マルチメディア通信と分散処理　93－4
（1999．4．16）

Webオブジェクトオーガナイザ

－自律分散型Web負荷管理システム－

山中 英樹
富士通研究所 情報科学研究部

アブストラクト

このシステムは，主にイントラネット，あるいは遅延の大きな衛星通信などのネットワークに設置されたWeb (HTTP, FTP, CORBA (IIOP), SSL 等を含む) サーバ群，プロキシサーバ群に掛かる負荷を管理者の介入なしに自律的に負荷分散するシステムである．これは，既存の各サーバに1個ずつ併設されるだけで，サーバのシステム，サーバの設定をほとんど変更せずに，サーバ群を数秒程度で高速に協調させて負荷分散させることができる．

さらに，次世代HTTP通信で採用されることになっている多重化によるHTTP通信の高速化機能を先取りしているので，次世代HTTPを待たずにこの機能のメリットを享受できる．(シミュレーション実験では，HTTP 1.0並列コネクション通信に比べ最大4.53倍程度の高速化が達成できている．)

Web Objects Organizer

－ Autonomous Distributed Web Load Management System －

Hideki Yamanaka
Information Science Lab., Fujitsu Laboratories Ltd.

Abstract

In this paper, we propose an administration free Web (including HTTP, FTP, CORBA(IIOP), SSL, etc.) load management system that can hide effectively network latency from the terrestrial communication to satellite one. This system can be easily introduced in any working Web servers and proxies with few configuration changes, and extend the network throughput and reliability by making the dynamic cooperation of the loads of distributed Web servers and proxies every few seconds.

Moreover, in this system we have adopted a multiplex communication technique similar to that of next generation HTTP and can enjoy its throughput which is up to 4.53 times as large as that of usual parallel HTTP communication (shown in our simulation experiment).

1. はじめに

WWWの普及が進み，最近では部門，部，課，学部，学科のような単位でWeb (HTTP, FTP, CORBA (IIOP), SSL他を含む) サーバが立ち上げられることも普通のことになってしまった．さらに，WWWのための通信が急速に増大したためにWebサーバへの負荷集中によるレスポンスの低下，イントラネットの輻輳が問題となり，トラヒックの低減のために職場，部署単位でプロキシサーバも立ち上げられるになって来た．このようなサーバの普及にともなって，個々のサーバ管理だけでなく，イントラネット全体としてのトラヒックを考慮したサーバ管理の増大，繁雑化，専門化が問題となっており，管理を省力化，自動化する機構の必要性は年々高まっている．

Webは元々個々に独立した分散データベースサーバ群として設計されていたため，サーバ間を協調させる機能が弱く，大域的にトラヒックを管理する機能がないに等しい．プロキシサーバは，Webオブジェクトをネットワークの中継点にキャッシュすることで，不必要なトラヒックの発生を抑えるという消極的な意味のトラヒック管理ができるが，キャッシュ間の連係，協調が弱いので効果が限定されている．最新のプロキシサーバは，この連係，協調 ([ICP]) を強化し区別されてキャッシイングサーバと呼ばれるようになって来たが，ルータがしているような負荷に従ってトラヒックパスを動的に変更できる訳ではなく，サーバの故障，復旧に迅速に自律的に対応できる訳でもない．

さらに，Webの通信プロトコルHTTP([HTTP])はTCPというストリーム通信向けのトランスポートを採用しているが，HTTP自体は本質的にトランザクションのためのパケット通信プロトコルであるため，この間に重大なミスマッチ ([LAT,MUX]) がある．例えて言えば，同じ人に10件の要件があるときに，1件毎に電話を掛け直して時間を無駄にするようなもので，ネットワークを不必要に輻輳させてしまう．

耐故障性からの要請で, 通信相手の突然の通信途絶, 復旧後直ちの通信への再参加に対処できるための特定しない複数の計算機間の通信手段は重要である. イントラネット, インターネットでは, IP マルチキャスト通信という手段が整備されていて, 大域的なトラヒック管理のためのルータ間通信プロトコルで既に利用されている. トラヒックの管理は今まで (IP パケットレベルの) ルータだけでなされて来たが, Web 通信に関してはルータだけでは不十分であることが明らか ([NCSA,CISCO,IBM]) になり, Web のトラヒックを自律的に管理する専用ルータ (アプリケーションレベルルータ) が求められている. それは当然, マルチキャスト通信に基づくルーティングプロトコルを持ち, Web サーバ群の負荷とネットワークの負荷を最小化しながら, しかも両者をバランスさせることができなければならない.

