WWWにおけるJava Appletの性能評価実験

WWWにおけるJava Appletの性能評価実験

その他（別言語等）のタイトル	An Experiment to Estimate Efficiency of Java Applet on WWW
著者	渡辺　操, 畑中　雅彦, 田島　和典, 青木　貴
雑誌名	室蘭工業大学紀要
巻	48
ページ	75-80
発行年	1998-11-13
URL	http://hdl.handle.net/10258/174

室工大紀要第48号(1998)75～80

WWWにおけるJava Appletの性能評価実験

渡辺　操[*1]，畑中　雅彦[*1]，田島　和典[*2]，青木　貴[*2]

An Experiment to Estimate Efficiency of Java Applet on WWW

Misao WATANABE, Masahiko HATANAKA, Yasunori TAJIMA and Takashi AOKI

(原稿受付日　平成10年5月8日　　論文受理日　平成10年8月31日)

Abstract

Today, in The Internet, there are many accesses from indefinite client machines to a World Wide Web (WWW) server, at a time. If the WWW server accepts the requests from clients in traditional ways such as Common Gateway Interface (CGI) and/or Server Side Include (SSI), the load of this server is increased in proportion to the number of client machines. To reduce the load of server, there is distributed processing approach: the WWW server sends applet code with some data to the client instead of processing its request directly, and the applet code runs on this client to do its own request. We have studied to evaluate efficiency of applet code on WWW environment in comparison with SSI code. In this paper, we report an experimental result which shows that distributed processing approach using small portable applet is more promising than simple server-clients method using its native SSI code, for Intranet environment such as a laboratory network.

Key words : WWW, Common Gateway Interface, Server Side Include, Applet, Java

1. はじめに

今日，電子メールや World Wide Web (WWW) に代表されるインターネット (The Internet) サービスは社会に広く・深く浸透しており，現代人にとって必要不可欠なものになりつつある．特に，マルチメディア情報の公開・交換・収集に優れている WWW 技術は，不特定多数を対象とするインターネットのみならず，組織内部の情報流通・公開を目的としたイントラネット (Intranet) [1] システムにも利用されている．

WWW では，情報を保持し提供する Web サーバと情報を入手して表示する Web クライアント (ブラウザ browser) が基本構成要素となるが，情報検索も含めたより高度な情報提供を可能にするために，Web サーバの背後にデータベース・サーバを配置するシステムが急増している [2]．このようなシステムでは，Web クライアントからの要求を適切に処理してデータベース・サーバに渡したり，データベース中のデータを Web クライアントが表示できるように変換して Web サーバに渡すなどの処理が必要になる．従来これらの処理の大部分は，CGI (Common Gateway Interface) [3] や SSI (Server Side Include) [4] 等の外部プログラムにより Web サーバ側で行われてきた．しかし，インターネットに代表されるネットワーク・システムの普及により，同一の Web サーバに多数のブラウザからの要求が同時に発生する頻度が増えており，サーバの負荷の増大に起因するシステム障害が問題になってきている．特に，CGI や SSI 方式では追加処理をサーバ側で行なうため，サーバの安全性と高速性の維持は難しくなる傾向にある．

[*1] 情報工学科

[*2] ニッテツ北海道制御システム(株)

一方，WWW の対話型アプリケーション開発言語として登場した Java は，Web クライアント上で動作するプログラム (Applet) を実現させた．CGI や SSI で行っていた処理の大部分を肩代わりする Applet をブラウザに送り込めば，これらの処理は各ブラウザ上で行われるのでサーバの負荷が軽減する可能性がある [5]．しかし，プラットホーム独立性を重視した Java はインタープリタ言語であり，Applet の処理速度は高くない等の欠点も有している．

そこで我々は，WWW を対象に Applet による (ブラウザ上での) 分散処理の特性および性能について，サーバ側での集中処理とみなせる SSI との比較により評価する実験を検討した．

現実の WWW 環境で要求される処理は，動画や音声データをも含めたマルチメディアに関する複雑で多様な処理へと移行しつつある．しかし，これら複雑な処理を直接対象として，サーバ集中型処理方式とクライアント分散処理方式に関する再現性のある基本的な特性を得ることは困難と考え，今回は，データ処理の基本操作の一つである整列を題材に実験を行ったので，その結果について報告する．

