C-027

長時間トレース技術を用いた組込みソフトウェア開発の効率向上に関する検討

A Study of Effective Embedded Software Development Using Large Data Tracing

長野 岳彦† 今井 光洋†
Takehiko Nagano Mitsuhiro Imai

1. はじめに

組込み機器メーカのシェア競争の激化から、機器の開発期間の短縮や低コスト化に対する要求が強くなっている[1]。そこで開発コスト削減を目的として、ソフトウェアのバグに代表される障害の対策期間削減による開発効率の向上が望まれている。

ソフトウェアの障害には、再現性の高い物と低い物がある。障害の再現性が高い場合、原因調査に直ぐ着手出来、早期解決が期待出来る。しかし、障害の再現性が低い場合、再現要因の特定に時間がかかり、対策工数は増大する。そこで本研究は、再現性の低い障害の対策効率向上を目的とする。

従来の再現性の低い障害の対策では、再現条件特定、解析情報取得に試行錯誤を繰り返し、解析工数増加の要因となっていた。本研究では、障害解析に必要十分な情報を取得出来る長時間トレースを採用し、従来試行錯誤していた工程を 1 回の情報取得にする。また、長時間トレースにより、解析対象の情報が大幅に増加し、解析効率が低下する。その問題を解決するため、データマイニングを利用して解析対象のデータ量を削減する手法を提案する。本稿では、各手法の提案・評価を報告する。

2. 提案手法

本章では、長時間トレースシステムと、データマイニングを用いた問題点の絞込み手法について提案、説明する。

2.1 長時間トレースシステム

組込みシステムは、システム毎にハード構成、周辺デバイスが異なり、ハードウェアを直接制御するプログラムも多い[1]。そこでアプリやデバイスドライバ、OS を俯瞰出来、問題点を切り分け可能なシステムワイドな解析をする必要がある。OS はハードウェアの抽象化、システム全体の制御をしており、その挙動情報が、システムワイドな解析に役立つ。OS の挙動情報を用いて時系列解析をするツールとして、LKST[2]や LTTng[3]等があり、組込み分野での適用も進んでいる。しかし、組込みシステムはメモリ等のリソースが少ないため、OS の挙動情報等を大量に保存出来ない。そこで、外部バス経由で記憶容量が大きい外部記録装置に OS の挙動情報を記録する方式を提案する（図 1）。OS の挙動情報の収集には LKST（Linux Kernel State Tracer）を使用する。従来の LKST では、長時間のトレースをするには、メモリ上に複数面のバッファを確保し、あるバッファ 1 面を使い切ると、バッファを切り替え、元のバッファの記録結果をファイルに出力するが、先に述べたとおり、組込みシステムに搭載されるメモリは容量に制限がある。また、出力先のストレージデバイスも無い場合が多い。そこで本システムでは、外部バスに LKST のトレース結果を送出するための出力モジュールを追加し、外部記録装置のインタフェースの有無を問わず、挙動情報の出力を可能にした。外部バスには、HDD を保有する長時間トレースハードウェアのプローブを接続し、バスにイベントの出力が流れる度、長時間トレースハードウェアに記録する仕組みになっている。

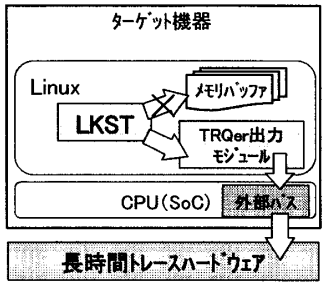

図1 長時間トレースシステム

2.2 大規模トレース結果の解析

本研究では、トレースデータ全体から、解析対象の範囲をデータマイニングで絞り込む(図２)。具体的には、Anomaly Detection[4]のアプローチを用いる。解析対象のソフトウェアの正常動作を元に、クラスタリングを用いて正常状態を学習、解析時には、学習結果から、単位時間単位で正常クラスタのセントロイド Ci と入力データ X の最小値 minD(X,Ci)を求め、その値を異常度として算出し、異常度の大きい箇所周辺を解析対象として絞り込み、解析効率を向上する。

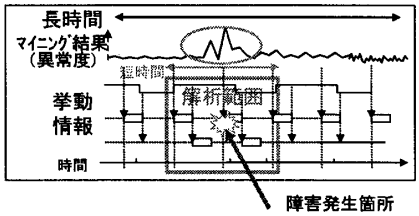

図２データマイニングを用いた解析イメージ

LKST のトレースデータをクラスタリングに活用するには、テキストデータをベクトルデータに変更する必要がある。本研究では、単位時間毎に LKST のイベント id 毎に発生回数を数え上げる手法を用いてテキストデータをベクトルデータに変更した。

3. 性能測定

提案する長時間トレースシステムの実用性を評価するため、Apache Bench を用いて、外部から評価対象に対し負荷をかけ、性能を測定した。評価項目は CPU の使用率と Apache Bench によるテスト実行時間を用いた。

3.1 測定環境

組込み機器（Atmark Techo 社 Armadillo-300）と負荷を外部から与える PC（OS:Linux）をローカルネットワークに接続し、組込み機器上で、web サーバ（thttp）を動作させ、PC から Apache Bench を使用し、組込み機器上の web サー

†株式会社 日立製作所 中央研究所 組込みシステム基盤研究所
Hitachi, Ltd., Central Research Laboratory Embedded System Platform Research Laboratory

