響を調査する．

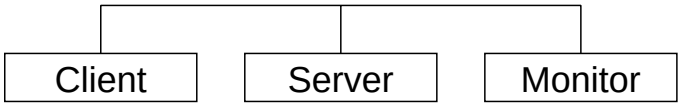

図 4.2　ネットワーク環境

表 4.1　実験構成

ホスト	OS	CPU	メモリ	アプリケーション
Client	FreeBSD 4.1.1	Pentium III 700MHz	256M Bytes	httperf 0.8
Server	FreeBSD 4.1.1	Pentium III 700MHz	256M Bytes	apache 1.2.12
Monitor	FreeBSD 4.1.1	Pentium III 700MHz	256M Bytes	enma

図 4.5.1 に，実験で用いたネットワーク環境を示す．表 4.1 に使用したソフトウェアおよびハードウェアを示す．表 4.1 に示すように，クライアントは httperf 0.8 を用い，リクエストレートは 300，総コネクション数はそれぞれ 300000 コネクションとした．サーバは apache 1.3.12，応答時間の計測のために ENMA を用いた．OS は全て FreeBSD 4.1.1 である．

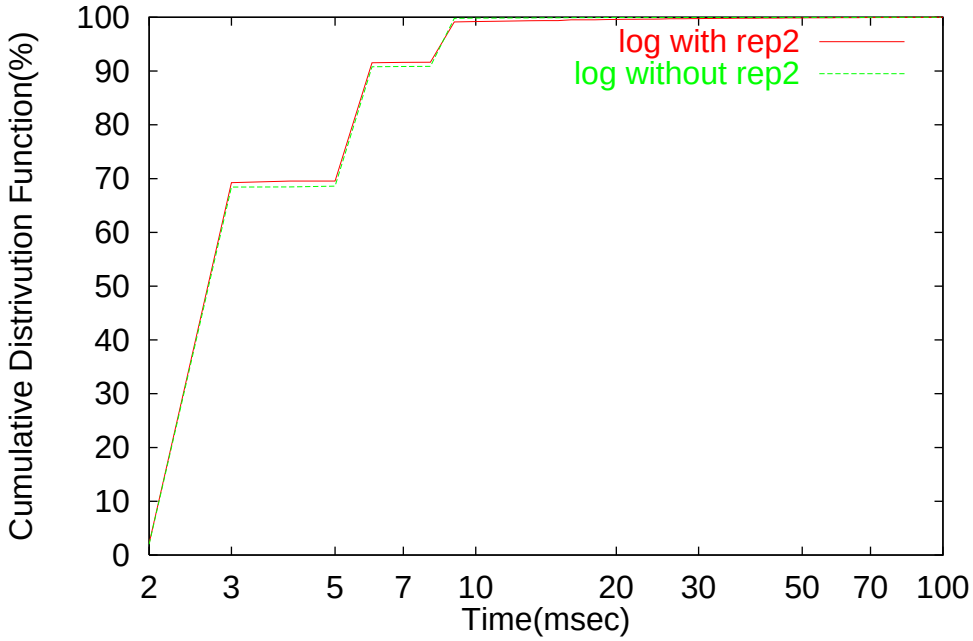

図 4.3 応答時間の累積分布

図 4.3 に測定結果を示す．図 4.3 は rep2 を用いた場合と用いない場合の両方の場合における応答時間の累積分布である．横軸に応答時間，縦軸に累積分布をとっている．図 4.3 からわかるように rep2 を用いる場合と用いない場合で，ほとんど差は見られない．これから，我々の提案した rep2 はサーバの性能にほとんど影響を与えないとみなせる．

4.5.2 実際のシステムへの適用例

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．

4.5.2.1 システム構成

図 4.4 にネットワーク構成図を示す．図 4.4 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．

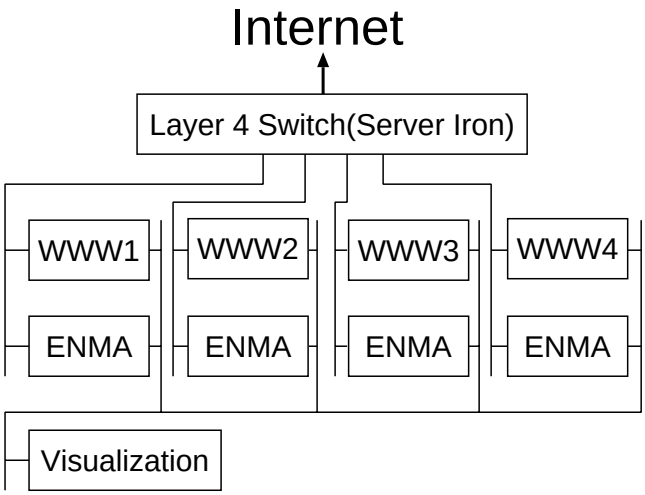

図 4.4　ネットワーク構成図

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 4.2 に示す．表 4.2 に示すように，WWW1 はapache を用い，WWW2〜WWW4 では，chamomile [32] 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 4.4 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．

4.5.3　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．

4.5.3.1　ENMA による観測結果

図 4.5 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．

表 4.2 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

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．

4.5.3.2 rep2 による内部観測

- プロセス数

  図 4.6 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 4.7 に apache のメモリ量とページフォルト数を示す．図 4.7 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 4.8 にメモリ量とプロセス数を示す．図 4.8 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，

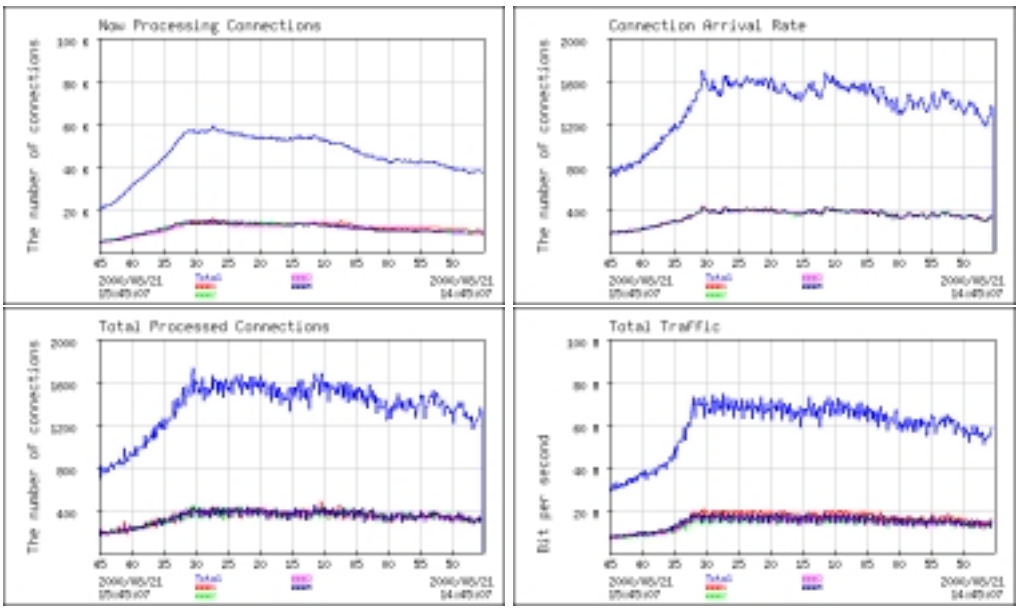

図 4.5　4 台合計の結果

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 4.9 に WWW1 および WWW2 のコンテキストスイッチの遷移を示す．図 4.9 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．

- システムコールについて

  図 4.10 に WWW1 および WWW2 のシステムコール発行数の遷移を示す．図 4.10 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