このような背景の基で, Web オブジェクトオーガナイザは Web 環境の大域的な自律負荷分散管理システム, Web 専用のアプリケーションルータとして開発された.

2. Web オブジェクトオーガナイザ

Web オブジェクトオーガナイザは, 既存の各 Web サーバ, プロキシサーバのフロントエンド, バックエンドとして動作する自律分散型 Web アプリケーションルータ (図 2.1 参照) で, 以下のように動作する.

1) オーガナイザは, 起動するとマルチキャスト通信 (図のネットワークの中央部の楕円部分) から他の既に動作しているサーバの構成, 負荷情報 (URL に基づくルーティング情報) を収集し, 必要ならばこれらのサーバとの間に多重化された通信を可能とするコネクション (データ通信用) を張る.
2) Web クライアント (図の右側 LAN 内) からのリクエストを受け取ったプロキシ (キャッシュ) サーバは, オブジェクトがキャッシュされていないときには, リクエストを最も近くにある指定されたオーガナイザに転送する. ローカルなオーガナイザは, プロキシサーバから上流のプロキシサーバと見做される.
3) このオーガナイザは, 1) でマルチキャストにより取得しておいた URL に基づくルーティング情報を用いてリクエストを適切な Web サーバに関係付けられたオーガナイザ (図中, 左の LAN に繋がるもの) へ多重化されたコネクション (データ通信用) を使ってリクエストを中継する.
4) 後者オーガナイザは, 関係付けられた Web サーバへリクエストを転送し, その応答を受け取る.
5) 応答は, リクエストが送られて来たパスを逆に辿り, プロキシサーバを経由して Web クライエントに送られる.

複数の Web サーバ (mirror) により負荷分散を行うときには, 各 Web サーバに 1 つずつのオーガナイザを割り当てておく. (通常は, Web サーバとオーガナイザの一対を同じ計算機上で稼働させる.) この場合には, 上の 3) のステップで複数の Web サーバの負荷, 障害情報を入力とし, 各オーガナイザの負荷分散アルゴリズムがリクエストを転送するオーガナイザを動的に決定することで, ネットワーク全体が自律的に負荷分散を行う.

2.1 オーガナイザプロトコル

Web オブジェクトオーガナイザ間の通信プロトコル (オーガナイザプロトコル) は, 1) トランスポート層に UDP,TCP の両方を適宜使い分けて用いる, 2) サーバとクライアントの区別の無い完全対象型で, 3) リクエストの応答を待たない非同期型 RPC(Remote Procedure Call) プロトコルである. UDP は, 主にサーバの負荷情報, 構成情報, システムの状態, オーガナイザ間のネットワークの遅延情報を交換するための制御用のマルチキャスト, ユニキャスト通信で用いられる. TCP は, 信頼性のある多重化データ通信, 信頼性が必要な制御通信に用いられる. 多重化通信部分のプロトコルは, 通常の大きさのオブジェクト (150KB 程度以下) を 1RTT 時間で転送できるようにウインドウイングしているだけで, 基本的に輻輳制御の無い簡約化 TCP に非常に近いものである. (これは, 次世代 HTTP([MUX]) で採用される多重化プロトコルとは全く違うマルチキャスト通信も併用する独自プロトコルである. 次世代 HTTP が標準化されたときには, このオーガナイザプロトコル上に次世代 HTTP 通信も多重化する予定である.) HTTP その他のオーガナイザで中継されるプロトコルは, この多重化プロトコルをトランスポート層としてこの上で中継される. (オーガナイザプロトコルの多重化通信部分自身は, TCP をトランスポート層としこの上で輻輳制御される. ネットワークのバンド幅が大きく輻輳の起きないときには, 多重化部分には輻輳制御が働かないので, 中継されるプロトコルは TCP の制約 (slow start 機構 ([SLS]), コネクション確立のためのオーバヘッド, その他の輻輳制御) から自由になる.)

