B-013

Tender における OS 動作の把握を目的とした可視化情報の解析部の実現と評価

Implementation and Evaluation of Analyzer of Visualization Information for Understanding Operating System Behaviors in *Tender*

川江純平† 田端利宏† 谷口秀夫†
Junpei Kawae Toshihiro Tabata Hideo Taniguchi

1．はじめに

OS は処理流れが複雑なソフトウェアであり，処理流れを把握するのは難しい．このため，不具合発生時，不具合箇所の特定は難しい．これまでに，OS 動作状態の可視化の研究[1][2]は数多く行われているが，OS の処理流れ，特に割込や例外の実行を可視化する研究は少ない．

本稿では，***Tender*** オペレーティングシステム[3]において割込を考慮した OS 動作の可視化情報を基に，OS の処理流れを把握する可視化情報の解析部の実現方式と適用事例について述べる．適用事例により，OS 動作の解析と不具合箇所の特定が可能であることを示す．

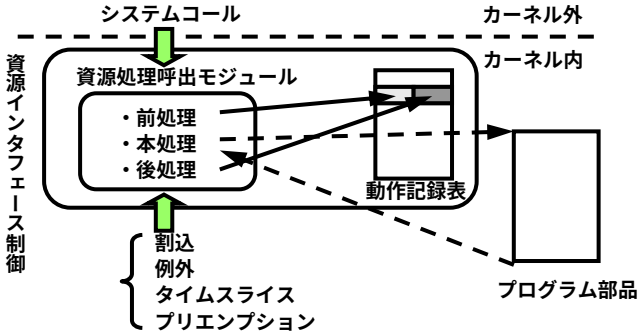

図 1 情報収集機構の概要

2．*Tender* の可視化機能[3][4]

2.1 *Tender* の特徴

Tender では，操作する対象を資源として分離し，独立化している．また，各資源は，資源識別子と資源名で識別される．

Tender 特有の部分として,資源インタフェース制御(RIC : Resource Interface Controller)がある．RIC は，各資源を操作するプログラム部品の呼出を制御している．これにより，プログラム部品の更新，追加，削除を容易にし，OS 動作の可視化を可能にしている．

2.2 基本構造

可視化機能の処理は，以下の 4 つの部分で構成される．

(1) 可視化情報の収集(収集部)
(2) 可視化情報の受け渡し(転送部)
(3) 可視化情報を基にした OS 動作の解析(解析部)
(4) 解析データの可視化表示(表示部)

収集部は，情報収集の対象となる処理を常に監視する必要があるため，OS 内部に実現する．解析部と表示部は，データ処理や画面表示の処理を行う．これらの処理部は，OS 内部に実現すると，OS 内部の負荷が増加する要因となるため，OS 外部に実現する．OS 内部の収集部と OS 外部の解析部の間で可視化情報を受け渡すため，転送部を設ける．

2.3 収集する可視化情報

可視化情報として，以下に示す 6 つの項目を資源処理の呼出ごとに記録する．

(1) 処理の要求先を示す情報(処理の対象となった資源の資源識別子)
(2) 処理内容に関する情報(OPEN，CLOSE，READ，WRITE，CONTROL)
(3) 処理結果情報
(4) 処理開始契機情報
(5) 処理開始時刻
(6) 処理終了時刻

これらの情報を記録するために，動作記録表と呼ぶデータ構造を用意する．動作記録表は，上記の項目を 1 エントリとする配列形式のデータ構造である．

†岡山大学 大学院自然科学研究科

2.4 情報収集機構

情報収集機構を図 1 に示す．RIC 内の資源処理呼出モジュールが各プログラム部品の呼出前と呼出後に，必要な情報を記録する．こうすることで，全ての資源処理に対して情報の収集を行うことができる．ここで，プログラムポインタ表を参照し，要求されたプログラム部品を呼び出す処理を本処理，本処理の前後に行う記録処理をそれぞれ前処理と後処理と呼ぶ．

