性能データの可視化分析ツール

小野　美由紀　　山本　昌生　　平井　聡　　久門　耕一
（株）富士通研究所

1　はじめに

コンピュータにおける性能問題の解決を目的として、動作プログラムのプロファイルデータを時系列に分析・可視化するツールを開発した。本ツールでは、統計情報の提示だけでなく、時系列に可視化することによって、瞬間的に発生する障害原因の追及を可能とする。本発表では、ツールの特徴と分析例について説明する。

2　背景

コンピュータで実行されるプログラムの性能問題を解決するために、実行時間のかかる関数を特定することが必要である。これには時間ベースサンプリングによるプロファイリングが有効である。

従来のプロファイラ(OProfile[1]など)には測定時間全体に対するプロセスや関数の実行割合をまとめた統計情報をテキストベースで出力するものが多い。しかし、このような統計情報だけでは、瞬間的に発生する性能劣化の原因などを特定することができない。

また、1台のマシンに多数のCPUが搭載される場合、1CPUのみで動作するプログラムの動作割合は計算機全体から見ると低い。このようなプログラムは一見問題なさそうに見えるが、他のCPUの動作を妨げる場合もある。Linuxの性能分析ツール（timechart[2]）ではCPUやプロセスの動きを時系列に分析できるが、関数レベルの分析や対話的な分析機能が不足している。

そこで、我々は、プロセスや関数の実行割合の時間による変化をマシン全体およびCPU単位で可視化する機能を備えた分析ツールを開発した。

3　ツールの特徴

本ツールは、プロファイラで採取されたサンプリングデータをGUIにより分析するものである。サンプリングデータには各採取時点のCPU、動作していたプロセスや関数などを特定する情報が含まれる。

本ツールでは、段階的に問題のあるCPUや時間帯を絞り込んで分析していくことを想定する。分析の手順を以下に示す。

1）測定時間全体におけるプロセスや関数の実行割合を確認する。
2）1）で性能問題の原因が明らかにならない場合は、マシン全体あるいは各CPUに対して実行割合の時間による変化を調査し、問題と思われるCPUや時間帯を絞り込む。
3）2）で絞り込んだものに対して詳細を調査する。

分析時には分析対象時間帯などをユーザ指定可能とすることにより、対話的な分析を可能にする。以下では2）と3）で使用する機能について述べる。

3.1　時系列グラフ

ある時間帯に突出して動作したプロセスや関数を把握するため、測定時間をいくつかに分割して各時間帯におけるプロセスや関数の動作割合を集計することにした。この機能により作成したグラフを時系列グラフと呼ぶ。

集計した各時間の値をグラフ化することにより、プログラムの挙動の変化の把握や瞬間的に発生する障害の発見を可能にする。

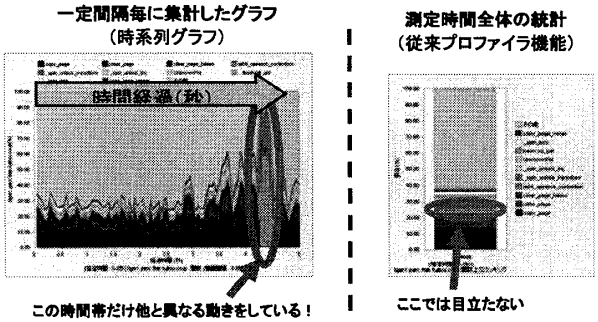

図1　時系列グラフ

Visualization and Analysis Tool of Performance Data
Miyuki Ono, Masao Yamamoto, Akira Hirai, Kouichi Kumon
Fujitsu Laboratories, Ltd.