2.2 IP ネットワークからの独立

現在の Web 環境は IP 通信を前提とするため, その制約でネットワークを物理的に広い範囲で動的に構成変更できない. それは全ての機器に IP アドレスが振られていて, 物理的に設置されたルータが IP アドレスに基づいて通信を中継するため, ルータから離れた場所には機器を置きにくいためである. また, 各機器に振られた IP アドレスにはそれに関するルーティングを最終的に管理するルータがあり, そのルータから別のルータの管理するネットワークに (IP アドレスを換えずに) その機器を移すとパケットがルーティングされず, 通信が途絶してしまう.

HTTP(化) プロトコルは, 単にトランスポート層として TCP/IP を採用しているだけで, 直接 IP プロトコルに依存している訳ではない. URL の中で指定されたサーバに対してコネクションを張るため, サーバ名を IP アドレスに変換する必要があり, このために IP 通信の制約を受けている. 従って, サーバ名からその IP アドレスへの変換を経由しない別の機構を使ってコネクションを張ることができれば, IP 通信の制約を回避できる. 例えば, 既にプロキシサーバの機構に組み込まれているリダイレクション機能も, この仕組みの 1 つである. ただし, このリダイレクションは予めプロキシサーバのコンフィギュレーションファイルに静的に記述されてされていて, 動的にリダイレクション先を切替える機能を持たない.

IP ネットワークと独立した Web 通信環境を構築するためには, HTTP(化) プロトコル専用のネットワーク, Web ルータが必要である. しかし, 新たに IP ネットワークと別に作らなければならないものではなく, IP ネットワークの中にいわゆるトンネリングの手法を使って仮想的なネットワーク, ルータを構築するだけで十分である. このような環境は全く新しいものではなく, 原始的ではあるが, 既にプロキシサーバの多段接続の形で出現している. 逆に言えば, 十全な Web ルータがプロキシサーバの将来の発展した姿であるとも考えられる. 現在, プロキシサーバをルータと言うに憚られるのは, ルータに本質的な自律的ルーティング機能が無いからである. この機能は, プロキシサーバの概念に則して言えば, 負荷変動, 障害等に自律的に対応できる動的なリダイレクション機能である. この環境では, URL が直接ルーティング対象のアドレスで, サーバという物理的概念がなくなり, URL で指定されたオブジェクトがネットワーク上に存在するだけであり, 物理的概念から自由になる.

Web オブジェクトオーガナイザは, URL で HTTP(化) リクエストを直接ルーティングする Web 専用ネットワークのルータとして位置付けられる. オーガナイザ間の TCP コネクションが IP ネットワーク内をトンネリングし, このオーガナイザのネットワークが Web 専用ネットワークを構成する. 各オーガナイザは, URL に基づくルーティング情報を IP マルチキャスト通信で相互に交換し合うことで, リクエストの負荷の効果的な分散が可能で, また, 耐故障性も同時に高めることができる.