3．解析部の実現

3.1 要求と対処

解析部の目的は，収集部で取得した情報を解析し，各資源操作の性能や呼出関係を利用者に分かりやすく伝え，OS 動作の不具合箇所調査の支援，および性能改善の効率を向上させることである．

解析部では，OS 処理に大きく影響を与える資源操作を明らかにできることが要求される．このため，資源操作の呼出回数や，処理時間を示す．呼出回数が多く，かつ処理時間が大きい資源操作は OS 処理に与える影響が大きい．また，不具合箇所調査のため，動作の安定性の程度を明らかにできることが要求される．このため，処理時間の分散を算出し処理時間のバラつきを示す．

また,可視化情報の量は非常に多い．そこで,利用者の情報分析の負担を減らすために，不具合箇所，およびボトルネックの可能性が高いと推測される資源操作の抽出を容易にできることが要求される．このため，各項目(呼出回数，処理時間，分散)の値の大きいものを抽出できるようにする．

3.2 処理の流れ

収集部で取得した可視化情報は，転送部により OS 外部へ転送され，解析部へと渡される．図 2 で示すように，解析部では，可視化情報を流れ識別子，資源の種類，操作の種類ごとに分類する．これにより,各資源操作をより細かく分析することができる．この分類にしたがって， 各資源操作の呼出回数と処理時間を集計する．この結果 を利

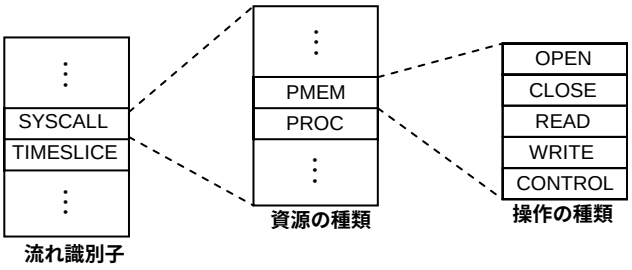

図 2　解析部による資源操作の分類

用して，各資源操作の処理時間と分散を算出し，一覧表を出力する．これにより，全処理中で出現回数が多い資源操作，処理時間が大きい資源操作，および処理時間が不安定な資源操作を把握することができる．

また，各資源操作の呼出回数，処理時間，および分散の中で値の大きいものを 10 項目抽出し，降順にソートし表形式で出力する．この出力例を図 3 に示す．

4．事例による評価

4.1 事例

解析部を用いて ***Tender*** の性能改善と不具合箇所の発見を試みた．具体的には，***Tender*** 上に Apache Web サーバを動作させ，その被アクセス時における可視化情報を解析した．

4.2 分析

各資源操作を処理時間で降順に並べたものを図 3 に示す．これより，最も処理時間の大きい資源操作は，仮想領域の生成処理である．また，仮想領域の生成処理は，処理時間の分散が全資源操作中 4 番目である．

そこで，可視化情報を用いて，仮想領域の生成処理の内部動作の調査を行った．仮想領域の生成処理内では，複数の実メモリ生成処理が行われている．***Tender*** では，実メモリは原則 1 ページずつ資源として確保するよう設計されているため，要求したページ数と同回数の実メモリ生成処理が行われる．ここで，実メモリ生成の前処理から後処理までに要する時間は，高々1[μs]程度の処理時間である．しかし，仮想領域生成処理中で実メモリ生成処理が 1 回呼ばれるたびに，50～100[μs]ほど処理時間が増加していた．このことから，実メモリ生成の前処理以前，後処理以降の箇所で大きな処理時間がかかっていると考えられる．各資源操作には資源名が割り当てられるため，資源生成処理では，本処理を呼出す前処理の前に，RIC が生成する資源の資源名と既に登録されている資源名とで資源名の重複がないか検査を行う．調査の結果，この検査処理に大きな時間を要していることがわかった．実メモリは ***Tender*** の起動処理中に数百個生成されており，大半は起動後も解放されずに残っている．これらの起動時に生成された資源と，生成しようとする資源との資源名重複検査を行うため，実メモリ生成処理前の処理時間は大きくなると考えられる．