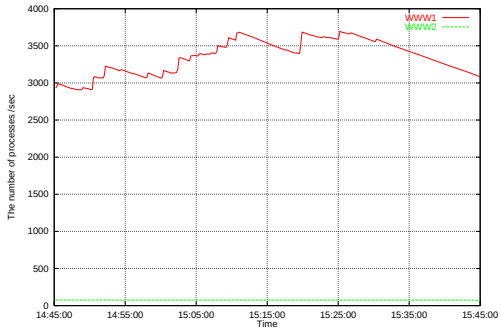

図 4.6 プロセス数の遷移

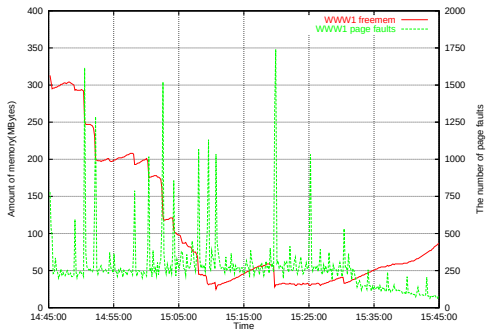

図 4.7 メモリ量とページフォルト数の遷移

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．

- 割り込みについて

  図 4.11 に WWW1 および WWW2 の割り込み発行回数の遷移を示す．図 4.11 に示すように，WWW1 と WWW2 ともに同様の値を示している．リクエスト到着数がほぼ同じなので，パケット到着による割り込みの発生も同じになったと考えられる．

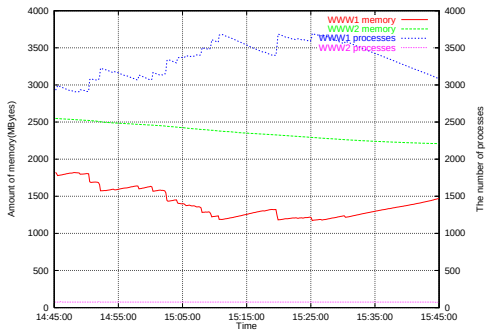

図 4.8　メモリ量とプロセス数の遷移

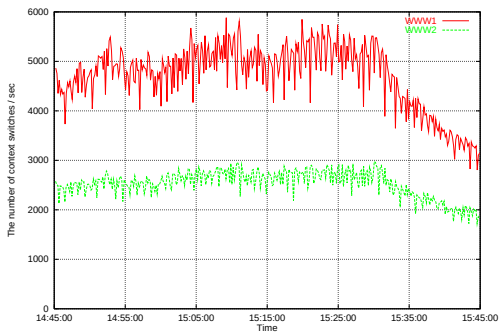

図 4.9　コンテキストスイッチの遷移

- CPU 利用率について

  図 4.12 に WWW1 および WWW2 の CPU 利用率を示す．図 4.12 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 [11] によると，カーネル利用率が全体の 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．

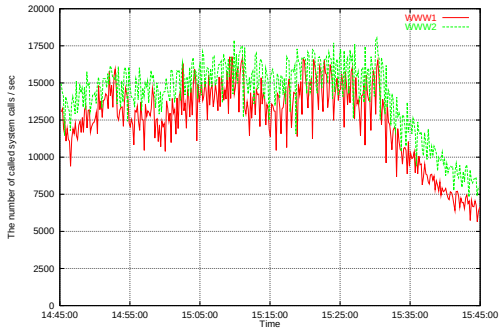

図 4.10 システムコール数の遷移

図 4.11 割り込み数の遷移

4.5.3.3 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．

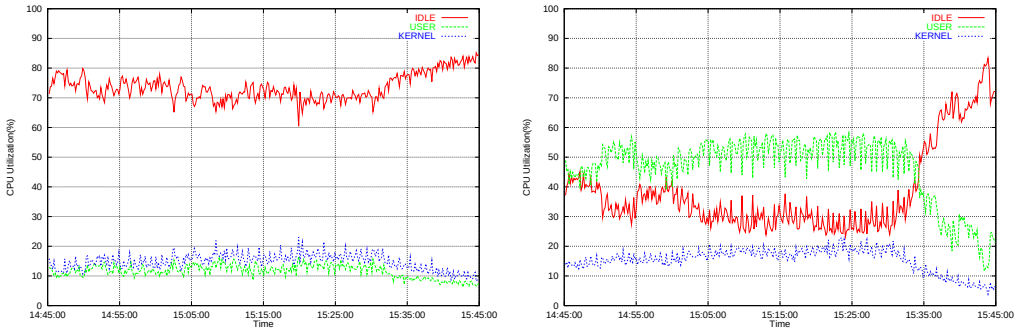

図 4.12　CPU 利用率の遷移

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．

4.6.　外部で記録する問題点

rep2 では，サーバシステム内部ではなく，外部でデータを記録することによる問題点が生じる．例えば，ネットワークトラブルが発生した場合，rep2 による計測は出来ない．この問題点を解決するための方法として，カーネルモニタリング用に専用のネットワークを構築することが考えられる．さらに，ENMA のリアルタイム計測と併用することで，計測専用のネットワークの効率を高めることが出来る．

4.7.　まとめ

WWW サーバ内部のボトルネックを調査する場合，カーネルモニタリングが不可欠である．これまで，カーネルモニタリング手法は，サーバシステムに負荷を与えるため有効でないと考えられてきた．従来のカーネルモニタリングシステムの問題点はログ出力と計測が同一ホストで行われている事が考えられる．これにより負荷が高く実用的でな

かった．そこで我々は計測とログ出力を異なる計算機で実行できるシステムを構築した．そして，新たなカーネルモニタリングシステムを実際に運用されているサーバシステムに適用し，その有効性を示した．

第 5 章　代理サーバの性能計測

代理サーバの目的の 1 つとして，広域に対するトラヒックの削減があげられる．サイトの規模が大きければ大きいほど，代理サーバの負荷が高くなる．代理サーバの管理者は，サイトの規模に対してどの程度のサーバシステムが必要かを知りたいと考えている．しかし，そのような項目を知るための手段はまだ確立されてない．そこで本章では，パケットモニタによる，代理サーバの性能計測を行う方法を提案し，本手法を用いたシステムの設計と実装を述べ，観測結果を述べる．

5.1.　あらまし

代理サーバの目的の 1 つとして，広域に対するトラヒックの削減があげられる．多くのサイトでは，キャッシングを行う代理サーバに WWW アクセスの中継をさせることにより，トラヒックの削減を実現しようとしている．しかしながら，サイトの規模が大きくなればなるほど，代理サーバの負荷が大きくなる．代理サーバの管理者は，サイトの規模に対してどの程度のサーバシステムが必要かを知りたいと考えている．しかし，そのような項目を知るための手段はまだ確立されてない．したがって，代理サーバの管理者は勘と経験でシステムを構築している．代理サーバの性能評価を行う方法としてログ解析手法やベンチマーク手法があげられる．ベンチマーク手法では，Winsconsine Proxy Benchmark (WPB) [33] や Polygraph [34] と呼ばれるベンチマークシステムが開発され，利用されている．この様なベンチマーク手法では，性能計測を行う際に特別な環境を必要とする．また，この特別な環境を構築するために，多くのコストを必要とする．したがって，運用中の代理サーバをベンチマーク手法で性能計測することは，適切とはいえない．また，ログ解析手法では，代理サーバの出力するログを解析することにより，そのサイトのアクセス傾向や，ヒット率を知ることができる．しかし，管理者が知りたいと考える代理サーバの性能指標を得ることはできない．例えば，管理者は最大，同時に

何本のセッションを処理できるか，もしくは，最大，どれくらいのセッションが到着しているかが知りたい．しかし，これらは，OS 内部の処理であるため，アプリケーション層では計測できない．

この様な問題点を解決するために，我々は ENMA を用いて代理サーバの性能計測を行う．代理サーバの外部からパケットモニタによってサーバの性能計測を行うことは，以下の利点がある．