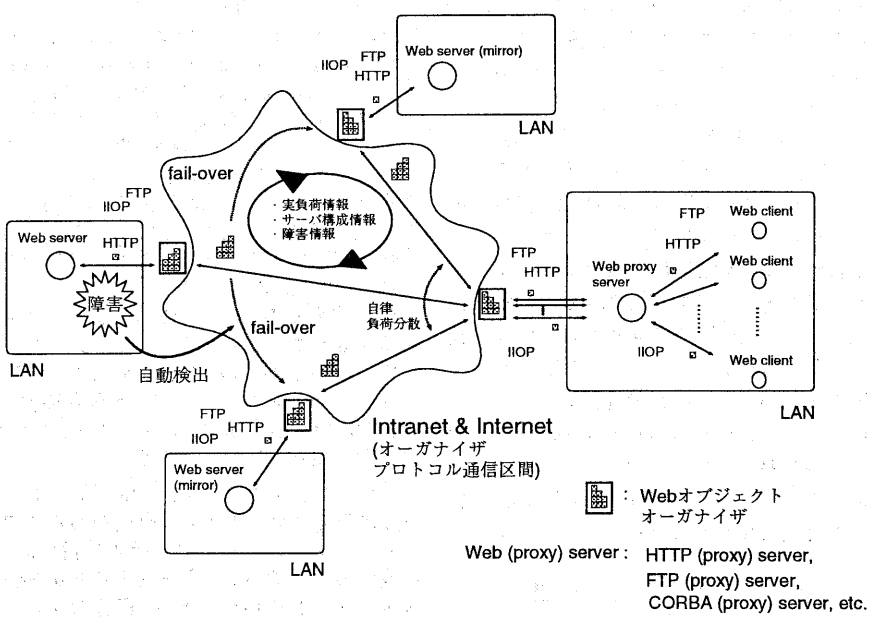

図 2.1 Web オブジェクトオーガナイザの概観

3. シミュレーション実験と評価

ここでは, Web オブジェクトオーガナイザの多重化通信による HTTP 通信の高速化についてシミュレーションを行う. (HTTP 通信の高速化が, 他のサポートされているプロトコルの通信より大きいので HTTP 通信だけを測定した.) LAN 環境での HTTP エージェント (ロボット) による負荷分散のシミュレーションは, 予め負荷の掛かり具合が判り, 単純にラウンドロビンで負荷を分散させるだけでほぼ理想的な負荷分散が達成できるので, ここでは評価しない.

測定に使った Web サーバは, Sun Web Server

1.0 ([SWS]) で, 参考データの測定に使ったプロキシサーバは, Squid 1.1.22 ([SQ]) である. 測定環境は, 計算機がエージェントシミュレータ用 4 台, Web サーバとオーガナイザ共用 1 台, オーガナイザ用 1 台 (各々Pentium II 233Mhz Solaris 2.5.1 マシン) で, ネットワークが 100BASE-TX (half-duplex mode) である. 測定パラメータは, エージェント数 (同時並列コネクションの数) とオーガナイザ間の遅延 (RTT: Round Trip Time) である. 測定項目は, オーガナイザの総スループットとプロキシサーバ Squid の総スループットである. (エージェント, Web サーバの TCP パラメータは, MSS:Maximum Segment Size 1460 バイト, 送受信バッファサイズ 6MSS, 初回送信 2MSS である.)

シミュレーション環境の構成 (図 3.1) は, 計算機 A に Web サーバとオーガナイザ A, 計算機 B にオーガナイザ B, 計算機 C,D,E,F に計測用 HTTP エージェントのシミュレーションプログラムを配置する. エージェントは, 計算機 C,D,E,F 上に均等に分布し, 各エージェントからの HTTP1.0 リクエストをオーガナイザ B がフロントエンドとなって受け取り, それをオーガナイザプロトコルの多重化通信でオーガナイザ A と通信する. オーガナイザ A は Web サーバに HTTP1.0 プロトコルでコネクションを張り, エージェントからの HTTP リクエストを中継する. 従って, エージェントシミュレータとオーガナイザ間, オーガナイザと Web サーバ間は HTTP1.0 の TCP コネクションの並列接続で, オーガナイザ間は持続接続された単独の TCP コネクション上を多重化通信で結ばれる. 測定を高速 LAN 内で行うため, オーガナイザ間の通信に WAN の疑似的な遅延を挿入する. (この遅延は, オーガナイザプログラム中に組み込まれた, TCP レベルで遅延をシミュレーション (slow start と sliding window([SLS]) のみ) するモジュールで挿入される. このモジュールは, エージェントシミュレータにも組み込まれていて, 後で Squid の計測に使われる.)

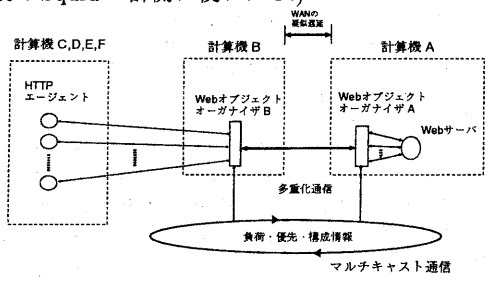

図 3.1 シミュレーション環境の構成

測定は, エージェント数 1,4,8,16,32,48,100,200, 遅延 0,10,50,100,200,500msec の全ての組み合わせについて行った. また, オーガナイザのプロキシサーバとしてのオーバヘッドをプロキシサーバ Squid と比べるために, 計算機 B の位置のオーガナイザを Squid で置き換え, Squid と計算機 A の Web サーバを HTTP1.0 で直接結ぶ構成にし, 計算機 C,D,E,F のシミュレータと Squid の間に上と同様の疑似遅延を挿入した場合の総スループットも測定した. (測定に使った Web オブジェクトは平均 19.5KB のこの種の測定に持ちいられる標準的な負荷 (表 3.1) である.)