図 1 は測定時間全体を対象として上位 10 関数を集計したものである。左が時系列グラフであり、右は従来のプロファイラ機能に相当するグラフである。左のグラフは、横軸が経過時間、縦軸が関数の実行割合であり、各時間帯における関数の動作割合を面グラフで表している。このグラフでは、ある時間帯（丸で囲んだ部分）にのみ突出して動いた関数群を一目で見つけることができる。一方、右のグラフは測定時間全体での実行割合を棒グラフにしたものである。このグラフから一時的に動作した関数群を把握することはできない。

3.2 詳細時系列グラフ

問題と思われる時間帯でのプロセスや関数レベルでの具体的なプログラム挙動を把握するため、時系列に採取しておいたサンプリングデータを集計処理せずに、採取した時間順にそのまま表示する機能を作成した。この機能により作成したグラフを詳細時系列グラフと呼ぶ。

表示する時間帯が長い場合にはすべてのデータを表示することができないため、表示領域の大きさに応じて間引き表示を行う。

詳細時系列グラフの例を図 2 に示す。グラフは横軸が経過時間、縦軸が CPU 番号である。この例では、ある時間帯に 3 つの CPU で動作したプロセスを表示している。本来は各 CPU でプロセスが動作するはずであるが、問題発生時刻（図中の縦線部分）には、一番下に表示されている CPU3 しか動作していなかったことがわかる。

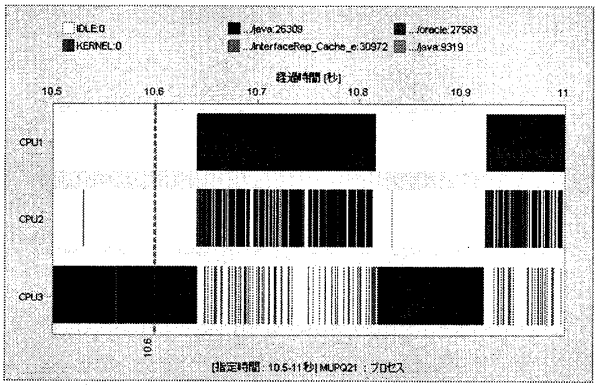

図 2　詳細時系列グラフ

4 分析例

ここでは、複数 CPU 環境において Java を利用したアプリケーションで多重度を上げても CPU の負荷が上がらず、性能が向上しない問題が発生したケースを例として説明する。このような場合には CPU 単位での分析が必要となる。図 3 は問題発生時に採取したデータをもとに作成した時系列グラフと詳細時系列グラフである。2 つのグラフは同一 CPU 群の同一時間帯を対象としている。時系列グラフは上位 10 プロセスの動作割合の変化を表している。本来は各 CPU でプロセスが動作するはずである。しかし、丸で囲んだ時間帯では一番下に表示されている CPU3 のみで Java プロセスが動作していたことがわかった。Java プロセスで実行された関数の内訳を詳細時系列グラフにしたところ、CPU 3 では GC 関連の関数が動作しており、このために他の CPU で動作しなかったことがわかった。

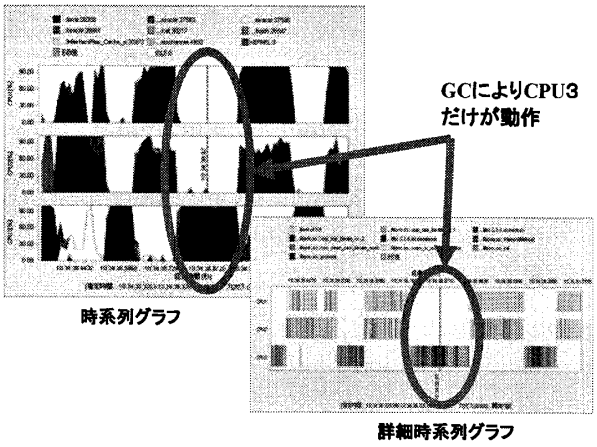

図 3　分析例

5 おわりに

本ツールでは、プロファイルデータを時系列に見せることによって、測定時間全帯の統計情報を見せるだけでは解決できなかった、瞬間的に発生する性能問題の分析や CPU 単位での動作の分析を可能とした。コンピュータに搭載される CPU 数は今後ますます増加することが予想される。今後は、多数 CPU の効率的な分析を支援する機能を検討していきたい。

最後に、本ツールを共同開発させていただき、適切な助言をいただいた富士通株式会社田代殿に深く感謝いたします。

参考文献

[1] http://oprofile.sourceforge.net/about/
[2] http://blog.fenrus.org/?p=5