- 運用中の代理サーバの性能計測を行うことができる．
- 計測システムが代理サーバの性能に影響を及ぼすことがない．

そこで本章では，パケットモニタによる，代理サーバの性能計測を行う方法を提案し，本手法を用いたシステムの設計と実装を述べ，観測結果を述べる．

5.2.　設計および実装

同時にクライアントに対して何本のコネクションを処理しているかを知るためには，正確なコネクションの生存時間を知る必要がある．そこで，図 5.1に代理サーバでのコネクションの流れを示す．

正確なコネクション生存時間を計測するためには，図 5.1における，点 1 から点 6 までの時間を計測すれば良い．我々はこの時間をコネクション継続時間と呼ぶ．また，図 5.1 よりキャッシュヒット時とミス時において，点 4 から点 5 の間の時間に大きな差があると考えられる．この点 4-5 間の時間を応答時間と定義する．さらに，点 5-6 の間をデータ転送時間と定義する．

パケットモニタを行うために，我々は libpcap を用いた．この libpcap は BPF や snoop といったパケットモニタシステムを利用したライブラリである．libpcap を用いることで，汎用性を実現することができる．また，libpcap を用いることでユーザ空間でパケットの処理を行うことができるので，カーネルを変更せずに代理サーバの性能計測を行うことができる．

5.3.　観測例

本節では，ENMA を実際に運用されている代理サーバに適用することで，その有効性を示す．

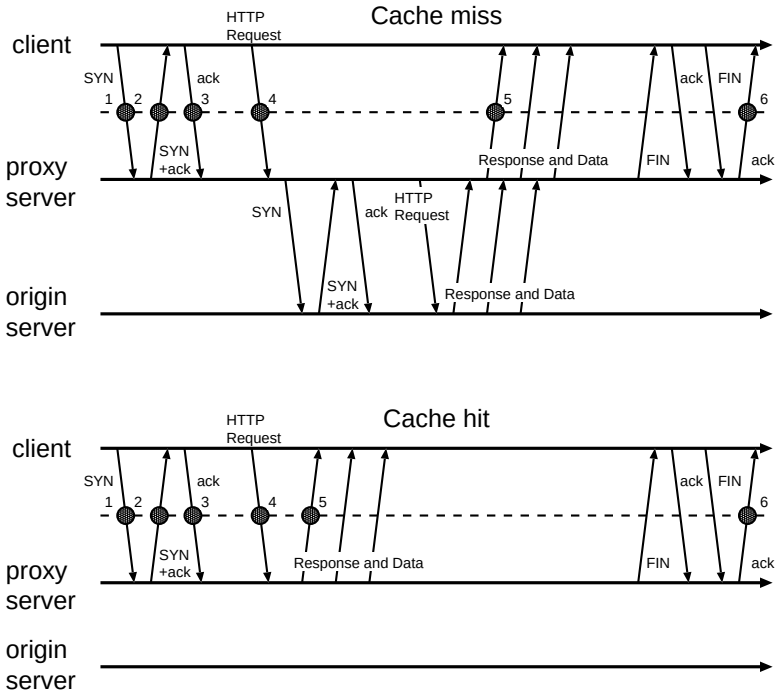

図 5.1 代理サーバにおけるコネクションの流れ

5.3.1 実際の運用例

計測対象として，奈良先端科学技術大学院大学で運用されている代理サーバ (Origin200 R10000 180MHz 4CPU Memory 4096MB) とした．

モニタシステムの計算機は PC(Pentium II 266MHz Memory 64MB) を用いた．図 5.2 に観測環境を示す．図 5.2に示すように，モニタホストと代理サーバを同一セグメントに設置した．

5.3.2 結果および考察

図 5.3に一日における同時コネクション数の遷移を示す．これによると，一日のピークでは 1 秒間に約 370 本のコネクションをクライアントに対して処理していることがわかる．図 **??**は，応答時間，およびデータ転送時間の頻度分布を示している．図 5.4によると，応答時間 1-10 ms の間で かつ データ転送時間 1-10ms 間の領域に一つの分布が見られる．これはキャッシュヒット時の傾向であると考えられる．そして，応答時間が 10

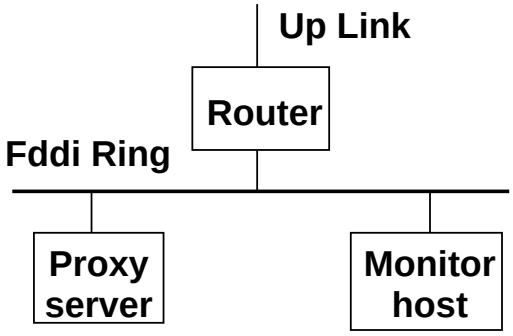

図 5.2　測定環境

ms 以上の領域にも一つの分布が見られる．これは，キャッシュミス時の傾向であると考えられる．以上より，本研究により代理サーバをその外部からパケットモニタにより性能計測を行うことが可能である．

5.3.3　ベンチマークによる比較

代理サーバの高速化手法として，メモリベース・キャッシングが、いくつか提案されている。しかし、実際に実装したという例はなく、どの程度の高速化がなされるのかといった具体的な測定データも示されていない。

そこで，メモリベースキャッシング代理サーバの 1 実装として，tmproxy [35] を用い，性能を計測した．そして，tmproxy と比較するために squid [36] の性能も計測した．

本節では，このような代理サーバの実装によりどの程度の性能差が現れるのか確認するために，ベンチマークによる代理サーバの性能計測を行った．

5.3.3.1　実験環境

表 5.1に各々の計算機の性能およびソフトウェア構成を示す．表 5.1に示すように，クライアントおよびサーバアプリケーションには polygraph 1.3.2 を用いた．代理サーバは tmproxy，squid 2.2 STABLE4 を用いた．性能計測には ENMA を用いた．ENMA をクライアント側，サーバ側の両方に設置する事により，代理サーバの挙動をより詳細に観測する事が可能である．

図 5.5に実験におけるネットワーク構成図を示す．図 5.5に示すように実験環境は 3 つ

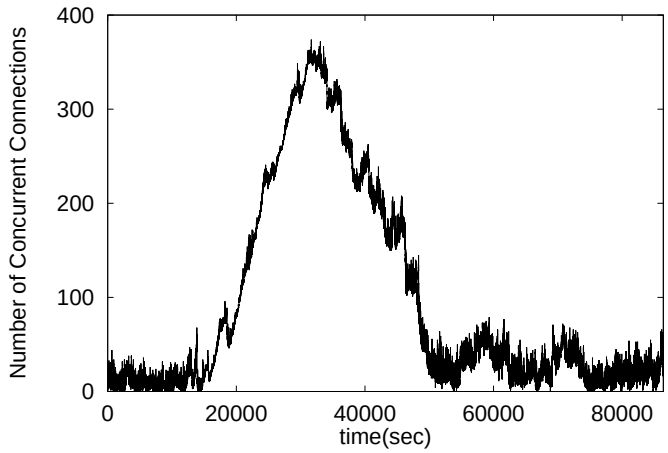

図 5.3 同時コネクション数の遷移

のネットワークから構成されている．以下に 3 つのネットワークセグメントの役割を述べる．

- クライアント側のネットワーク

  図 5.5に示すように，1 台のクライアント計算機が代理サーバへリクエストを送信する．このネットワークセグメントでは，クライアント，代理サーバ間の通信しか発生しない．したがって，このネットワークセグメントにおけるコネクションを観測する事で，代理サーバのクライアントに対する挙動を知る事が可能である．

- オリジンサーバ側のネットワーク

  図 5.5に示すように，1 台のサーバがリクエストの受付を待っている．このネットワークでは，代理サーバ，サーバ間の通信しか発生しない．したがって，クライアント側のネットワークとオリジンサーバ側のネットワークにおけるコネクションを比較する事で，代理サーバの大まかなヒット率を算出する事が可能である．また，広域ネットワークのエミュレーションを行うために，dummynet [37] を用いた．dummynet を設定する事で，片方向 100ms の遅延を挿入した．

