1A-2

メモリバストレースによるメモリ復元手法の検討

望月　泰行　　五十嵐　史生　　黒澤　寿好
三菱電機（株）情報技術総合研究所

1. はじめに

携帯電話などの高機能な組込み機器のソフトウェア開発では、ソフトウェアが大規模化、かつ複雑化している。これにともなって、ソフトウェアの障害（バグ）も複雑化し、バグの解析負荷の増大が製品開発コストを増加させる傾向にある。

組込みシステムの開発におけるバグの解析には、ICE(In-Circuit Emulator)が利用される。ICE には一般的なデバッガの機能に加え、CPU のメモリアクセスをトレースする機能がある。メモリアクセスのトレースから任意時点のメモリイメージを復元することができれば、メモリ破壊／波及型バグ等の解析が大幅に効率化される。

本稿では、メモリアクセスのトレースからメモリを復元するにあたり、障害発生の最終時点のメモリイメージから時間を遡りながらメモリを復元する手法について検討する。また、連続する講演では、メモリ復元手法の評価について発表する。

2. メモリアクセスのトレース機能

今回、我々が取り扱っているプロセッサは、携帯電話などに使われることが多い ARM プロセッサである。ARM プロセッサには ETM と呼ばれる回路がオプションで組込まれ、ARM コアによるメモリアクセスのトレースが可能となっている。ETM の機能は ARM コアの本来の動作に対して副作用なく、ARM コアと 1 次キャッシュとの間のデータのやりとりをトレースして ETM 専用の端子に出力する。（図 1）

なお、メモリアクセスのトレースデータは、読出し／書込みの別、アクセス先のアドレス、および読み書きしたデータ値からなる。

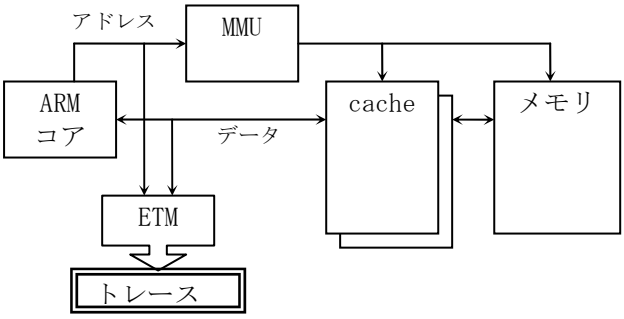

図 1 ARM11 の構成

Study of memory restoration from memory bus trace
Yasuyuki MOCHIZUKI, Fumio IGARASHI, Hisayoshi, KUROSAWA
Information Technology R&D Center, Mitsubishi Electric Corporation

3. メモリ破壊／波及型バグ

開発現場で発生するバグの大半は、以下に示すようなツールを使って原因を解析している。

- PC 上のエミュレータ
- メモリのコアダンプ
- トレース／ログ
- ICE(In-Circuit Emulator)

一方で、これら既存のツールでは容易には原因を解析できないバグがあり、発生件数は少ないものの、製品の納期や開発コストに重大な影響を及ぼすことがある。解析難度の高いバグのうち、メモリの復元機能によって解析が効率化されるバグの代表がメモリ破壊／波及型のバグである。

メモリ破壊型バグとは、あるプログラム A が利用している特定のメモリ領域のデータを、別のプログラム B が不正に破壊するバグである。破壊された領域のデータを読出すプログラム A が誤動作することでバグの存在が発覚する。

メモリ破壊型バグは、データを破壊したプログラム（原因）と誤動作したプログラム（結果）の間になんらかの関連性があれば、類推が可能となるため、原因の特定が比較的容易である。

一方、原因と結果の間に関連性がない場合には、類推が困難になることに加え、誤動作の症状と発生するタイミングがランダムになる傾向が強くなる。これにより、既存ツール利用の効果が薄くなり、その分、解析作業の工数が多くなる。

さらに、メモリが破壊されたことによる誤動作が別のメモリ破壊を引き起こす波及型の場合、根本原因の特定は困難を極める。

4. メモリ破壊／波及型バグの解析

メモリ破壊／波及型バグを解析する手順について、図 2 を使って説明する。

4.1. 各タスクの動作

バグが発覚するまでの、各タスクの動作は次のとおりである。

(1) T2 時点で、タスク A の障害原因により、タスク B のメモリ領域をタスク A が破壊する。
(2) T1 時点で、タスク A によるメモリ破壊が原因でタスク B が誤動作し、それにより、タスク C のメモリ領域の破壊に波及する。（この時点ではタスク B とタスク C は継続して動作）
(3) T0 時点で、タスク C に波及したメモリ破壊が原因でタスク C が誤動作し、システム停止などの致命的な症状によってバグが発覚する。

