システムソフトウェアと
オペレーティング・システム 84−23
（２０００．５．２５）

命令トレーサを用いたネットワークサーバプログラムの性能評価

毛利 公一[†] 森本 洋行[††] 池谷 敬之[†††] 吉澤 康文[†]

[†]東京農工大学工学部
[††]東京農工大学大学院工学研究科
[†††](株) 日本トータルシステム

ネットワークサービスを提供するサーバマシンは，クライアントからの要求の受付けと応答をスムーズに処理することが要求される．この要求を満たす方法として，サーバマシン上で動作させるプログラムのオーバヘッドを軽減する方法がある．本論文では，我々が開発した命令トレーサ鶴を用い，WWW サーバ (httpd) の性能を定量的に評価する．性能評価によって，4.6%のオーバヘッド軽減が可能であることを示した．また，評価に基づき，オーバヘッドの軽減手法の指針を示した．

Performance Evaluation of Network Server Programs using Instruction Tracer

Koichi Mouri[†] Hiroyuki Morimoto[††] Hiroyuki Ikeya[†††]
Yasufumi Yoshizawa[†]

[†]Faculty of Engineering, Tokyo University of Agriculture and Technology
[††]Graduate School of Engineering, Tokyo University of Agriculture and Technology
[†††]Japan Total Systems Corporation

Server machines which provide network services are required to process requests from clients smoothly. To achieve the demands, overheads of programs which are executed on the server machines must be reduced. In this paper, an overview of the instruction tracer "TURU" which we developed is described. Furthermore, we evaluate the performance of WWW server(httpd) using the TURU. The evaluation shows that 4.6% of execution time can be reduced in memory management. We also indicate ways which are based on the evaluation to reduce the overheads.

1 はじめに

WWWサーバを始めとする，ネットワークサービスを提供するサーバマシンは，インターネット上に存在する多数のクライアントからの要求の受付けと応答をスムーズに処理することが要求される．このような，多くの要求を処理する場合，1つの要求の処理を短くすることがサーバマシンおよびサーバプログラムにおける目的の1つとなると言える．処理を短くするためには，サーバマシンの処理速度を向上させる方法と，サーバマシン上で動作させるプログラムのオーバヘッドを軽減する方法がある．我々は，特に後者の方法に注目し，サーバプログラムの性能の評価を行うことを目的としている．

プログラムの性能は，それ自体の命令数，入出力の回数や時間，メモリ参照の回数やその参照先などから求めることができる．しかし，サーバプログラムなど，アプリケーションの性能評価においては，具体的にどれだけの命令を単位時間当りに処理することが可能であるかという性能を定量的に示すことが困難である．これは，計測対象の性能に，アプリケーションだけでなく，オペレーティングシステム(以下，OSと記す)の性能も含まれるためである．特にOSの性能評価手法に関しては，アプリケーションなどのユーザモードのプログラムがOSの処理内容を参照することが不可能であることと，割込みなどの非同期イベントが発生することなどの理由により，性能を定量的に評価することは困難となっている．これらの問題点を解決するために，我々は，カーネル組込み型の命令トレーサ“鶴[1]”を開発した．鶴は，性能評価に必要となる情報を出力することができる．さらに，そのデータを解析するための解析ツール群を提供している．

本論文では，命令トレーサ鶴を用い，ネットワークサービスとして多く用いられているWWWサーバ(httpd)の性能を評価する．鶴と解析ツールを用いることによって，プログラムの命令数，命令が配置されたメモリ，データとして参照されるメモリのページ数，処理に必要なサイクル数を求めることができる．本論文では，特に，httpdがHTMLファイルを転送する場合と，CGIプログラムを実行する場合についての評価を行う．

以下，本論文では，2章で命令トレーサの概要について述べる．3章では，httpdの性能評価について述べ，4章で性能評価に対する考察を述べる．最後に，5章で本論文のまとめを述べる．

2 命令トレーサ“鶴”

OS内部の処理を把握したり，評価することは，OSの目的と構成上困難である．これらの理由として，OSは，仮想的に高機能な計算機を作り出すことを目的としていることが挙げられる．また，OSが独立して計算機システムを構成しているのではなく，アプリケーションと密接な関係を持って成り立っているためである．さらに，割込みなどの非同期イベントが発生するため，プログラムソースを参照するだけでは制御の流れを追跡できない場合もある．上記を解決するために，我々は，評価と解析に適した命令トレーサ鶴を開発した．以下鶴の概要を述べる．