2. SSI 方式と Applet 方式の概要

サーバ側での処理方式として CGI 方式もあるが，実行途中で利用者の入力作業を必要とするなど処理時間の計測が煩雑なので，今回の実験では SSI 方式のみを採用した．

SSI では，外部プログラムはサーバ上の HTML ファイルから起動され，処理結果はサーバ上の HTML ファイルに反映される [4]．ブラウザは処理結果を含んでいる HTML ファイルを受け取って表示するだけであり，殆どの処理はサーバ側で行われる (図 1 参照)．

Applet 方式では，ブラウザが受け取った HTML ファイルから外部プログラム (Applet) の送付要求が出され，Web サーバから受け取った外部プログラムをブラウザ上で動作させて，処理結果を得て表示を行なう (図 2 参照)．即ち，HTML ファイルと外部プログラムは，サーバから個別に送信される [6]．

3. 実験に使用したプログラム開発言語について

Applet 作成に使用する Java 言語環境では，ソースコードは，一度コンパイラによって中間コード (バイトコード) に変換される．そして実行時に，インタプリタがバイトコードを解釈して，機械語レベルの翻訳作業を行なう．インタプリタを用いることで，処理速度はコンパイラ言語に比べて 10 倍程遅くなる [5]．しかし，最近の Java 言語環境では，インタプリタによる処理速度の低下を抑えるために，実行時にバイトコードの一部をそのマシンのネイティブコードに変換 (コンパイル) する JIT (Just-In-Time) コンパイラが導入されている [7]．

本実験では，JIT コンパイラの有無が Applet の処理速度に与える影響についても調べる．参照実験として，SSI 方式についてもインタプリタ型言語とコンパイラ型言語で作成した同様のプログラムを用いて，性能評価を行なった．今回用いた言語は，前者として Perl を，後者として C を選択した．両言語とも SSI や CGI 用の外部プログラム作成時に，広く用いられているものである．

