Linuxシステムの解析を可能とする命令トレーサ:TURUNGの開発

内藤　義郎†　　小宮山　彰一郎†　　吉澤　康文‡
東京農工大学大学院工学研究科†　　東京農工大学工学部‡

1. はじめに

サーバに特化した計算機システムの性能解析では,主要なプログラムがCPU，メモリ,I/Oの各資源をどのように使用しているかを正確に把握する必要がある．しかし，OSの内部処理を把握したり，評価することは，OSの目的と構成上困難である．OSの提供する仮想化，抽象化はOSの内部構造を隠蔽するため,その処理解析や性能評価を困難にする．

そこで我々は,Linuxシステムのカーネル及び全ユーザプロセスの処理を機械語レベルで記録する命令トレーサ:TURUNGならびに解析ツール群:TURUNG Toolsを開発した．これらを用いることでOS及びAPを含めたLinuxシステムの総合的な分析を行うことが可能となる．

2. TURUNGの概要

TURUNGの開発における最大の目的はLinuxシステムの統合的な分析を行うための環境を提供することである．上記の目的を実現するためにTURUNGが実現した具体的な機能を次に示す．

（1）機械語命令トレース

TURUNGが提供する最も基本的かつ重要な機能は,ユーザ空間上で実行されている全てのアプリケーション(以下APと記す)とオペレーティングシステム(以下OSと記す)の機械語命令の記録である．これにより，APの流れ，そこから呼び出されるOSの処理，割り込みの制御の発生による制御の移行を,時系列に従って一連の流れとして解析することができ，APとOSの統一的な性能評価が可能となる．

（2）メモリアクセス解析

TURUNGでは,命令トレース時に命令のメモリアクセスを監視し，アクセスのあったアドレスを記録する．この機能により，実行時のより詳細なメモリアクセスの分析が可能となる．またOSとAPにおけるページの参照回数やページの使用数などの統計分析も行うことができる．

（3）例外，割り込み,システムコールの記録

TURUNGでは例外,割り込み,システムコールの種類と発生時刻を記録する．システムコールの記録ではシステムコール番号と共に引数も記録する．また，ページフォールトに関しては，その理由と発生アドレスも記録する．

（4）Loadable Kernel Module(LKM)化

TURUNGのLKM化により必要なときにTURUNGを組み込むことが可能となる．またTURUNGはデバイスドライバとして実装されているので，ユーザはデバイスドライバ標準のアクセス手段で簡単にTURUNGを利用することができる．

（5）TURUNG ToolsとTURUNG Library

TURUNGによって書き出されたトレースデータファイルを解析するためのツール群:TURUNG Toolsと,その開発者向けの共有ライブラリである TURUNG Library を提供する．TURUNG Toolsを使えば，次のようなものが表示可能となる．

- 関数遷移とダイナミックステップ数
- 例外/割り込みの統計
- 命令の逆アセンブル
- ページの参照頻度と使用数
- 使用システムコールの統計

3. TURUNGの構成と実装

TURUNGの全体構成を図1に示す．TURUNGは大きく分けて，TURUNG Core，TURUNG Library,TURUNG Tools の3つの部分から構成される．TURUNG CoreはLinuxカーネル内に実装され,トレースデータの収集を行う一番重要なモジュールである．これはIntel x86 アーキテクチャ上で動作し，Linux カーネル2.4.18に実装されている．TURUNG LibraryとTURUNG Toolsは前述した通り，トレースファイルを解析するためのユーザ空間で動作するプログラムである．

3.1. シングルステップ割り込み

Intel x86 アーキテクチャが提供するデバッグ支援機能の一つである．TURUNGの基本動作は図2に示すように命令境界で発生するシングルステップ

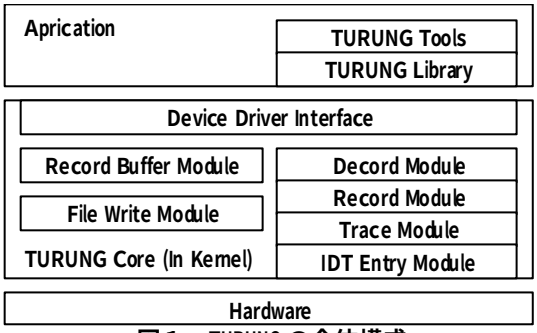

図1　TURUNGの全体構成

An Instruction Tracer for analyzing Linux System(TURUNG)
†Yoshiro NAITO, †Syoichi KOMIYAMA and ‡Yasufumi YOSHIZAWA
†Graduate School of Engineering, Tokyo University of Agriculture and Techology
‡Faculty of Engineering, Tokyo University of Agriculture and Technology