2.1 鶴の概要

鶴は，ソフトウエアで実現し，特別なハードウェアの追加を必要としない．このような命令トレーサには，カーネル組込み型とユーザプロセス型の2つの手法がある．カーネル組込み型は，カーネル内部に命令トレーサを組み込む手法であり，OS自身のトレースや，割込み処理をトレースすることが可能である．ユーザプロセス型は，1命令ごとに例外割込みを発生させ，命令トレーサプロセスを起動する仕組みをカーネル内部に組み込む手法である．命令デコード，記録レコードの作成やデータの保存は，ユーザプロセスとして実現されたサーバプロセスが行う．しかし，この手法は，実現が容易であるが，スケジューラや割込み禁止区間などの部分をトレースすることができない．鶴は，OSとアプリケーションをトレースするこ

とが目的であるため，カーネル組込み型を採用している．

鶴は，プログラムの実行を機械語レベルで命令をトレースする．記録する情報を次に示す．

- 機械語命令列
- 参照したメモリアドレス
- 割込み
- システムレジスタの変更

プログラムのダイナミックステップ数を計測することにより，リアルタイム性の評価，サーバプログラムにおける性能評価オーバヘッドの大きな部分の特定や改善が可能となる．さらに，対象CPU アーキテクチャにおける命令の使用頻度分布や，平均命令長を求めることも可能となる．

メモリ参照は，CPU 処理と比較して低速である．参照したメモリアドレスを取得することにより，メモリ参照の局所性や，そのアクセスパターンを調べることができ，効率的なメモリ配置を求めることができる．

割込みは，カーネル処理の中でも発生タイミングを特定することが困難な部分である．また，リアルタイム性を保証するためのデータとして，割込み禁止区間のダイナミックステップ数を取得可能であることは重要である．

システムレジスタには，特権レベルや割込みの許可や不許可などの状態が格納されている．また，ページフォルトが発生したときのアドレスなどの情報もシステムレジスタに記録される．システムレジスタを取得することによって，CPU の状態を示すこれらの値を取得することができる．

2.2 鶴の構成

鶴は，Intel i386 アーキテクチャ上で動作するLinux(Ver.2.0.34) カーネル内に実装されている．鶴の構成は，図 1に示すように，シングルステップ実現部，トレースデータ生成部，データ出力部の 3 つの部分から構成される．トレース対象のプログラムの機械語命令が実行される度に，鶴へ制御が移行する．鶴は，次に実行される命令列を解析し，記録データとして出力する．

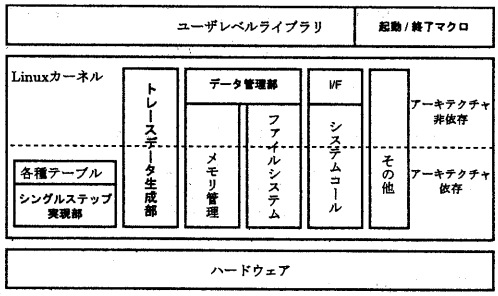

図 1 鶴の構成

(1) シングルステップ実現部

1 つの命令が実行されるごとに，鶴に制御を移行させる機構を提供する．

(2) トレースデータ生成部

対象プログラムの命令トレースと，割込みの発生，システムレジスタの変更を監視する機構を提供する．

(a) 命令デコード部

次に実行される命令のデコードを行い，命令長，オペランドの個数などの判定，セグメント内オフセットアドレスと，リニアアドレスの取得を行う．さらに，命令レコードとして記録する．

(b) 割込みイベント処理部

割込みが発生した際に，割込みイベントレコードを記録する．

(c) システムレジスタ監視部

システムレジスタに変更があった場合に，システムレジスタレコードを記録する．

(3) データ出力部

(a) 静的メモリ領域への保存

鶴が確保したピンダウンされたメモリ領域にトレースデータを出力する．鶴終了後に，コマンドによって，ハードディスクへのデータ保存を行うことができる．

(b) ファイルシステムへの保存

トレース実行時に逐次，ファイルシステムへトレースデータを出力する．

2.3 評価ツール

鶴の出力したトレースデータを解析するために提供されているツール群を以下に示す．

(1) 各レコードデータ表示ツール

各レコードの値を表示する．このツールを基本として，他のツールを作成することができる．

(2) 関数遷移表示ツール

鶴起動時に取得したシンボルテーブルと各命令の仮想アドレスの対応を調べ，C言語の関数を単位とした関数遷移の様子を表示する．

(3) メモリアクセス分布表示ツール

各命令のリニアアドレスを元に，メモリアクセス分布を表示する．

上記のツール群は，アプリケーションとして実現される．すなわち，ユーザの目的に応じて自由に拡張したり，新規作成することが容易となっている．

3 HTTP サーバの性能評価

2章で述べた命令トレーサを用いてhttpdをトレースした．トレースは，次に示す2つの場合について行った．

- HTMLファイルを転送した場合
- CGIプログラムを動作させた場合

以下，それぞれの場合における性能評価について述べる．ただし，性能評価では，ハードウェアとして，Intel Pentium 100MHz，メモリを96MB搭載したマシンを用いた．また，ソフトウェアとして，OSにLinux(Ver.2.0.34)カーネル，httpdにapache-1.3.12を用いた．