バに大量のアクセスを発行、負荷を与える。使用したコマンドラインは ab –n 30000 –c5 アクセス先 URL である。CPU の使用率は、Snap Gear 社の Greg Ungerer 氏が OSS で公開している cpu.c を用いた。cpu.c は/proc/stat 以下にある CPU の動作状況から、システム時間、ユーザ時間等を算出するツールである。上記を使い、LKST 無し、LKST のトレース結果をメモリ上に記録、同トレース結果をファイルに記録（従来の長時間記録手法）、提案手法である長時間トレースシステムに記録の 4 例で比較した。このうち、トレース結果を定期的にファイル出力する方式は、Linux サーバ等で長時間記録する際に使用される手法である。LKST のマスクセット[2]は全て ON になっている

3.2 比較結果

結果を表 1 に示す。Apache Bench による負荷が高いため全体的に CPU 使用率は高いが、従来手法である LKST のトレース結果をファイルに記録する手法の CPU 負荷が最も高く、3.02%上昇している。それに対し、長時間トレースを用いる場合 0.23%の上昇と、それ程 CPU 負荷に影響が出てない。しかし、CPU 使用率のうち、OS が CPU を使用した結果であるシステムの項目を見ると、長時間トレースが、LKST 無しの場合に比べ 20.14%上昇している。理由は外部バスにトレース結果を出力するカーネルモジュールが頻繁に動作しているためである。従来手法のシステムの項目が低い理由は、メモリからファイルに結果を出力する際、ユーザランドのデーモンプログラムが頻繁に動作しているためである。

表 1　CPU 負荷測定結果

	LKST 無	LKST メモリ記録	LKST ファイル記録	LKST 長時間トレース
CPU 使用率	92.00%	92.45%	95.02%	92.23%
内 system	36.80%	42.32%	34.47%	56.94%

次に、Apache Bench の測定結果を表 2 に示す。LKST が動作しない場合に対し、ファイル記録は 156.62 秒の遅延、長時間トレースは 128.11 秒の遅延が発生している。双方とも、LKST 無しよりは性能が悪化しているが、ファイル記録（従来法）に比べ、提案手法は 17.88%性能が改善している。

表 2　Apache Bench 測定結果

	LKST 無	LKST メモリ記録	LKST ファイル記録	長時間トレース
所要時間(秒)	159.49	173.16 (+8.57%)	316.11 (+98.20%)	287.60 (+80.32%)

4. 障害発生箇所の特定と解析範囲絞込み評価

本章では、データマイニングを用いた問題点の絞込み結果の有効性を、異常度の変化と変化時刻から検証する。

本研究では、再現性の低いバグの解析を対象としている。そこで、(1)複数プログラムが並列動作することに起因するバグ、(2)ハードウェア等の外部要因に起因するバグ、を想定し、以下の 2 種類の障害発生時刻の特定を評価した。

4.1 デッドロックの検出（並列動作起因）

本節では、複数プロセスが関連し発生する障害の発生が特定可能か評価を実施した。Armadillo-300 上で同一の資源を利用するプロセスを 2 種類起動し、500 秒経過後に片側のプロセスが資源を解放せず、意図的にデッドロック問題を発生させた。表 3 に結果を示す。

表 3　デッドロック発生時の異常度変化

	通常時	デッドロック時
異常度	1	4.00

表 3 の結果からわかる通り、デッドロック発生時には異常度が 4 倍強上昇した。また、異常発生時刻と異常度の変化は同期しており、異常発生箇所を特定出来た。

4.2 ネットワーク障害の切り分け（外部要因）

本節では、開発対象のソフトウェアに対し、外部要因に起因して発生する障害の発生箇所を特定可能か、評価を実施した。

評価手法は負荷測定時と同様の機器構成で、Armadillo-300 上で動作する web サーバ（thttp）に対し、apache-bench を使用して 30 秒毎に外部から大量アクセスを発生させ、その際の平均の異常度の変化を測定した。表 4 に結果を示す。

表 4　ネットワーク攻撃時の異常度変化

	通常時	攻撃時
異常度	1	598.32

表 4 の結果からわかるとおり、外部から攻撃を受けている際は、異常度が平均で 598 倍上昇し、異常な箇所が推定出来た。また、4.1 同様、異常発生時刻と異常度の変化は同期しており、異常発生箇所を特定出来た。

5. まとめと課題

本研究では、長時間トレースの一手法として LKST の外部記録装置への記録手法を提案した。これにより、従来の LKST の長時間記録方法に比べ、17.88%の性能改善が見込める結果を得た。更に、長時間トレース結果に対しデータマイニングを適用し、再現性の低い障害の障害発生箇所を特定する手法を示した。これにより、障害再現期間、問題特定期間の短縮が出来、開発効率向上が期待出来る。

今後の課題は、より複雑な障害に対し、データマイニングによる障害発生箇所特定が可能か検証することである。

参考文献

[1]高田広章“組込みシステム開発技術の現状と展望”,情報処理学会論文誌,Vol.42,No.4(2001)

[2]畑崎恵介,中村哲人,芹沢一,“稼動中システムのデバッグを考慮した OS デバッグ機能”情報処理学会研究報告.[システムソフトウェアとオペレーティングシステム]pp.33-39.2003

[3]Mathieu Desnoyers et al , “The LTTng tracer; A low impact performance and behavior monitor for GNU/Linux”,OLS2006 Proceedings (2006)

[4]Christina Warrender et al, “Detecting Intrusions Using System Calls” ,Alternative Data Models,IEEE Symposium on Security and Privacy,pp.133-145,(1999)

[5]Roy A.Maxion,Kymie M.C. Tan, “Anomaly Detection in Embedded Systems” ,IEEE Transactions on computers,Vol51,No.2（2002）.