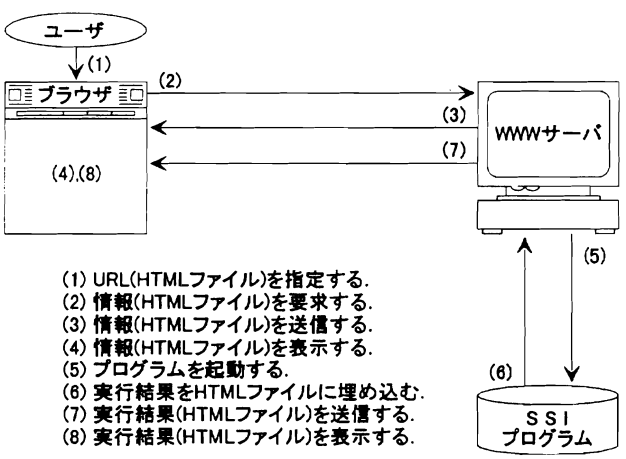

図 1　　SSI を用いた場合の処理の流れ

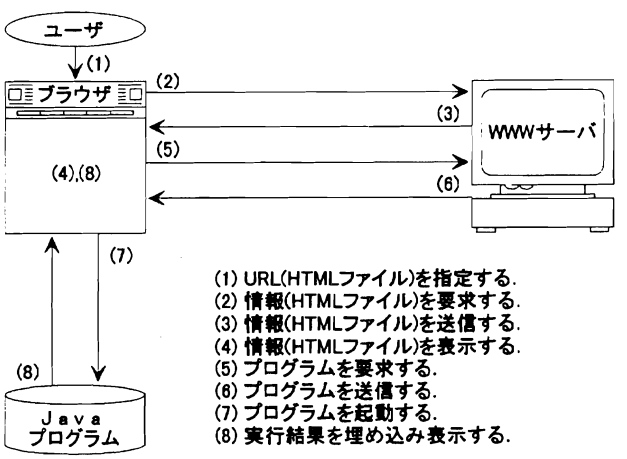

図 2　　Applet を用いた場合の処理の流れ

4. 実験方法と実験システムについて

4.1 SSI 方式と Applet 方式の性能比較実験

性能計測のために用意したテスト用外部プログラム(図3の処理 A) は，ランダムに並んでいる多数の数値データからなる Web サーバ上のファイルを読み込んで，バブル・ソート [8] により整列化 (sorting) を行い，ブラウザ上に整列後の結果を表示させる処理を行なうものである．ファイル上の数値データはテキスト形式で保存されており，そのデータ長はデータ1件当たり 12 Byte で，データ数 n は 100～3000 個とした．SSI 用テスト・プログラムは，C 言語版は GNU C 2.7 を，Perl 言語版は Perl 5.0.0.3 を使って作成した．Applet 用テスト・プログラムは Java 開発環境 JDK 1.1.5 により作成した (表1参照)．

SSI 方式と Applet 方式での処理時間の計測は，図3に示す HTML ファイル・プログラムにより，ブラウザからの要求 (アクセス) 発生直後の時刻と処理終了直前の時刻から，処理時間を算出した．

実験に用いた計算機の主な緒元とネットワークの構成を，表1，2および図4に示す．一部高速イーサネット (100 Mbps) が存在するが，10 Mbps の通常のイーサネットが基本となっており，全ての計算機はハブまたはスイッチング・ハブでネットワークに接続されている．図4の UNIX 計算機 (S) がサーバ・マシンであり，Web サーバとして Apache [9] が稼動している (表1参照)．図4の (a)～(g) の7台の Windows 計算機がクライアント・マシンであり，ブラウザ・ソフトとして Netscape 社の Communicator が動作している (表2参照)．

Applet 実行時の JIT コンパイラ使用有無の選択は，Communicator が使用する JIT コンパイラ用ライブラリ (DLL) の追加・削除によって実現した．計測は5回繰り返し行ない，その平均値を計測値とした．また，

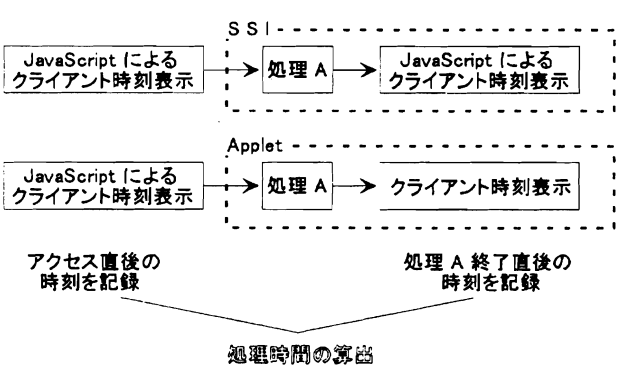

図3　処理時間計測用 HTML ファイルの構成

表1　WWW サーバ・マシンの諸元

	CPU /Clock(MHz)	RAM	OS	Software
(S)	μSPARC-II / 70	48MB	Solaris 2.51	Apache 1.1.3 GNU C 2.7 Perl 5.0.0.3 JDK 1.1.5

表2　クライアント・マシンの諸元

	CPU /Clock(MHz)	RAM	OS	Software
(a)	Pentium-II / 300	64MB	Windows 95	Netscape Communicator 4.04
(b)	AMD-K6 / 200	64MB		
(c)	PentiumMMX / 166	32MB		
(d)	PentiumMMX / 120	32MB		
(e)	Pentium / 133	24MB		
(f)	Pentium / 100	32MB		
(g)	Pentium / 75	32MB		