from	to	Ratio
0	1KB	0.350
1KB	10KB	0.500
10KB	100KB	0.140
100KB	1MB	0.009
1MB	10MB	0.001

表 3.1 標準的な HTTP サーバ上のオブジェクトの大きさと割合

測定結果は, 付録の図 6.1, 図 6.2 が各々エージェント数を 1,4,8,16,32,48,100,200 に固定したときのオーガナイザ (TCP バッファサイズ 1024KB) と Squid のネットワーク遅延 (RTT) とスループットの図である. Organizer (1024KB) theor. limit とラベルされた曲線は, オーガナイザ間の TCP バッファサイズを 1024KB にしたときのオーガナイザ間の総スループット理論限界である. このスループット限界を Th [Mbits/sec], TCP バッファサイズを $Bufsize$ [KB], 遅延を RTT [msec] とすると, $Th = Bufsize*8/1024/(RTT/1000) = Bufsize*7.81/RTT$ である. さらに高速な計算機を使う場合には, このバッファサイズによるスループットの抑制が顕在化するので, さらにバッファサイズを大きくする必要がある. ちなみに, このグラフには表示されていないが HTTP の総スループット理論限界は各コネクションのスループット理論限界 (図 6.3,6.4 の HTTP theor. limit) にコネクション数 (エージェント数) を掛けたものである.

オーガナイザの主な利用は, 図 2.1 にあるように Squid のようなキャッシュ機能があるプロキシサーバのバックエンドに用いてその負荷分散, 高速化を達成することである. オーガナイザが Squid 以上のスループットを持つことは, 既存のプロキシシステムにオーガナイザを付加しても, そのオーバヘッドによってスループットが減少しない保証になる.

図 6.3,6.4 は, 図 6.1,6.2 の図をパラメータであるエージェント数で割ったときの図で, 1 エージェント当たりの平均通信速度の図である. HTTP theor. limit とラベルされた曲線は, 標準的な負荷を HTTP 1.0 で通信したときの理論限界をプロットしたもので, どんなに高速な計算機, 広帯域なネットワーク

を用いてもこの限界を越えたHTTP 1.0通信はできない．理論式は，平均的大きさのオブジェクト19.5KBをTCPで転送するときに掛かる平均時間$4.53RTT$で割ったもの$19.5*8/1024/(4.53RTT) = 33.6*10^{-3}/RTT$ [Mbits/sec]である．($4.53RTT$は，各オブジェクトの転送に掛かる時間の理論値を平均したもので，大まかには，オブジェクトの大きさの平均値19.5KBを転送するのに掛かる時間の理論値(ネットワークのMSSが1460バイトのときに，TCPコネクションの確立で$1RTT$，初回$1460*2$バイト，2回目$1460*4$バイト，3回目$1460*6$バイト，そして最後に$1460*1+988$バイトのデータ転送が起こることから$4.33RTT$)として求められる．初回$1460*2$バイト転送するのは，Solaris以外の多くのOSで使われているBSDから派生したTCPモジュールのバグ([SLS]参照)と互換性を持たせるためである．)

Organizer theor. limitとラベルされた曲線は，単純にHTTP theor. limitを4.53倍したものである．(厳密には，全てのオブジェクトの転送に掛かるRTT時間の平均値である．大まかに説明すると，オーガナイザ通信では，リクエストをサーバに送ると$1RTT$時間でほとんど全てのオブジェクトを転送できるが，HTTP 1.0は平均$4.53RTT$時間掛ることから4.53が求められる．) 図6.4で，負荷が軽いときSquidのスループットがHTTP理論限界に達している．また，図6.3で実際のオーガナイザのスループットがこのHTTP理論限界を越えた領域に入り，オーガナイザ理論限界に迫っているので，既にSquidは計算機の能力ではなくHTTP通信の限界により性能が抑えられていることが判る．逆に，この領域ではHTTP通信を使う限り計算機に幾らお金を注ぎ混んでも性能は上がらず，無駄であることを示している．