- 実験制御用ネットワーク

  上の二つのネットワークに対しての外乱を防ぐために，計算機制御用のネットワークを設けた．これにより，より正確な計測が可能である．

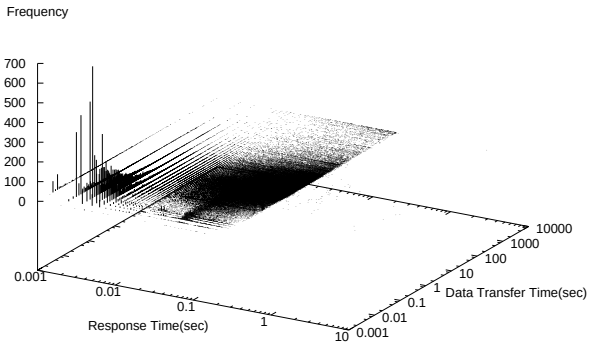

図 5.4　応答時間・データ転送時間の頻度分布

5.3.3.2　実験方法

squid, tmproxy, をそれぞれ，request rate を変更して測定を行った．測定した request rate は 100,200,300 である．400 は測定中にエラーが大量に発生したためデータを載せていない．それぞれの計算機において，以下のような設定で計測した．

- polyclt では，計測時間を 15 分，キャッシュヒット率を 55%，キャッシュ可能率を 80%，リクエストレートを 100,200,300 request/sec として設定を行った．
- polysrv では，オブジェクトサイズの平均分布を 10kbytes，オブジェクト生存期間を 2 年とした．

5.3.3.3　squid の設定

squid は代表的なキャッシング代理サーバである．squid には，独自に OS から独立した仮想記憶機能を持っている．これにより，ディスクとメモリを併用した記憶システムとなっている．今回の実験では，この仮想記憶機能におけるディスクに対する読み書きの性能を重点的に観測する事を目的としている．それを考慮して，以下のように設定した．

- メモリキャッシュサイズ 0
- ディスクキャッシュサイズ 256mbytes

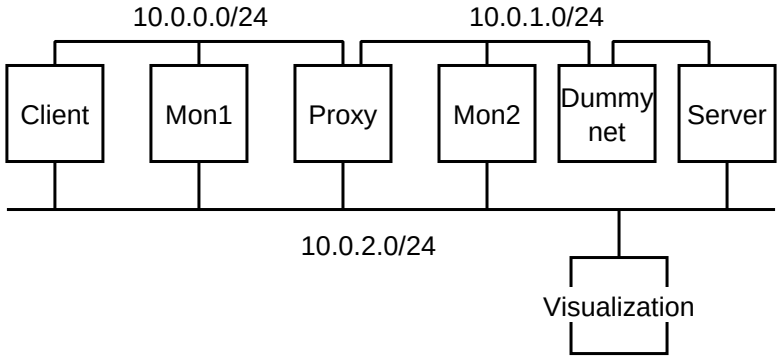

図 5.5 ネットワーク構成図

ただし，キャッシュを保持するディスクを MFS とする事で，メモリベースのエミュレーションを行っている．

5.3.3.4 ENMA の設定

リアルタイムに観測を行うための設定を行った．

- reporter

  モニタ 1,2 それぞれで起動し，リアルタイムデータを collector に転送する．

- collector

  visualization ホストで起動し，ポーリング間隔を 10 秒とした．collector を用いる事で，モニタホスト上のリアルタイムデータを他の計算機へ転送する事が出来る．

- graph generator

  collector により取得したリアルタイムデータを視覚化する．

5.3.4 性能指標

本実験では以下に示す項目を性能指標として用いた．

- Connection arrival rate(コネクション到着率)

1 秒間にリクエストが到着した数を示す．クライアント側のこの値は，polyclt のリクエストレートと同様の値となる．サーバ側のこの値は，代理サーバにヒットしなかったリクエストの数と考える事が出来る．

- Connection processed rate(コネクション処理率)

  1 秒間にコネクションを処理した割合を示す．クライアント側のコネクション到着率とコネクション処理率が同様の値を示しているなら，代理サーバは正しく機能していると言える．もし，代理サーバの処理が滞っているなら，コネクション処理率はコネクション到着率よりも小さくなると考えられる．

- Currently processing connections(コネクション処理数)

  現在，代理サーバが処理しているコネクション数を示す．この値は，代理サーバ内部のセッションスケジューリングポリシーを示していると考えられる．

- Traffic(トラヒック量)

  実際に流れたトラヒック量を示す．クライアント側とサーバ側の両方のトラヒック量を比較する事により，どの程度のトラヒックを代理サーバを用いる事で削減可能であるかを推測する事が出来る．

5.3.4.1　結果 1(squid+MFS)

図 5.6, 5.7に squid を測定した場合の結果を示す．図 5.6, 5.7は 3 つの部分から構成されている．

1. 1/19 0 時 47 分〜57 分

   polyclt によるリクエストレートは 100 である．しかし，図 5.7におけるコネクション到着率は約 50 となっている．したがって，ヒット率はおおよそ 50%となっていると言える．これは，polyclt の指定通りと言える．図 5.6でのコネクション処理レートは 100 で安定しているので，代理サーバは正常に動作していると言える．

2. 1/19 1 時 00 分〜15 分

   polyclt によるリクエストレートは 200 である．図 5.7 におけるコネクション到着率は 100〜120 と徐々に上昇している．これは，実験中において，徐々にヒット率が低下している事を示している．

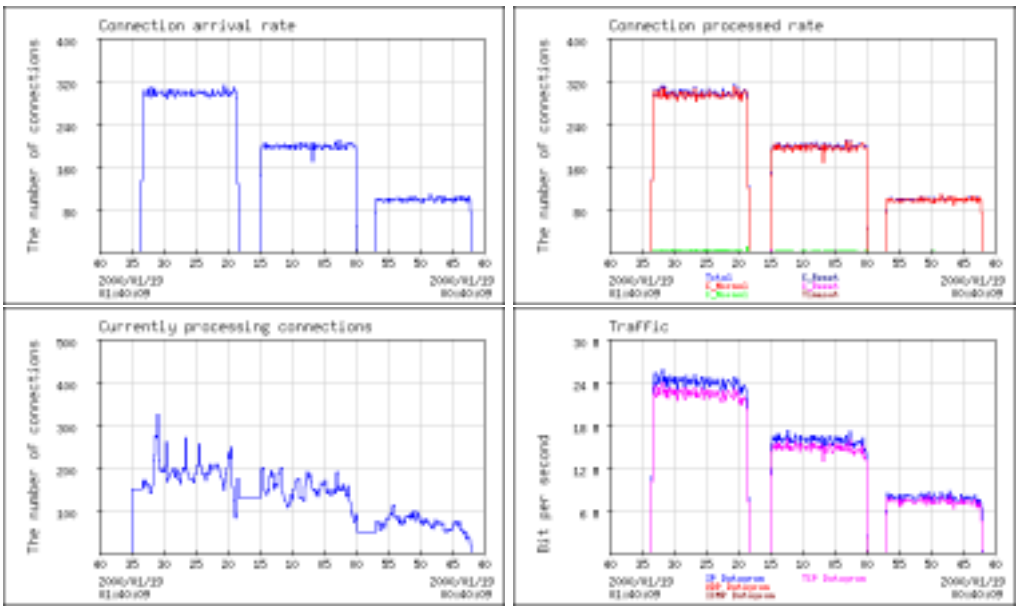

図 5.6 squid(クライアント側)