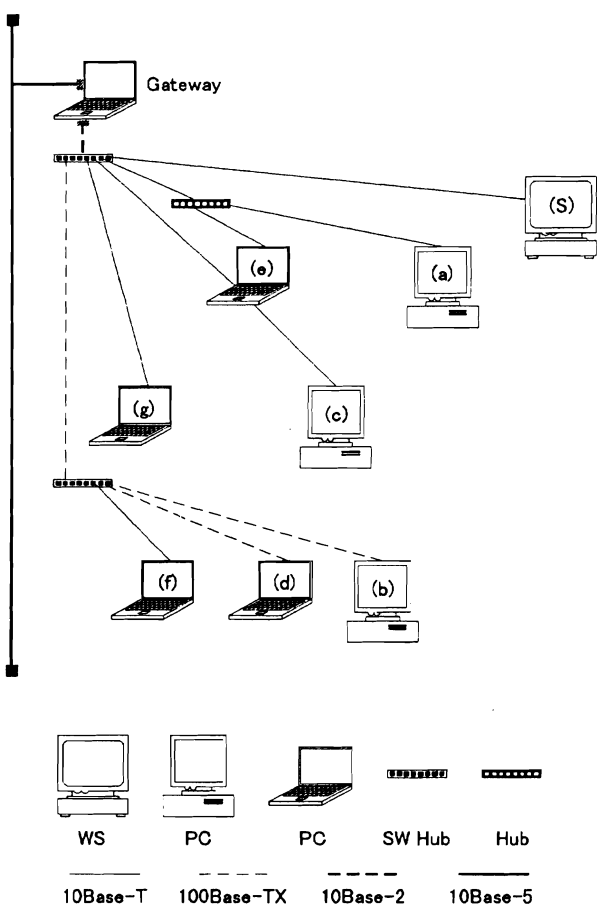

図4　実験に用いたネットワークシステム

Communicator が有するキャシュ機能の影響を取り除くため，計測前に必ず毎回キャシュをクリアしながらホーム・ページをリロードする操作を行った．

4.2 使用したブラウザ・マシンの性能評価実験

上記テスト用 Applet (バブル・ソート) のブラウザ上での処理速度は，各ブラウザ・マシンの性能に強く依存すると予想される．この計測結果とブラウザ・マシンの性能の関係を調べるために，Windows95 用のフリーのベンチマーク・テスト HD Bench 2.6.10 [*1] および Communicator 上で動作するフリーのベンチマーク Applet JMARK 2.0 [*2] を用いて各ブラウザ・マシンのベンチマーク・テストを行った．

5. 結果と考察

5.1 テスト・プログラムの動作確認

図３に示す HTML ファイル内の一連のプログラムの動作確認を兼ねて，Web サーバと表２ (図４) に示したブラウザ・マシン (a) １台の構成におけるバブル・ソート処理時間を計測した．結果を図５に示す．図中の記号で，C, Perl は C 言語および Perl 言語によるテスト・プログラムを用いた SSI 方式での結果，Java-JIT , Java は JIT コンパイラ有り，無しでの Applet 方式での結果である．データ数を n とするとバブル・ソートの時間計算量は O (n^2) であるが [8]，図５の結果を２次関数でカーブ・フィットしたところ非常によく近似できている．このことから，今回作成した計測用プログラム (図３参照) が正常に動作したと判断した．

図３から，処理時間 T は，SSI 方式であるか Applet 方式であるかよりも使用した言語に強く依存しており，コンパイラ型言語の処理時間 (C 言語 $T \sim 3.61 \times 10^{-6} n^2$) は，インタプリタ型言語の処理時間 (Perl 言語の場合 $T \sim 5.69 \times 10^{-5} n^2$，JIT コンパイラ無しの Java 言語の場合 $T \sim 3.61 \times 10^{-4} n^2$) より極端に短くなっている．バイトコードを一部コンパイルする JIT コンパイラ有りの Java 言語の場合は，$T \sim 2.01 \times 10^{-5} n^2$ となり，両者の中間の結果となった．特に C プログラムによる SSI 方式は，圧倒的に高速であった．同じ Applet でも JIT コンパイラの有無により，処理時間に大きな差が生じた．以上の結果より，C と Perl を比較すると，C の方が１６倍，JIT コンパイラ有りの Java と JIT コンパイラ無しの Java を比較すると JIT コンパイラ有りの Java の方が１８倍の処理速度であることが確認できた．JIT コンパイラの効果が判明したので，以後の Applet の実験は JIT コンパイラ有りの条件下でのみ行った．

5.2 Web サーバへの同時アクセス時の結果

次に，Web サーバに同時にアクセスするブラウザ・マシンの数を変化させて，処理時間を計測した．SSI 方式では，全ブラウザからの要求をサーバ側で処理するので，原理的には処理時間はブラウザ・マシンの数に比例して増加する．逆に Applet 方式では，処理は各ブラウザ・マシン上で実行されるので，処理時間はマシン数に依存しないと予想される．ただし，処理時間はブラウザ・マシンの性能に左右される．

