2H-09

高信頼な組込みシステムに向けた障害情報収集機能を持つ監視用OSの試作

山本　遼介† 片山　吉章‡ 明田　修平† 瀧本　栄二† 毛利　公一†

† 立命館大学　　‡ 三菱電機株式会社 情報技術総合研究所

1 はじめに

近年，カーナビ，POSレジ，スマートフォンなど多くの組込み機器は，ハードウェアの高性能化，アプリケーションの多様化，インターネットへの接続などの理由から汎用OSを利用している．しかし，汎用OSは組込み向けOSと比較し，ソフトウェアとして規模が大きく，動作が複雑である．それゆえに，OS自身にバグが潜在し，障害発生の原因となる．さらに，汎用OSは障害発生時の再現が難しく，障害の原因を特定することが困難である．

このような障害の原因を特定するために，障害発生時にメモリダンプを収集する手法を用いる．しかし，既存のツール[1]では，障害情報の収集後にOSを再起動させるため，サービスの再開に時間を要する．一方で，障害情報を収集する前にOSを再起動させた場合，再起動処理によりメモリの中身が書き換えられるため，障害情報を収集できない．

以上の課題を解決するために，障害情報の収集とシステムの再起動を同時に行う手法を提案する．これにより，メモリダンプを収集しつつ迅速なシステムの再起動が可能になる．提案手法では，汎用OSの外部に監視用OSを動作させて障害情報を収集する．本稿では，メモリダンプを収集する監視用OSを実装し動作確認を行ったので，それらについて報告する．

2 提案手法

2.1 概要

障害情報の収集を行う場合，収集完了まで汎用OSを再起動できないためサービス再開が遅れる．この課題を解決するために障害情報の収集と汎用OSの再起動を同時に行う．

ただし，汎用OSに障害が発生した場合，汎用OSの動作は保証されない．そのため，確実に障害情報を収集するために，汎用OSの外部に障害情報を収集する監視用OSを別途動作させる．

複数のOSを動作させる手法としてLPAR[2]を採用する．LPARは，メモリやCPUなどのハードウェア資源を各OSに分割し複数のOSを動作させる．そのため，仮想化と比較して低オーバヘッドで複数のOSを動作させることができる．

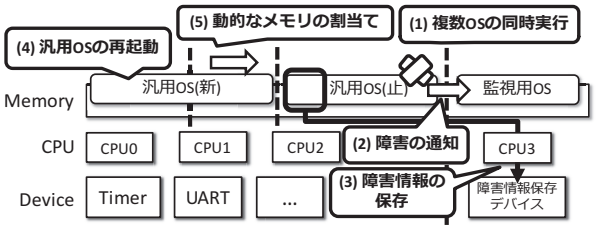

図 1: 障害発生時の動作概要

しかし，LPARはCPUやメモリを複数のOSに分配するため，汎用OSのパフォーマンス低下が考えられる．汎用OSのパフォーマンス低下を最小限に抑えるために，ハードウェア資源の割当てを動的に行うこととする．具体的には，汎用OSが正常に動作しているときは，汎用OSに最大限ハードウェア資源を割り当て，障害発生時は2つのOSが動作できるように割り当てる．

2.2 処理方式

提案手法では，汎用OSと監視用OSを同時に動作させる．汎用OSが正常に動作しているとき，全てのCPUコアやデバイスを汎用OSに割り当て，監視用OSが利用しないメモリ領域の全てを汎用OSに割り当てる．障害発生時の動作概要を図1に示す．以下に処理内容を示す．

(1) 各OSにハードウェア資源を分配し動作させる．
(2) パニックなど汎用OSの障害発生時に，汎用OSは監視用OSに対して障害の発生を通知する．
(3) 監視用OSは，汎用OSが利用していたメモリ領域からメモリダンプの取得と障害情報保存デバイスへの保存を開始する．
(4) 汎用OSの再起動に必要な容量分のメモリダンプの取得・保存が完了後，そのメモリ領域を用いて汎用OSを再起動させる．メモリダンプの取得・保存も継続する．
(5) 汎用OSの再起動中もメモリダンプの収集が完了した領域を随時汎用OSに割り当てる．

以上の手順で，メモリダンプを取得しつつ汎用OSを再起動させ，障害発生前と同等の状態へと回復させる．

A Prototype of Monitoring OS with Fault Information Collector for Dependable Embedded Systems

Ryosuke YAMAMOTO†, Yoshiaki KATAYAMA‡, Shuhei AKETA†, Eiji TAKIMOTO†and Koichi MOURI†