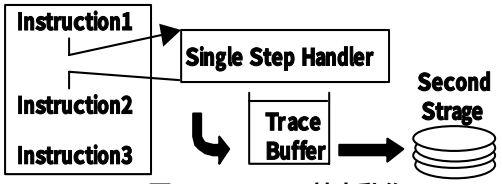

図２ TURUNG の基本動作

```
int fd;
struct turung_info tinfo;
fd = open("/dev/turung", O_RDONLY);
TURUNG_DEF(tinfo);
ioctl(fd, TURUNG_TRACE_ON, &tinfo);
      /* Trace Program */
ioctl(fd, TURUNG_TRACE_OFF);
close(fd);
```

図３ トレースプログラムの作成

割り込みを捕捉して,命令の記録を行う．このとき TURUNG は再帰的な自己トレースを防ぐために割り込み禁止で動作する．

3.2. シングルステップ例外の発生

TURUNG は次のようにして，全ての命令にシングルステップ例外を発生させる．

（１）現在の CPU の EFLAGS のトラップフラグ(TF)をオンにする．

（２）全プロセスのタスク構造体をたどり，そこからカーネルスタックに保存されている EFLAGS イメージの TF をオンにする．これにより全てのユーザプロセスでシングルステップ例外が発生する．

（３）シングルステップ例外以外の例外および割り込みで TF をオンにする．これにより，割り込みのトレースが可能になる．

3.3. IDT の置き換え

TURUNG はトレース開始時に元の割り込みディスクリプタテーブル(IDT)を保存して，IDT を TURUNG のトレースモジュール呼び出すための独自の IDT に置き換える．TURUNG は割り込みが入るとその IDT を経由して TURUNG トレースモジュールに入ることができる．シングルステップ例外の場合は命令を記録してそのまま iret 命令で処理を終了する．他の例外または割り込みの場合は割り込みレコードを記録した後 TF を ON にして元の割り込みハンドラにジャンプして命令トレースを開始する．

3.4 TURUNG によるトレースの開始と終了

トレースの開始は図３に示すように TURUNG デバイスをオープンしたあと，TURUNG に渡すパラメータを初期化して，ioctl 関数でトレースの開始と終了を TURUNG に指示する．TURUNG はトレースの開始時にはトレースバッファを確保し，スタックの TF をオンにする．その後元の IDT の保存と TURUNG 独自の IDT への置き換えを行う．トレース終了時には TF をオフにして IDT を復旧した後，トレースファイルを 2 次記憶装置に書き出す．

3.5 TURUNG Library

TURUNG ライブラリは主に二つの部分から構成さ

```
<!Exception! type="Page-Fault Exception">
<#page_fault#/>
<do_page_fault>
  <find_vma = 18/>
  <handle_mm_fault>
    <pte_alloc = 20/>
    <do_no_page>
      <filemap_nopage>
        <__find_get_page = 15/>
        <mark_page_accessed>
          <activate_page = 30/>
        </mark_page_accessed = 17>
      </filemap_nopage = 77>
    </do_no_page = 52>
  </handle_mm_fault = 42>
</do_page_fault = 79>
<#ret_from_sys_call#/>
</!Exception! type="Page-Fault Exception">
```

図４ 関数遷移の表示例(ページフォールト)

```
0807f575:  83 ec 0c        sub     $0xc,%esp
0807f578:  ff 75 08        pushl   0x8(%ebp)
0807f57b:  e8 18 fa ff ff  call    0x807ef98

[!Exception!  type="Page-Fault  Exception"  reason=
"NP/R/U" addr="0x807ef98"]
<page_fault>
c0107020:  68 50 f9 10 c0  push   $0xc010f950
c0107025:  e9 66 fe ff ff  jmp    0xc0106e90
c0106e90:  1e              push   %ds
c0106e91:  50              push   %eax
```

図５ 逆アセンブル表示例

れる．一つ目はトレースデータ解析部で,トレースデータにレコード単位でアクセスするためのインタフェースを提供する．二つ目はシンボルテーブル部で,シンボルファイルからシンボルを検索するためのインタフェースを提供する．また，シンボルファイル作成用にカーネルと LKM および対象プログラムとその共有ライブラリのシンボル取得を自動化するツールも作成した．

4. 使用例

図４と図５は Web サーバの Apache をトレースしたときのページフォールト処理部分である．図 4 の関数遷移では XML 風の表示にして見やすさを工夫した．シンボルの後ろの数字はダイナミックステップ数である．図 5 では，ユーザプログラム実行中に，ページフォールトが起こり，その後制御がカーネル内のページフォールトハンドラに移行しているのが分かる．

5. おわりに

本論文では，Linux システムを解析するための命令トレーサ:TURUNG ならびに解析ツール群: TURUNG Tools について述べた．本機構を用いることで AP と OS の統一的な性能評価が可能となった．

参考文献

森本 洋行,小宮山 彰一郎,毛利 公一,吉澤 康文,“性能評価のための命令トレーサの開発,”電子情報通信学会論文誌(D-I)，Vol．J84-D-I，No．6，pp．584-593，2001.