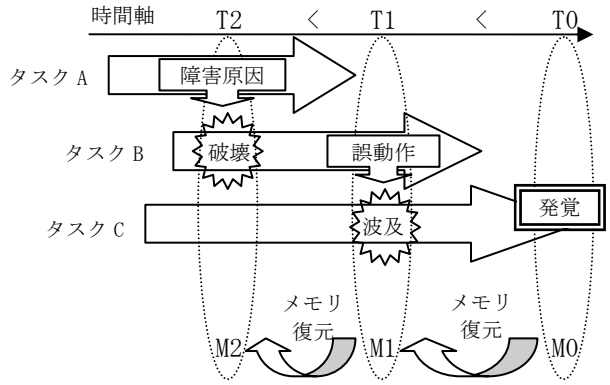

図 2 メモリ破壊／波及型バグの解析

4.2. メモリ復元を利用した原因の解析

メモリアクセスのトレースデータを使ってバグの原因を解析する手順は次のとおりである。

(1) メモリイメージ M0 のスタックなどを既存ツールで分析し、致命的症状の発生原因（タスク C のデータ破壊箇所）を特定する。
(2) トレースデータから、データ破壊が「波及」して発生した時点 T1 を特定する。これは、破壊された領域への書込みが発生した最後の時点である。
(3) T1 時点のメモリイメージ M1 を復元する。
(4) メモリイメージ M1 を既存ツールで分析し、タスク B の誤動作の原因（タスク B のデータ破壊箇所）を特定する。
(5) 同様に、タスク B とタスク A に対する分析を行い、タスク A の障害原因を特定する。

4.3. メモリの復元アルゴリズム

T1 時点のメモリイメージ M1 を復元する手順を例に、メモリを復元するアルゴリズムを説明する。なお、以下では、読出しのトレースデータを"r(a,d)"、書込みを"w(a,d)"とする。（a はアドレス、d はデータ）

メモリを復元する際には次の２つのワーク領域を利用する。

- 配列 M：復元メモリイメージ（アドレス a のデータ値が M[a]）
- 配列 U：不定値フラグ（U[a]の値が 1 の場合はアドレス a のデータ値が不定）

まず、配列 M を M0 で初期化し、配列 U の各要素を 0 で初期化する。

次に、トレースデータを T0 時点から T1 時点まで逆順にたどりながら以下の処理を繰り返す。

(1) トレースデータが r(a,d)なら、M[a]に d を格納し、U[a]に 0 を格納する。
(2) トレースデータが w(a,d)なら、U[a]に 1 を格納する。（書込み w(a,d)に対して、書込み後のデータ値は d に確定するが、書込み前のデータ値はアドレス a に対する直前のトレースが現れるまで確定しない。この意味で、不定値フラグ U[a]を 1 とする。すなわち、U[a]が 1 のとき、M[a]の値は無効である。）

4.4. 不定箇所の復元

以上の手順では、メモリが復元されない不定箇所（アドレス a に対して U[a]が 1 となっている箇所）が残る。不定箇所のデータ値 M[a]を求めるためには、トレースデータを T1 時点からさらに逆順にたどり、アドレス a に対するメモリアクセス（r(a,d)または w(a,d)）が現れたら、このときの d がメモリ破壊時点のデータ値 M[a]である。

5. その他の検討事項

5.1. DMA 転送の影響

DMA 転送は CPU を介さずにメモリを読み書きする。DMA と CPU は非同期に動作するため、DMA 動作中のメモリをトレースデータから完全に復元することは不可能である。DMA の影響について、２つのケースに分けて考察する。

まず、DMA ドライバにバグがある場合や、他の影響によって DMA ドライバが誤動作する場合は、メモリ復元の結果を信頼することができないためバグ解析に利用することができない。

次に、DMA ドライバが正常に動作することを仮定すると、トレースデータ中の命令フェッチ列から DMA の動作タイミングを絞り込むことが可能となるため、実際のバグ解析では多くのケースでメモリ復元結果が有用であると考えられる。

5.2. CPU レジスタの復元

メモリの復元に加え、CPU レジスタの復元が実現すると誤動作発生のメカニズムの解明に役立つ。

CPU レジスタは、トレースデータ中の命令フェッチを分析することによって復元できる可能性がある。ただし、CPU レジスタ復元のアルゴリズムは CPU の命令セットごとに個別に検討する必要がある。

6. おわりに

組込みソフトウェア開発のデバッグに利用される ICE には CPU のメモリアクセスをトレースする機能があり、これを利用したメモリ復元機能はメモリ破壊／波及型のバグの原因特定に有用なことを示した。

今後は、DMA 転送の影響について検証等を実施し、メモリ復元機能の有用性を確認する。また、デバッグ作業への適用に向け、CPU レジスタの復元機能や割込みの再現機能を実現する。

参考文献

[1]アーム(株), ARM アーキテクチャリファレンスマニュアル, http://www.jp.arm.com/
[2]アーム(株), ETM9 テクニカルリファレンスマニュアル, http://www.jp.arm.com/