†Ritsumeikan University

‡Mitsubishi Electric Corporation

表 1: 動作環境

項目	内容
ボード	Jetson TK1
CPU	Cortex-A15 4Core
汎用 OS	Linux 3.10.24(NVIDIA 社による改変あり)

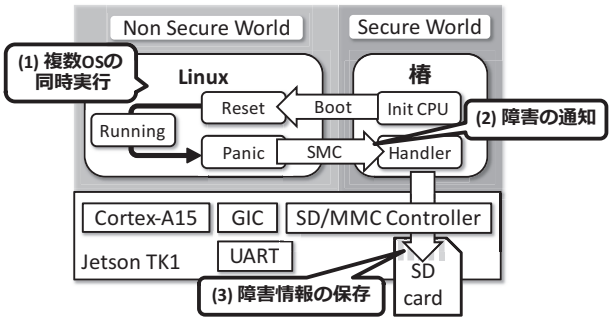

図 2: ソフトウェア構成

3 実装

3.1 実装環境

提案手法を実現するために，監視用 OS として「椿」を実装し，汎用 OS である Linux と同時に動作させた．動作環境を表 1 に示す．

今回，汎用 OS の障害情報を取得するために，提案手法における (1) 複数 OS の同時実行，(2) 障害の通知，(3) 障害情報の保存を行う処理を実装した．ソフトウェアの構成を図 2 に示す．以下，それぞれの実装について述べる．

3.2 複数 OS の同時実行

複数の OS を同時実行させるために，椿は ARM アーキテクチャの拡張機能である TrustZone を利用した．TrustZone は，ハードウェア資源を分割することで，ソフトウェアの動作環境をセキュアワールドとノンセキュアワールドに分割する．そのため，TrustZone は LPAR を実現するにあたり有効である．Linux のハードウェア資源を制限するために，椿はセキュアワールドで動作し，Linux はノンセキュアワールドで動作する．

以下の要素を各 OS に分配することで，Linux と椿を同時に動作させた．

- 割込み
  TrustZone を利用した割込みコントローラの機能を用いることで，各ワールドに割込みを分配する．
- メモリ
  Linux のブートパラメータを変更し各 OS のメモリ領域が重ならないようにした．
- CPU
  椿が各 CPU の初期化処理を行い Linux を起動する．起動後，ホットプラグを利用することで，椿に CPU を割り当てる．

3.3 障害の通知

汎用 OS の障害発生を通知するためには，OS 間での通信を行う必要がある．OS 間でデータを渡す方法として，共有メモリを利用する方法が考えられるが，障害発生時に共有メモリ内のデータの正確性が保証されない．そのため，汎用 OS は Secure Monitor Call(SMC) 命令を実行することで障害発生を通知する．SMC 命令は，ノンセキュアワールドからセキュアワールドに遷移する命令である．Linux が SMC 命令を実行することで，椿に制御を移し，障害の発生を通知する．

3.4 障害情報の保存

障害情報を保存するために，監視用 OS がメモリダンプを取得し，障害情報保存デバイスに対して書込みを行う．汎用 OS を再起動させる際に OS 間でデバイスの競合が発生するため，OS 間でデバイスを共有する必要があるが，これは今後の課題とする．今回は，障害情報保存デバイスとして SD カードを採用し，SD カードにメモリダンプを書き込む機能を椿に実装した．

4 動作確認

前章で述べた実装により，椿が Linux のメモリダンプを収集できることを確認する．

はじめに，椿と Linux を同時に動作させ，Linux 上でパニックを発生させる．パニック発生時に Linux が SMC 命令を実行するように改変することで，椿に対して障害の発生を通知する．椿は，障害の通知を受け，Linux が利用していたメモリ領域のダンプを取得し SD カードに保存する．

以上の処理が行われ，SD カードにメモリダンプが保存されていることを確認した．

5 おわりに

本稿では，組込み機器で動作する汎用 OS と監視用 OS を同時に動作させることで，システムの再起動と障害情報の収集を同時に行う手法を提案した．また，提案手法の実装について述べた．

今後，汎用 OS の再起動，動的に汎用 OS へメモリを割り当てる機能を実装する．また，監視用 OS が障害情報を保存するためのデバイスを占有する課題が存在するため，OS 間でのデバイスの共有方法についても検討する．

参考文献

[1] LKCD Team: LKCD: http://lkcd.sourceforge.net/(2005)

[2] Shimosawa, T. and Matsuba, H. and Ishikawa, Y.: “Logical Partitioning without Architectural Supports”, Computer Software and Applications, 2008. COMPSAC ’08. 32nd Annual IEEE International, pp.355-364(2008).