この計測のエージェント数は，通常のWeb環境でのユーザ数に直接対応する訳ではない．例えば，オーガナイザの13Mbits/secの総スループットは，常にコネクションを張りオブジェクトを転送し続けた場合なので，これを通常のWeb環境に当てはめて計算すると，全くストレスの無い快適な状態で同時に487ユーザを賄える性能に相当する．(ユーザが1分間に1度Webのアンカをクリックするとし，そのWebページに10オブジェクト(平均20KB程度)が存在すると仮定する．13Mbits/secのスループットのとき，20KBのオブジェクトを$20*10^3*8/(13*10^6) = 12.3*10^{-3}$ secで転送できるので，1分間に$60/(12.3*10^{-3}*10) = 487.8$ユーザ分のオブジェクトを転送できる．従って，この速度で同時に487ユーザを非常に快適な状態で賄うことができる．) 現状のWeb環境では，ネットワークの渋滞のためにストレスを感じるのが普通であり，キャッシュサーバがクライアントと同じLAN内にあり重複したデータ転送を抑制(ヒット率40%程度)するので，13Mbits/secの総スループットで同時に1000～3000ユーザの規模を賄える計算が成り立つ．さらに，オフィスでは同時に全てのユーザがWebクライアントを使う訳ではないので，オフィスの規模としてはその数倍のユーザ数になる．

4. まとめ

HTTPとそのトランスポート層であるTCPのミスマッチを解消する手段が多重化通信であり，そのためにはHTTPを直接にトランスポート層とする仮想ネットワークの考え方が有効であることを述べた．今回開発したWebオブジェクトオーガナイザは，この仮想ネットワークのルータ(URLのプレフィックスとIPアドレスの組でルーティング)として位置付けられるもので，直接Web (HTTP, FTP, IIOP, SSL他を含む)ネットワークをIPネットワーク上にトンネリングという手法で仮想的に構築する単位である．

現在の通常のTCP並列コネクションを使ったHTTP 1.0通信の中継スループットを世界中で広く使われている高速プロキシサーバSquidを使って，多重化HTTP通信の中継スループットをWebオブジェクトオーガナイザを使って，疑似的にTCPレベルでシミュレーションしたWANの中で計測した．この結果から，多重化したHTTP通信の中継スループットは，並列HTTP通信のそれより50msec以下の遅延の小さな領域で1～1.2倍，遅延の大きな領域では2～4.53倍高速であることを示した．

また，例えば(同時連続)通信エージェント数32以下では，既にSquidはHTTP通信の理論限界性能に達している(図6.4)ので，これ以上どんなに高速な計算機，広帯域ネットワークを用意してもこの性能を越えることができないが，オーガナイザはHTTP通信の制約を受けないので，この理論限界以上の性能を出すことが可能で，実際に本実験でもこの限界を越える性能(図6.3)を出している．オーガナイザ通信の理論限界性能は，HTTP通信の4.53倍で，遅延の非常に大きな領域でオーガナイザは既にこの性能に達している．エージェント数の大きな負荷の高い場合，オーガナイザはCPU能力不足のため理論限界性能以下であるが，それでもSquidの約2倍程度の性能を確保している(図6.3参照)．

オーガナイザの通常の利用形態は，図2.1にあるようにWebクライアントが一旦キャッシュサーバを経由してオーガナイザに接続し，オーガナイザ間を独自の高速多重化通信で結び，オーガナイザからWebサーバに接続するものである．従って，オーガナイザ間のスループット性能がキャッシュサーバの持つ中継スループットを越えていれば，オーガナイザを挿入したことによるボトルネックは生じない．オーガナイザの性能が，遅延の大きな領域だけでなく遅延の小さな領域でもこのスループットを越えたことで，オーガナイザはほとんどのWANの領域で効果を発揮できる．

5. 参考文献

[CISCO] Web Scaling Products and Technologies, http://www.cisco.com/warp/public/751/ .