図６に，マシン (a) 上で計測したデータ数 n が

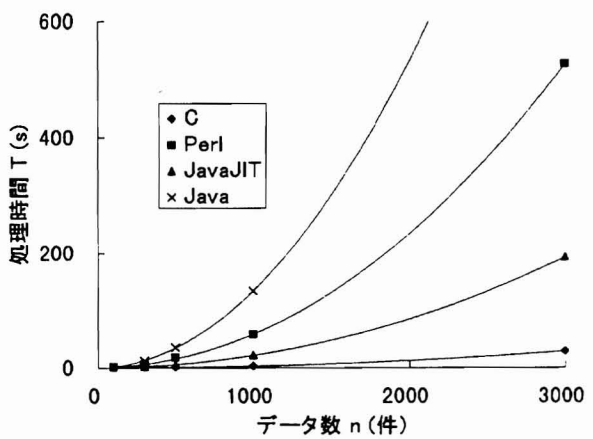

図５　ブラウザ・マシン (a) １台の時の結果
(処理データ数 n とバブル・ソート処理時間)

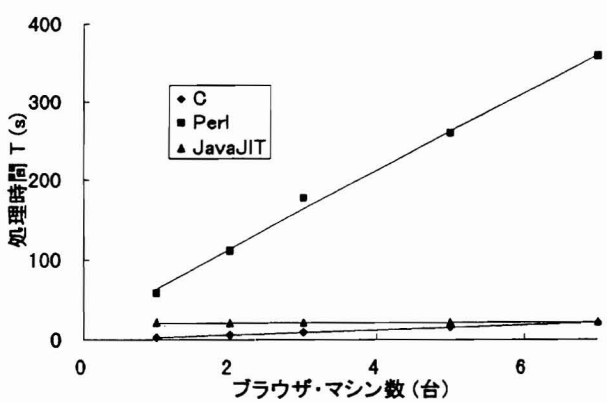

図６　ブラウザ・マシン数と処理時間との関係
(ブラウザ・マシン (a) 上にて計測)