3.1 HTMLファイルの転送

3.1.1 評価方法

トレースは，クライアントからのTCPコネクションの確立待ちの箇所から，TCPコネクションを切断するまでを対象として行った．その間httpdは，大まかに以下の処理を行う．

(1) クライアントからのTCPコネクションを確立する

(2) クライアントからの要求を受取り，解釈する

(3) 図2で示すHTMLファイルを読み込み，クライアントに送信する

(4) TCPコネクションを切断する

```
<HTML>
<HEAD>
   <META HTTP-EQUIV="Content-Type"
         CONTENT="text/html;
                  charset=Shift_JIS">
</HEAD>
<BODY>
<CENTER>AA</CENTER>
</BODY>
</HTML>
```

図2 HTMLのソースファイル

3.1.2 評価結果

トレースの結果，httpd，共有ライブラリ，カーネルの3つの各部分におけるダイナミックステップ数は表1に示す通りとなった．この処理を実行

表 1　HTML の転送におけるステップ数

httpd	共有ライブラリ	カーネル
90,655 (29.9%)	117,204 (38.6%)	95,720 (31.5%)

表 2　HTML の転送における使用ページ数

	httpd	共有ライブラリ	カーネル
命令	48	37	45
データ	150	268	466

表 3　HTML の転送におけるページの参照回数

	httpd		
	最小	最大	平均
命令	13	15,825	1,888
データ	1	5,575	106
	共有ライブラリ		
	最小	最大	平均
命令	14	35,305	3,167
データ	1	33,389	352
	カーネル		
	最小	最大	平均
命令	31	16,952	2,127
データ	1	5,175	60

するためのサイクル数は 817,599 サイクルで，処理時間は評価環境では 8.18ms となる．

また，httpd，共有ライブラリ，カーネルにおいて，命令として参照されたページ数と，命令のオペランドでデータとして参照されたページの数を表 2に，ページが参照された回数の最小，最大，平均を表 3に示す．

3.2　CGI プログラムの実行

3.2.1　評価方法

httpd で CGI プログラムを呼び出した場合のトレースを行った．トレースは，クライアントからの TCP コネクションの確立待ちの箇所から，TCP コネクションを切断するまでを対象として行った．その間 httpd は，大まかに以下の処理を行う．

```
#include <stdio.h>

int main(int argc, char **argv)
{
    puts("Content-type: text/html\n\n"
         "Hello World.");
    return(0);
}
```

図 3　CGI プログラムのソースコード

(1) クライアントからの TCP コネクションを確立する

(2) クライアントからの要求を受取り，解釈する

(3) 図 3で示す CGI プログラムを実行し，その結果をクライアントに送信する

(4) TCP コネクションを切断する

3.2.2　評価結果

トレースの結果，httpd，CGI，共有ライブラリ，カーネルの 4 つの各部分におけるダイナミックステップ数は表 4に示す通りとなった．この処理を実行するためのサイクル数は 2,810,292 サイクルで，処理時間は評価環境では 28.1ms となる．また，トレース中における仮想アドレス空間切替え (CR3 レジスタの変更) 回数は 10 回であった．

httpd，CGI，共有ライブラリ，カーネルにおいて，命令として参照されたページ数と，命令のオペランドでデータとして参照されたページ数は表 5，ページが参照された回数の最小，最大，平均は，表 6の通りとなった．

4　考察

Pentium では，Translation Lookaside Buffer (以下，TLB と記す) は，命令用とデータ用の 2 つが独立して提供されている．それぞれのエントリの数は，命令用 TLB が 32 エントリ，データ用 TLB が 64 エントリである．TLB は，プログラムによって指定したエントリをフラッシュすることができる．アドレス空間が切替えられたとき，

表 4　CGI の実行におけるステップ数

httpd	CGI	共有ライブラリ	カーネル
68,023 (4.0%)	15,885 (0.9%)	991,342 (58.5%)	620,139 (36.6%)

表 5　CGI の実行における使用ページ数

	httpd	CGI	共有ライブラリ	kernel
命令	47	16	65	71
データ	330	1,629	2,911	2,478

表 6　CGI の実行におけるページの参照回数

	httpd		
	最小	最大	平均
命令	10	13,142	1,447
データ	1	4,450	71
	CGI		
	最小	最大	平均
命令	20	7,615	1,134
データ	1	6,801	24
	共有ライブラリ		
	最小	最大	平均
命令	2	795,477	15,251
データ	1	250,317	145
	カーネル		
	最小	最大	平均
命令	5	99,877	8,734
データ	1	17,128	63

すなわち CR3 レジスタの値が変更されたときは，自動的にフラッシュされる．また，TLB は，ミスした場合のコストが大きく [2]〜[4]，ヒットした場合よりも 9〜13 サイクル余分なサイクルが必要となる．