3. 1/19 1 時 18 分〜33 分

   polyclt によるリクエストレートは 300 である．図 5.7 におけるコネクション到着率は 150〜180 と 2 同様，徐々に上昇している．トラヒックを見てみると，300 request/sec においては，クライアント側で約 24Mbit/sec のトラヒックが流れている．しかし，サーバ側では 12〜16Mbit/sec となっている．したがって，キャッシング代理サーバを用いる事で 33〜50%のトラヒックの削減が可能であると言える．

5.3.4.2 結果 2(tmproxy)

図 5.8, 5.9に tmproxy を測定した場合の結果を示す．図 5.8, 5.9は 3 つの部分から構成されている．

1. 1/19 3 時 14 分〜29 分

   クライアント側のリクエストレートは 100 である．図 5.9 におけるコネクション到着率は約 50〜60 である．したがって，ヒット率はおおよそ 40〜50%となっている．

2. 1/19 3 時 33 分〜48 分

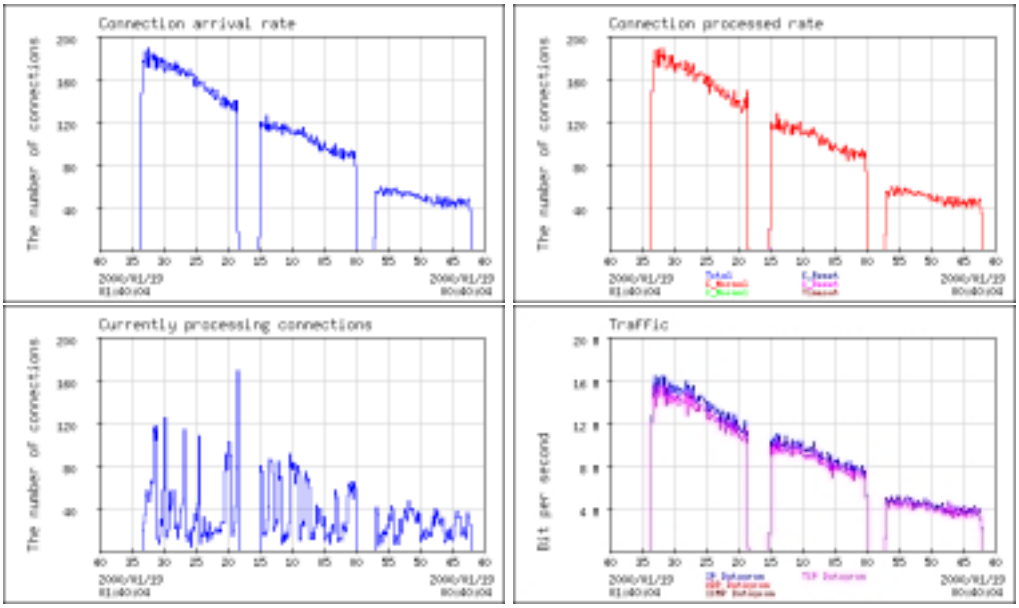

図 5.7　squid(サーバ側)

クライアント側のリクエストレートは 200 である．図 5.9 におけるコネクション到着率は 100〜130 と徐々に上昇している．これは，squid の実験と同様，実験中において，徐々にヒット率が低下している事を示している．

3. 1/19 3 時 52 分〜4 時 7 分

クライアント側のリクエストレートは 300 である．図 5.9におけるコネクション到着率は 130〜200 と上昇している．トラヒックを見てみると，300 request/sec においては，クライアント側で約 24Mbit/sec のトラヒックが流れている．しかし，サーバ側では 12〜18Mbit/sec となっている．したがって，キャッシング代理サーバを用いる事で 33〜50%のトラヒックの削減が可能であると言える．また，図 5.8の処理中コネクション数を見ると，100,200 の場合と比較して急激に上昇している．これは，代理サーバ内部のセッション管理スケジューリングの処理が徐々に滞っていると考えられる．

全体に言える事として，コネクション処理率において，正常終了でないコネクションが約 10%程度存在する．これは，tmproxy 内部のバグである可能性が高い．

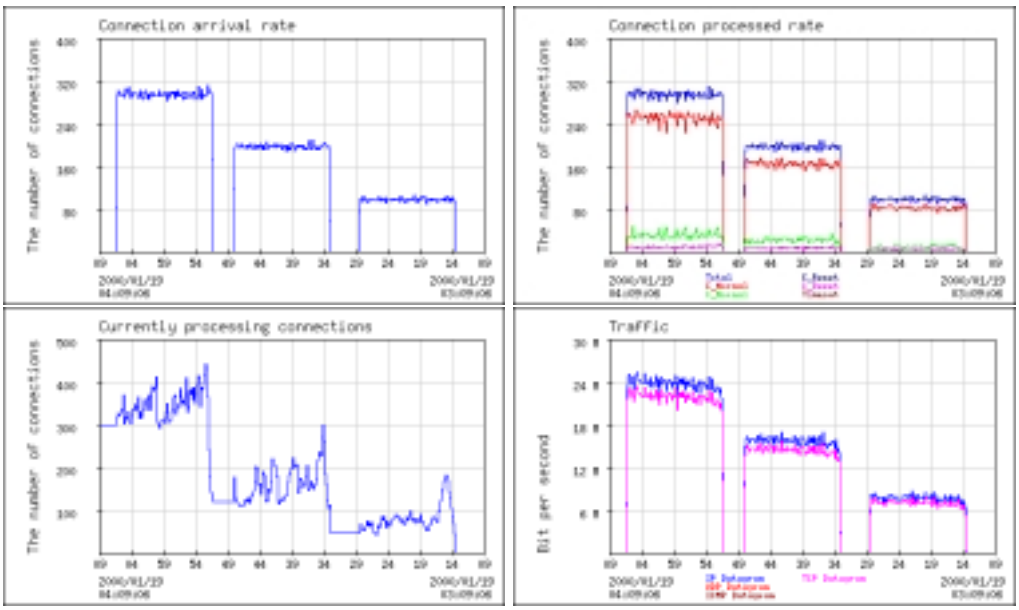

図 5.8 tmproxy(クライアント側)

5.4. まとめ

従来の計測手法では測定できなかった，同時に何本のセッションを処理できるか，もしくは，最大，どれくらいのセッションが到着しているかという項目を計測するために，本章では，パケットモニタによる，代理サーバの性能計測を行う方法を提案した．そして，本手法を用いたシステムの設計と実装を述べ，観測結果を述べた．観測結果から，実装によるシステムの性能の差は明らかに存在し，実装手法を変更することで性能向上が見られることが明らかとなった．

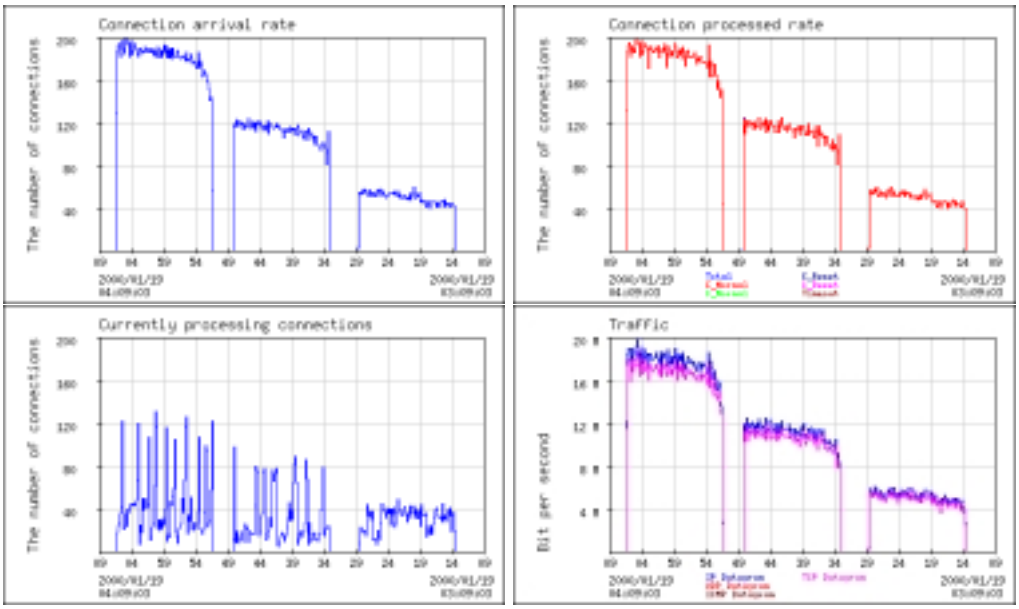

図 5.9　tmproxy(サーバ側)