[*1] http://www.lares.dti.ne.jp/~ep82kazu 参照

[*2] http://www.zdnet.com/zdbop/jmark/jmark.html 参照

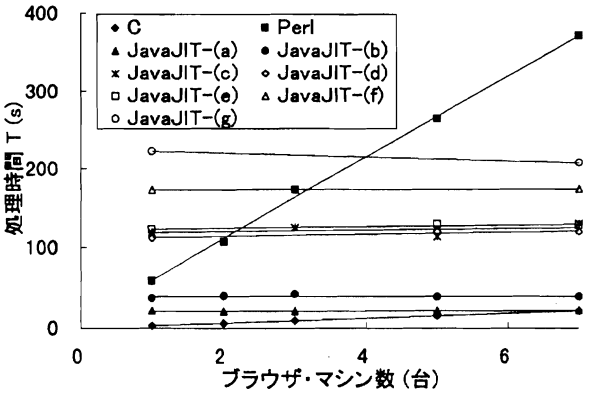

図7　各ブラウザ・マシンの 処理時間の結果

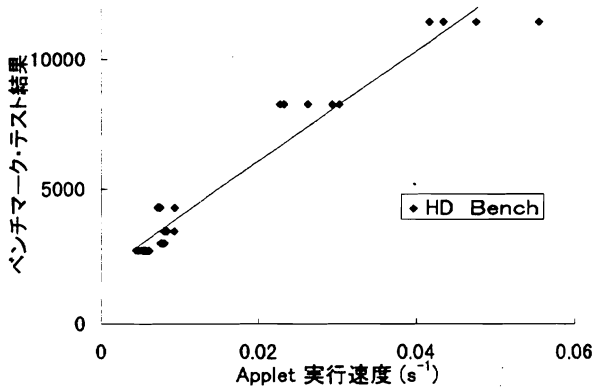

図8　Applet 方式でのバブル・ソート処理速度と HD Bench によるベンチマーク結果との関係

1000 件の場合の処理時間の結果を示す. SSI 方式 (図中の C と Perl) では，予想通り，ブラウザ・マシンの数に比例して処理時間が増加した．直線近似における傾きは, C の場合で 3.04 (s/台), Perl の場合は 52.28 (s/台) であった．これは，図５に示したデータ数 1000 件の時の処理時間 (C の場合は 3.61(s), Perl の場合は 56.9(s)) とよく対応している．図６の中で JavaJIT と示した測定値から, Applet 方式では, 傾きが 0.12 (s/台) とほぼ水平となり，予想通り処理時間はブラウザ・マシンの数に依存しない結果となった．また平均処理時間は 21.08(s) で，図５で得られた値 20.1(s) に近い結果となった．

他のブラウザ・マシン (b)～(g) 上で計測した Applet による処理時間を，図７にまとめて示す．各ブラウザで処理時間は大きく異なるが，同時に要求を出しているブラウザ数には殆ど依存しない結果となった.(ブラウザ・マシン (a)～(g) の傾きの平均は 0.89 (s/台) で，ほぼ水平である．)

この実験結果から．C 言語プログラムを利用した SSI 方式と JIT コンパイラ付 Applet 方式における処理時間を評価すると，８台のブラウザ・マシンが同時に SSI 方式サーバにアクセスすると，その処理時間は 3.04×8=24.3(s) となり，マシン (a) クラスの性能を有する Applet 方式での処理時間 21.1(s) と同程度になる．しかし，マシン (b) クラスの性能を有するブラウザ・マシンでは，１３アクセスが同時に発生した時の SSI 方式の処理時間に等しい時間が必要であり，最も遅い (g) クラスのマシンでは，同時アクセス数が７０程度ある SSI 方式に相当する処理時間が必要となる．

5.3 ベンチマーク・テストの結果

図６, 図７から求めたバブル・ソート Applet の実行速度 (処理時間の逆数) を横軸に，HD Bench にて計測した各ブラウザ・マシンの総合ベンチマークの値を縦軸にとった結果および回帰直線 (相関係数は 0.97) を図８に示す．また，縦軸に JMARK で得られた Processor 性能指標をとった，同様の結果および回帰直線 (相関係数は 0.95) を図９に示す．

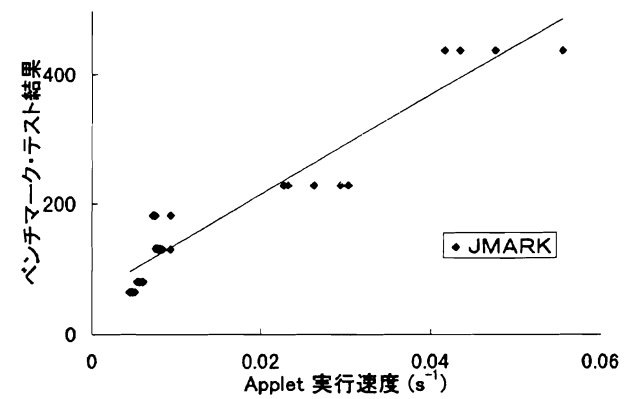

図9　Applet 方式でのバブル・ソート処理速度と JMARK によるベンチマーク結果との関係

測定値にバラツキはあるが，本バブル・ソート Applet の処理速度は，ネイティブコードで書かれたベンチマーク・テスト HD Bench の結果に対しても，良好な比例関係を示した．このことから，本実験条件においては，複数のブラウザ・マシンからの同時アクセスによって生ずる Web サーバの性能特性の変化やネットワークのパケット転送能力の変化はなかったと思われる．

6. まとめ

本研究結果として着目すべき重要事項は，1). コンパイラ言語とインタプリタ言語の実行形態による処理速度の差異，2). Applet を実行する際の JIT コンパイラ使用の有無による性能への影響，3). 複数ブラウザからの同時アクセスに対する WWW サーバにおける SSI 方式の特性，4). 複数ブラウザからの同時アクセスに対する WWW サーバにおける Applet 方式の特性，5). 実行速度から見た Applet の有用性，である．以下，上記の項目についてまとめる．