HTML ファイルの転送では，命令が 130 エントリ，データが 884 エントリの TLB を使用する．すなわち，TLB のミスのために 9,126〜13,182 サイクルが必要となる．これは，全体の 1.1〜1.6% となる．また，CGI プログラムを実行した場合は，命令が 199 エントリ，データが 7,348 エントリの TLB を使用し，10 回のアドレス空間の切替えが発生している．すなわち，TLB ミスのために，少なくとも 67,923〜98,111 サイクルが必要になる．これは，全体の 2.4〜3.4% となる．

TLB のミスの他に，メモリに関するオーバヘッドとして，セグメントの切替えがある．Pentium では，セグメントの切替えに 39 サイクルを必要とする．セグメントレジスタは 6 種類あるため，234 サイクルとなる．Linux では，カーネルと個々のプロセス毎にセグメントを設定している．これは，システムコールが発行された場合や，プロセスの切替えが発生した場合にセグメントの切替えが発生することを意味する．

HTML ファイルの転送の場合，システムコールは 48 回発生している．これは，11,232 サイクルを要する．また，CGI プログラムを実行した場合，システムコールが 127 回発生している．さらにプロセスの切替えが 10 回発生している．これは，32,058 サイクルを要する．これを，TLB のミスと合わせると，全体のサイクルに占める割合は，前者で 2.4〜2.9%，後者で 3.5〜4.6% となる．

Pentium 100MHz を用いた場合，以上のオーバヘッドを軽減することによって，HTML ファイルの転送においては，1 秒当り 122 回の要求を 126 回まで，CGI プログラムの実行においては，1 秒当り 35 回の要求を 1 秒当り 37 回まで処理回数を増やすことが可能となる．

また，共有ライブラリとカーネルのダイナミックステップ数は，HTML の転送では，表 1より，全命令の 70.1%を占める．CGI プログラムの実行では，表 4より，全命令の 95.1%を占める．共有ライブラリを広義の OS と考えた場合，OS の設計がアプリケーションの性能を大きく左右することがわかる．

さらに，表 2と表 5より，命令よりもデータにおいて参照されるページ数が多いことがわかる．表 3と表 6より，データが配置されたページの参照回数は，命令が配置されたページの参照回数よりも少ない．すなわち，命令の参照の局所性よりも，データの参照の局所性の方が低い．性能を向上させる場合，データの参照の局所性を向上させることが重要であることが示されている．

特定のサービス専用のサーバを開発するなど，性能を向上させる場合，サーバプログラムが使用

する共有ライブラリの関数やシステムコールを特定する必要がある．使用しない関数やシステムコールを削除し，使用する命令やデータを少数のページにまとめることによって，参照するページ数を軽減することができる．また，削除しない場合においても，参照するページの数を抑えるようなメモリ配置にする手法を採ることができる．

5 おわりに

本論文では，我々が構築した命令トレーサ鶴の概要を述べた．これによって，特別なハードウェアを用いることなく，種々の解析の元となるデータの収集を行うことが可能となった．さらにOSも含めた解析を行うことが可能となった．

さらに，WWWサーバを例として，HTMLファイルを転送した場合と，CGIプログラムを起動した場合の性能を評価した．特に，ダイナミックステップ数，命令とデータにおける参照ページ，サイクル数を計測し，これらを基にTLBのフラッシュ，セグメントの切替えによるオーバヘッドを示した．さらに，これらのオーバヘッドを軽減することで，処理速度を4.6%向上させることができることも示した．また，命令とデータにおけるページの参照回数の分散を示すことによって，プログラムの処理効率を向上させるための指針を示した．

今後は，キャッシュも解析の対象とすることで，性能評価の精度を向上させる予定である．また，WWWサーバ，共有ライブラリ，OSの3つの要素において，どの部分を変更することによって，性能をどれだけ向上させることが可能となるか，性能評価手法と性能向上の手法に関して研究を進める予定である．

参考文献

[1] 池谷 敬之，吉澤 康文，“性能評価のための命令トレーサの開発，” 第58回全国大会講演論文集，Vol. 1，pp. 123–124，情報処理学会，1999.

[2] 毛利 公一，大久保 英嗣，“マイクロカーネルLavenderの設計と開発，” 電子情報通信学会論文誌，Vol. J82–D–I，No. 6，pp. 730–739，1999.

[3] Hermann Härtig, Michael Hohmuth, Jochen Liedtke, Sebastian Schönberg, Jean Wolter, “The Performance of μ-kernel-Based Systems, ” Operating System Review 1997 December, pp. 66–77, 1997.

[4] Jochen Liedtke, “On μ-Kernel Construction, ” 15th ACM Symposium on Operating System Principles, pp. 237–250, 1995.