表 5.1　システム構成

host	OS	CPU	memory	software
Client	FreeBSD	Celeron 400	128M	polygraph 1.3.2
Server	FreeBSD	Celeron 400	128M	polygraph 1.3.2
Proxy	FreeBSD	PentiumII 450x2	512M	tmproxy
				squid 2.2 STABLE4
Monitor1	FreeBSD	PentiumII 450	512M	enma
Monitor2	FreeBSD	PentiumII 400	128M	enma
Dummynet	picoBSD	PentiumII 400	128M	dummynet

第 6 章　サーバ管理に関する議論

本章では，これまでの結果を踏まえて，サーバ管理者の管理業務とその行動について議論する．サーバ管理者の管理業務は様々である．本章では，障害回避，キャパシティプランニングそしてパフォーマンスチューニングについて述べる．

6.1.　障害回避

6.1.1　ハードウェア的な故障

サーバ管理者はサーバシステムに障害が発生した時，障害の生じている場所を特定し，その問題からサーバシステムを復旧させなくてはならない．ENMA を用いることで，トラヒックの計測が可能となるので，ネットワーク的なトラブルが発生したなら検知することが出来る．例えば，トラブルに関して以下のような手法が考えられる．

- ハードウェア的な故障は部品を交換する
- 帯域不足の場合は，さらなる帯域確保

これらの解決方法は多くの費用を必要とする．しかし，機械的な故障の場合は部品を交換する以外に解決策はない．帯域的な問題の場合，サーバシステムが提供しているコンテンツをトラヒックの少ないテキスト形式のものにするなどしてトラヒックを減らす事が出来る．

6.1.2　サーバの性能上限

サーバシステムがハードウェアの性能限界に到達していた場合の対処方法について述べる．6.1.1 節の場合と同様にシステムに対して多くの費用を用いる事が可能であるなら，より高性能のサーバシステムを購入する事が容易な解決策である．しかし，多くのサイ

トにおいてはそのような費用を持ち合わせている事は少ない．そこで，費用を用いずにサーバシステムの性能向上を図る手法としてパフォーマンスチューニングがある．サーバの能力不足の検知は容易ではないが，rep2 を用いてカーネルモニタリングすることで，ボトルネックのポイントがある程度予測できる．パフォーマンスチューニングの詳細については，6.3節で詳述する．

6.2.　キャパシティプランニング

サーバ管理者は上述した障害の回避よりも，障害を発生させないことに注意を払う必要がある．最も多く発生する障害は，サーバシステムの性能限界を超えてリクエストが到着することである．このため，サーバ管理者は長期間の計測から将来どの程度のリクエストが到着するかの予測を立てる必要がある．現行システムの性能限界と新システム導入後の性能限界を把握しなくてはならない．そして新システム導入の際にはコスト算出する必要がある．

多くのサイトではキャパシティプランニングに求められている事は

- 何人のユーザにサービスするには，この程度のサーバが必要である
- 何人のユーザが存在する場合，リクエストはこれぐらい到着する

といった数値である．更に具体的に言うと,「このサーバはこれぐらいまでのリクエストを処理できる」といった数値を求めている．この様なサーバシステムの上限を調査するために，これまではベンチマークテストが行われている．しかし，本論文でこれまで述べてきたようにベンチマークテストの結果が実際の運用されているサーバにそのまま適用できるとは限らない．

ENMA で観測する

- リクエスト到着レート
- リクエスト処理レート

の長期的な観測により，ある程度の予測が出来ると考えられる．また,「同時セッション数」を観測することで，ある程度のユーザ数を予測して，性能の上限を予測することも出来る．本論文では，このサーバシステム性能限界に関する研究は行っていない．より現実に近いベンチマークシステムを構築する事が今後の課題として挙げられる．

6.3. パフォーマンスチューニング

本節では，サーバシステムのパフォーマンスチューニングについて述べる．パフォーマンスチューニングに関しては，主に二つに分類される．それはサーバアプリケーションの設定およびオペレーティングシステムの設定である．以下，順に述べていく．

6.3.1 アプリケーションの設定

サーバアプリケーションのパフォーマンスチューニングを考えるには，アプリケーションのソフトウェアアーキテクチャを知る必要がある．ソフトウェアアーキテクチャは表 6.1 に示す 8 種類が考えられる．

表 6.1 ソフトウェアアーキテクチャ

種類	プロセス	スレッド	セッション
1	シングルプロセス	シングルスレッド	シングルセッション
2	シングルプロセス	シングルスレッド	マルチセッション
3	シングルプロセス	マルチスレッド	シングルセッション
4	シングルプロセス	マルチスレッド	マルチセッション
5	マルチプロセス	シングルスレッド	シングルセッション
6	マルチプロセス	シングルスレッド	マルチセッション
7	マルチプロセス	マルチスレッド	シングルセッション
8	マルチプロセス	マルチスレッド	マルチスレッド

この表 6.1 で，大規模システムに拡張可能な種類は，2 以降である．アプリケーションレベルでのパフォーマンスチューニングに関しては，シングルプロセスとマルチプロセスで大別できる．

シングルプロセスの場合，アプリケーション自身が消費するメモリは少ないので，データ領域に多くのメモリを確保する事が出来る．一方，マルチプロセスの場合，起動するプロセス数によって，性能が変化する．3章で述べたように，起動する最大プロセス数と最小プロセス数の差を小さくする事でアプリケーションの性能向上を図る事が出来る．

rep2 において着目する性能指標はプロセス数，メモリ，コンテキストスイッチ数そして CPU 利用率である．マルチプロセスアーキテクチャの場合，プロセス数とコンテキス

トスイッチの数には依存関係があるので，両方の項目を着目することで，プロセス数の設定を行うことが出来る．シングルプロセス，メモリに関する使用量を注目すれば，どの程度で飽和するかが把握できる．また，サーバシステムの CPU 負荷の度合は CPU 利用率を見ることで明らかとなる．

6.3.2　OS の設定

サーバシステムの性能向上を図るパフォーマンスチューニングとして，プロトコルスタックに関するチューニングおよびバッファ(メモリ) に関するチューニングが考えられる．プロトコルスタックに関するチューニングは表 6.1 の全てに適用できる．

- 最大セッション数を大きくする
- listen queue を長くする
- send buffer や recerve buffer を大きくする
- PCB のハッシュサイズを大きくする
- Path MTU discoverty を無効にする
- TCP の TIME_WAIT 時間を短くする

などと言った事が考えられる．

マルチプロセスのシステムの場合，シングルプロセスに比較して，メモリの消費量が多いため，メモリのチューニングが有効である場合がある．またスワップ領域の増強なども有効な手段である．