	flowid	rsc	ope	num	avetime	totaltime	variance
1	SYSCALL	:VREGION	:OPEN	[65]	[1434]	[93212]	[762578]
2	INTR	:EXEC	:CONTROL	[39]	[1410]	[55009]	[6145747]
3	SYSCALL	:PROC	:WRITE	[12]	[4136]	[49639]	[1596882]
4	SYSCALL	:PROC	:READ	[21]	[1204]	[25291]	[1271088]
5	SYSCALL	:VMEMORY	:CONTROL	[352]	[38]	[13668]	[1887]
6	SYSCALL	:VUM	:OPEN	[168]	[66]	[11209]	[3028]
7	SYSCALL	:VREGION	:CLOSE	[62]	[90]	[5606]	[726]
8	SYSCALL	:VREGION	:CONTROL	[3997]	[0]	[3663]	[104]
9	SYSCALL	:PLATE	:CONTROL	[6]	[459]	[2756]	[8150]
10	UNKNOWN	:VMEMORY	:CONTROL	[12]	[196]	[2357]	[26717]

図 3　解析部の出力例(時間の単位は(μs))

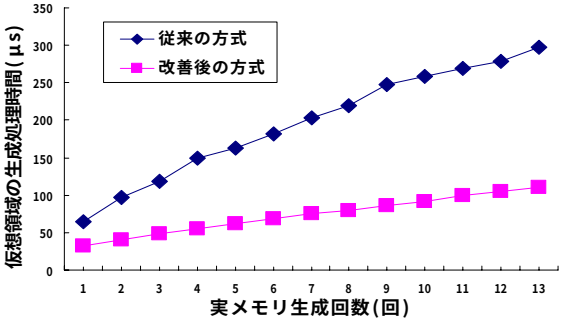

図 4　実メモリ生成回数と仮想領域の生成時間の関係

4.3 性能改善

実メモリの生成は，カーネル内で行われる．そこで，資源名を資源識別子と生成ページ番号により決定し，カーネル内で資源名が重複しないことを保障した．これにより，実メモリ生成処理に限り，資源名の重複検査を行わないように RIC を変更した．1 回の仮想領域の生成中に実メモリの確保を行う回数と処理時間の関係を示すグラフを図 4 に示す．図 4 より，仮想領域の生成処理において，50%以上の処理時間短縮に成功した．

5．おわりに

OS 動作の可視化のための割込を考慮した情報収集機構で収集した情報を基に，OS 動作を解析し，性能把握やボトルネック箇所特定を支援する機能を提案し，実現した．また，実現した解析部を用い，OS 動作の解析例により，ボトルネック箇所を特定できることを示した．また，改善により，性能向上を確定した．

残された課題として，転送部を含めた可視化機能全体の構築，資源呼出を基本とした詳細な動作の可視化方法の検討がある．

謝辞 本研究の一部は，科学研究費補助金 若手研究(B)(課題番号：18700030)による．

参考文献

[1] B. M. Cantrill, M. W. Shapiro, A. H. Leventhal：Dynamic Instrumentation of Production Systems, USENIX 2004 Annual Technical Conference, pp.15-28, 2004.

[2] V. Prasad, W. Cohen, F. Ch. Eigler, M. Hunt, J. Keniston, B. Chen：Locating System Problems Using Dynamic Instrumentation, In Proceedings of the Linux Symposium, Vol.2, pp.49-64, 2005.

[3]谷口秀夫，青木義則，後藤真孝，村上大介，田端利宏：資源の独立化機構による ***Tender*** オペレーティングシステム，情報処理学会論文誌，Vol.41，No.12，pp.41-48，2000.

[4]木下彰，河原太介，田端利宏，谷口秀夫：***Tender*** における OS 動作の可視化のための情報収集と表示の方式，情報処理学会研究報告，2007-OS-105，Vol.2007，No.36，pp.31-38，2007.