[HTTP] R. Fielding, J. Gettys, J. Mogul, H. Frystyk and T. Berners-Lee, **Hypertext Transfer Protocol – HTTP/1.1**, RFC2068, 1997.

[ICP] D. Wessels and K. Claffy, **Internet Cache Protocol (ICP), version 2**, RFC2186, 1997.

[IMB] eNetwork Dispatcher, http://www.software.ibm.com/enetwork/dispatcher/ .

[LAT] V. N. Padmanabhan and J. C. Mogul, **Improving HTTP latency**, University of California – Berkeley, Digital Equipment Corporation Western Research Laboratory, 1994.

[MUX] J. Gettys and H. Frystyk, **Simple Multiplexing Protocol**, W3C Working Draft, WD-MUX, 1997.

[NCSA] Thomas T. Kwan and Robert E. McGrath, **NCSA's World Wide Web Server: Design and Performance**, Computing Practices, pp.68-74, IEEE, 1995.

[SLS] W. Richard Stevens, **TCP/IP Illustrated Vol. 1: The Protocols**, Chap. 20, pp.275-296, Addison-Wesley,1994.

[SQ] **Squid Internet Object Cache**, http://squid.nlanr.net/Squid/ .

[SWS] **Sun Web Server 1.0**, http://www.sun.com/solaris/webserver/ .

6. 付録

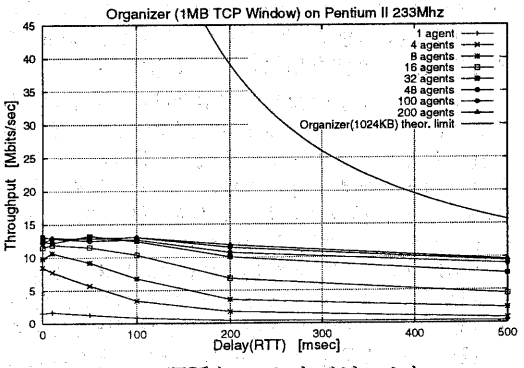

図 6.1 遅延と Web オブジェクトオーガナイザのスループット

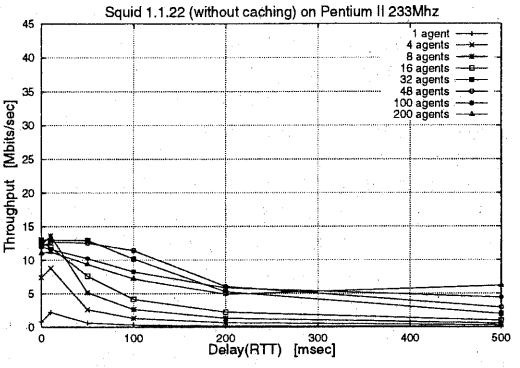

図 6.2 遅延とプロキシサーバ Squid のスループット

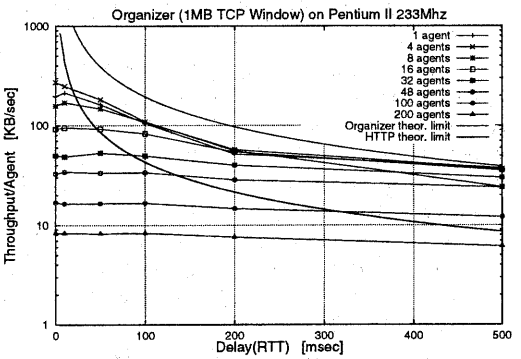

図 6.3 1 エージェント当たりの Web オブジェクトオーガナイザのスループット

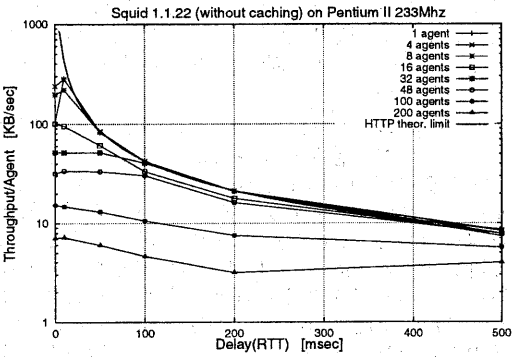

図 6.4 1 エージェント当たりのプロキシサーバ Squid のスループット