以上をまとめたフローチャートを図 6.1 に示す．

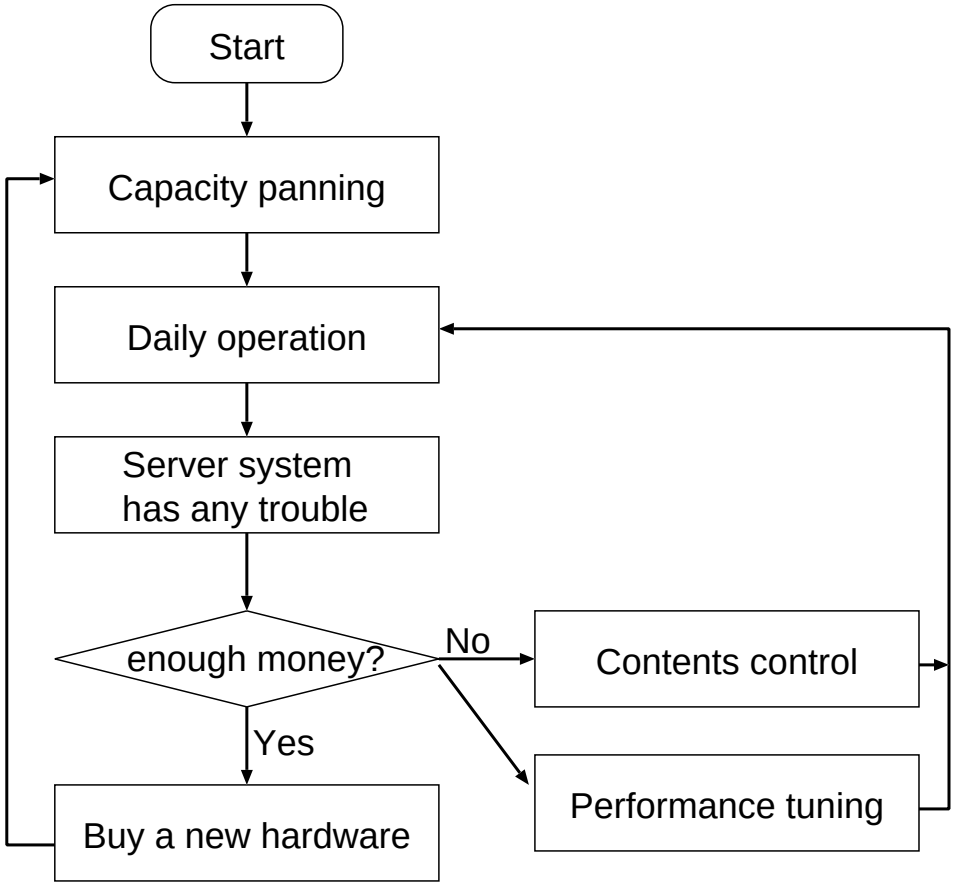

図 6.1 サーバ管理フローチャート

第 7 章　結論

現在，WWW に関して性能解析，モデリング，性能計測，性能評価といった分野で様々な研究が行われている．そして，計測のための様々なツールが開発されている．しかし，これまでの研究や開発されたツールはサーバ管理にフィードバックする事を念頭においいるものは存在しない．この様な理由から，サーバ管理者は自身の勘と経験を基にサーバを管理してきた．また，ネットワークサーバの状態を管理するツールは少なく，研究もなされていない．

そこで本論文では，サーバの状態を観測するために新たな計測ツール (ENMA) を開発した．我々は WWW サーバがパケットの送受信によってデータ交換している事に着目し，パケットモニタリングによってサーバの状態を観測するツールを開発した．この ENMA を用いる事で，サーバ管理者はリアルタイムにサーバの状態を把握する事が可能となった．

しかし，サーバの状態について我々は十分な定義をしていなかった．そのため，たとえサーバの処理が飽和していたとしても，どの部分にボトルネックがあるのか把握する事が出来なかった．そこで我々は独立した環境でサーバシステムに対して実験し，サーバシステムの状態について定義した．そしてサーバシステムの状態遷移の条件について考察した．その実験の結果，サーバシステムの状態遷移を把握するためには，サーバの内部情報が不可欠である事が明らかとなった．

我々はサーバの内部情報を抽出するプログラム (rep2) を設計，実装し評価した．従来，カーネルモニタリングによるサーバ観測は，負荷が高く実用的でないと考えられていた．本論文ではカーネル時間が 90%程度という状況をつくり出す事が出来なかった．そのため，状態遷移について確認できなかった．

7.1.　本論文によって得られた知見

本節では，本論文で得られた知見を述べる．第 2 章では，サーバの計測に関して，パケットモニタリング手法は有効な手段であり，本論文で実装した ENMA を用いることで，これまでの解析手法では明らかにできなかった応答時間や同時コネクション数を得ることが可能となった．

次に，第 3 章では，サーバ内部の状態遷移を解析し，サーバ管理の指標を得る為には，カーネルモニタリングが必要であることを明らかにした．

そして，第 6 章では，サーバに関する障害回避，キャパシティプランニング，パフォーマンスチューニングの指針を議論し，サーバ管理におけるフローチャートを得た．

7.2.　今後の課題

6 章で述べたように，キャパシティプランニングを実現するためのベンチマークシステムの構築．そして，キャパシティプランニングに関する考察．また，より高帯域のネットワークにおけるモニタリングが今後の課題として挙げられる．

7.2.1　現実環境に近いベンチマークシステムの構築

本節では，現在のベンチマークテストではなぜ運用されているサーバシステムに適用できる結果が得られない理由について考察する．

大きな理由は遅延時間である．実際に運用されているサーバへアクセスするユーザのネットワーク的距離は一様ではない．したがって，セッション毎の遅延時間が異なる．この現象をベンチマークシステムで実現することは困難である．

本論文では dummynet を用いて遅延時間のエミュレーションを行い実験した．しかし，このエミュレーションでは不十分である．セッション毎に異なる遅延時間を挿入したエミュレーションを行う必要がある．そこで，より現実に近いベンチマークシステムの構築のために必要なアイデアとして，

- 1 プロセスに 1 クライアントの割り当てる．
- 1 クライアントに対して 1 つの IP アドレスを割り当てる．
- dummynet を用いて IP アドレス毎のネットワーク遅延を挿入する．

以上を実装することで，より現実に近いベンチマークシステムの構築が実現できると考えられる．

7.2.2 キャパシティプランニングに関する考察

キャパシティプランニングは以下の要因から，評価が困難である．

- リクエスト到着予測の困難さ (ユーザの増加率の算出)

  長期的観測をすればある程度は予測可能と思われる．しかし，新たなサービスを開始する時にユーザがどの程度存在するかの調査は容易ではない．サーバ管理者は，ある程度「何人規模のサーバシステムが必要である」という前提を持ってシステムを構築する必要がある．

- ハードウェア的な性能限界の把握の困難さ

  様々なベンチマークシステムにより，性能限界の調査は容易にはなっている．しかし，ベンチマークによる結果と現実に発生する現象とは異なる．このため，ある条件の元での性能限界を知ることは出来ても，実際の環境での性能限界の把握は困難である．

- 導入後のシステムに対する妥当性

  プランニングした後のシステムに対する評価方法が確立されていない．基本的に払われたコストに対してどの程度の性能向上したかが判断できれば良い．しかし，先に述べたように性能限界の調査が容易ではないため，新しいシステムに対する評価も困難である．

7.2.3 広帯域ネットワークにおけるパケットモニタリング

これより先，数年の後にギガビットネットワークが普及することは間違いない．しかし，本論文で述べているパケットモニタリング技術では 100Mbps を超える高速ネットワークのモニタリングは容易ではない．その理由として,「ディスクへのログの記録がネットワークと比較して非常に低速である」ということが挙げられる．この様な問題を解決する為には，例えば Layer 4 Switch を用いた，モニタリングの分散システムが考えられる．

今後は以上の項目についての解決手法を提案し，実装すること目標としている．

謝辞

本研究を行う機会を与えて下さいました湊小太郎教授に感謝致します．本研究の基本的なアイデアや様々な叱咤激励をして下さった山口英教授に心より感謝致します．適切な助言を下さった福田晃教授に感謝します．遠隔地にも関わらず，研究の進捗を心配し助言をして下さった九州工業大学の尾家裕二教授に深く感謝致します．日頃から常に研究成果を見守って下さった砂原秀樹助教授に感謝致します．