1). 計算量の大きな処理を伴う機能を Web サーバ側が提供するとき，その外部プログラムの開発言語の選択に注意が必要である．今回用いた数値のソーティングでは，C のプログラムは Perl のプログラムのおよそ１６倍のスピードがあった．しかし，コーディングの容易さ・文字列処理での有利さなど，Perl の有用性が無視できないのは当然のことである．

2). Java もインタプリタ言語であるが，JIT コンパイラを導入することで，およそ１８倍の速度向上が得られた．Java の最新のバージョンでは，更に高速化を目指した HotSpot [10] と呼ばれる技術の導入が考えられており，通常のコンパイラ言語に比べて遜色ない処理速度が得られる可能性もある．

3). SSI 方式の場合，ユーザの要求処理が一つのプロセスであれば，WWW サーバの機能には影響がない．しかし，同時に多数のユーザから処理の要求があれば，その数だけ WWW サーバ上でプロセスが実行される．よって，ユーザの数に比例する形で WWW サーバの負荷は増大する．

4). Applet 方式の場合，ユーザの要求処理は一つのプロセスとして，ブラウザ・マシンで起動した Java のバーチャルマシン上でそれぞれ実行される．したがって，ブラウザ・マシン数が増加しても WWW サーバには影響がないので，常に安定した状態で情報の提供が行なえる．

5). 我々が行った実験環境下では，図６より，Applet 方式でのブラウザ・マシンに要求される性能の目安が (Web サーバの性能) / (同時にアクセスする可能性のあるブラウザの数) で与えられること，１０台程度のブラウザからの同時アクセスがあれば Applet 方式が SSI 方式と同等以上の性能を発揮することが判明した．勿論，この結果はブラウザ・マシンの性能に依存するが，サーバ用マシンのコスト・パフォーマンスの伸びと比べてブラウザ・マシンとしての PC のコスト・パフォーマンスの伸びが著しい現状においては，Applet 方式がより優位になる場合が多いと思われる．

今回使用した Applet はコードサイズの非常に小さなプログラムであったが，処理によっては大型の Applet の配送が必要な場合も考えられる．使用可能なネットワーク帯域に大きく依存するが，この状況下で Applet の配送時間が大きなオーバ・ヘッドとなり，SSI 方式が圧倒的に優位となる場合も考えられる．WWW 技術を利用する現実の処理では，Applet のサイズはこれらの間の値となる場合が多いと想像される．

なお，本研究の要旨の一部は，室蘭工業大学開発技術研究会 [11] にて発表した．

参考文献

[1]. 好川哲人: イントラネットとはなにか？, Open Design, 3, 5, CQ 出版社, (1996) pp.4-15

[2]. 花井浩之: Web と外部プログラムの連携, Software Design, 142, 技術評論社, (1997) pp.18-27

[3]. S.Gundavaram (田辺茂也 監訳): CGI プログラミング, オライリー・ジャパン, (1996) pp.1-55

[4]. S.Gundavaram (田辺茂也 監訳): CGI プログラミング, オライリー・ジャパン, (1996) pp.95-110

[5]. 有我成城, 衛藤敏寿, 佐藤治, 他: Java 入門, 翔泳社, (1996) pp.1-30

[6]. 小松憲一: whatis Java, UNIX USER, 5, 6, ソフトバンク, (1996) pp.32-37

[7]. Software Design 編集部: 保存版・Java 用語辞典, Software Design, 143, 技術評論社, (1997) pp.74-85

[8]. 近藤嘉雪: C プログラマのためのアルゴリズムとデータ構造, ソフトバンク社, (1992) pp.185-193

[9]. 鐙聡: 各種 WWW サーバ・ソフトウェアをインストールする, Interface, 22, 6, CQ 出版社, (1996) pp.104-124

[10]. 松田慎一: 最新 Java 事情～ JDK1.2 の新機能, JAVA PRESS, 1, 技術評論社, (1998) pp.2-19

[11]. 渡辺操, 小林亮, 畑中雅彦, 田島和典: Java によるデータベース利用の総合研究環境の構築について, 平成 9 年度室蘭工業大学開発技術研究会, pp.21-22 (1997)