論文執筆にあたり，筆者の稚拙な文章を根気良く直して下さった，そしてENMA の実装における貴重なアドバイスを下さいました情報ネットワーク講座の知念賢一助手に深く感謝致します．

本研究は，全国高校野球選手権大会のインターネット中継を対象とし，実験を行いました．全国高校野球連盟，朝日放送株式会社，サイバー関西プロジェクト (CKP)，WIDE プロジェクトの皆様の協力により実験を行うことが出来ました．実験に携わっていただいた全ての皆様に感謝致します．特に，サーバ設置やデータ収集に協力して頂きました，朝日放送株式会社の吉田豊一氏，香取啓志氏そしてNTT スマートコネクト株式会社の沖本忠久氏，白波瀬章氏に心より感謝致します．

本研究を進めるにあたり，議論に参加して頂いた河合栄治氏，森島直人氏，附属図書館研究開発室の羽田久一助手，新麗助手，ソニー株式会社の普天間智氏および情報科学センターの皆様に深く感謝致します．

参考文献

[1] *NCSA Mosaic.* http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html.

[2] *Hobbies' Internet Timeline v5.2.* http://www.isoc.org/guest/zakon/Internet/History/HIT.html.

[3] Jeffrey C. Mogul. Network Behavior of a Busy Web Server and its Clients. Technical report, DEC Western Research Laboratory, October 1995.

[4] *Status of the CERN httpd.* http://www.w3.org/Daemon/.

[5] *NCSA HTTPd Hoge Page.* http://hoohoo.ncsa.uiuc.edu/.

[6] Louis P. Slothouber. A Model of Web Server Performance. In *Fifth International World Wide Web Conference*, Paris, France, May 6-10 1996.

[7] Jeffrey C. Mogul. The Case for Persistent-Connection HTTP. In *SIGCOMM '95 Symposium on Communications Architectures and Protocols*, pp. 299–313, Cambridge, MA, August 1995.

[8] Venkata N. Padmanabhan and Jeffrey C. Mogul. Improving HTTP Latency. http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/DDay/mogul/HTTPLatency.html.

[9] John Heidemann. Performance Interactions Between P-HTTP and TCP Implementations. *ACM Computer Communication Review*, Vol. 27, No. 2, pp. 65–73, April 1997.

[10] Almeida Virgilio, Almeida Jussara and Murta Cristina. Performance Analysis of a WWW Server. Technical report, Computer Science Department, Boston University, August 5 1996.

[11] 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.

[12] Gaurav Banga and Peter Druschel. Measuring the Capacity of a Web Server. In *USENIX Symposium on Internet Technology and Systems*, Monterey, California, USA, December 1997.

[13] Jussara M. Almeida, Virgilio Almeida, and David J. Yates. *Web-Monitor: a Tool for Measuring World-Wide Web Server Performance.* http://www.firstmonday.dk/issues/issue2_7/almeida/.

[14] *the protocol packet capture and dumper program.* http://www-nrg.ee.lbl.gov/nrg.html.

[15] *packet sniffer and monitoring tool.* http://sniffit.rug.ac.be/ coder/sniffit/sniffit.html.

[16] *The Etheral Network Analyzer.* http://www.ethereal.com/.

[17] *The Lightweight Network Intrusion Detection System.* http://www.snort.org/.

[18] *The Public Netperf Homepage.* http://www.netperf.org/netperf/NetperfPage.html.

[19] *A Tool for Measuring Internet Path Characteristics.* http://www.employees.org/ bmah/Software/pchar/.

[20] *The Multi Router Traffic Grapher.* http://ee-staff.ethz.ch/ oetiker/webtools/mrtg/mrtg.html.

[21] Joel Apisdorf, k caffy Kevin Thompson and Rick Wilder. OC3MON:flexible, affordable, high performance statistics collection. In *INET'97 conference*, Kuala Lumpur, June 25-27 1997.

[22] *An explanation of the SPECweb99 benchmark*. http://www.spec.org/osg/web99/.

[23] David Mosberger and Tai Jin. httperf – A Tool for Measuring Web Server Performance. In *1998 Workshop on Internet Server Performance*, Madison, Wisconsin, June 23 1998.

[24] Gene Trent and Mark Sake. *WebStone: The first generation in HTTP server benchmarking*. http://www.sgi.com/Products/WebFORCE/WebStone/.

[25] *IETF(The Internet Engineering Task Force)*. http://www.ietf.org/.

[26] *Definitions of Managed Objects for WWW Services*. RFC 2549.

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

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

[29] *Apache HTTP Server Project*. http://www.apache.org/.

[30] 中村豊, 知念賢一, 砂原秀樹, 山口英. ENMA:パケットモニタによる WWW サーバの性能計測システムの設計と実装. 電子情報通信学会論文誌, Vol. J83-D-I, No. 3, pp. 329–338, 3 月 1999.

[31] *PicoBSD, the Small BSD*. http://people.freebsd.org/ picobsd/picobsd.html.

[32] *Chamomile Home*. http://minatow3/ eiji-ka/chamomile/index.html.

[33] Jussara Almeida and Pei Cao. *Wisconsin proxy benchmark 1.0*. http://www.cs.wisc.edu/ cao/wpb1.0.html.

[34] *Web Polygraph*. http://polygraph.ircache.net/.

[35] 梶田朋己, 中村豊, 知念賢一, 砂原秀樹. メモリ・キャッシング代理サーバの実装–アクセスピーク時における応答時間劣化軽減手法–. In *Internet Conference 98*, Dec 1999.

[36] *Squid Web Proxy Cache*. http://www.squid-cache.org/.

[37] L.Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *ACM Computer Communication Review*, Vol. 27, No. 1, pp. 31–41, January 1997.

著者研究業績

1．学術論文

(1) 中村豊, 知念賢一, 砂原秀樹, 山口英: ”パケットモニタによる WWW サーバの性能計測システムの設計と実装”, 電子情報通信学会論文誌, VOL.J83-D-I, NO.3, Mar. 2000.

(2) Yutaka Nakamura, Ken-ichi Chinen, Suguru Yamaguchi, Hideki Sunahara: ”An Analysis of WWW Server Status by Packet Monitoring”, IEICE TRANSACTIONS on Information and Systems, VOL.E83-D, NO.5, MAY. 2000.

2．国際会議（査読つき）

(1) Yutaka Nakamura, Ken-ichi, Chinen, Hideki Sunahara, Suguru Yamaguchi, Yuji Oie: ”ENMA: The WWW Server Performance Measurement System via Packet Monitoring”, Proceedings of INET’99, June 1997.

3．研究会

(1) 中村豊, 知念賢一, 山口英, 砂原秀樹: ”パケットモニタによる WWW サーバ挙動解析方法の提案”, Internet Conference 98, Dec. 1998.

(2) 中村豊, 知念賢一, 山口英, 砂原秀樹: ”パケットモニタによる代理サーバの性能計測”, Japan Web Cache Workshop, Apr. 1999.

・共著

(3) 梶田朋己, 中村豊, 知念賢一, 砂原秀樹: ”メモリ・キャッシング代理サーバの実装 –アクセスピーク時における応答 時間劣化軽減手法–”, Internet Conference 99, Dec. 1999.

(4) 中山貴夫, 中村豊, 知念賢一, 砂原秀樹: ”WWW サーバにおける特徴抽出のための過渡解析手法”, 信学技報, IN99-89, Jan. 2000.

4. 解説記事

(1) 中村豊, 小川晃通: ”中継システム –WEB Casting, DV Multicasting–”, 情報処理学会学会誌, 第 41 巻, 第 11 号, Nov. 2000.