1J-05

高信頼組込みシステムのための仮想計算機モニタ

古谷晶良† 山本遼介† 片山吉章‡ 水口武尚‡ 明田修平† 毛利公一†

† 立命館大学情報理工学部 ‡ 三菱電機株式会社 情報技術総合研究所

1 はじめに

近年，ハードウェアの高性能化によって，POS レジ，車載システム，スマート家電などの高機能な組込みシステムの需要が高まってきた．これらの組込みシステムの要求を満たすため，多くの組込みシステムでは汎用 OS を広く採用している．しかし，汎用 OS ではデバイスドライバのバグ，長時間の動作などにより障害が発生し，それらは一般に再現が困難である．このような障害発生の原因を特定するために，障害情報の収集を行うことがある．しかし，障害情報を収集する場合，障害情報の収集が完了してからサービスを再開させるため，迅速なサービスの再開を必要とするシステムでは，障害情報を収集することができない．このように，障害発生によるシステムの信頼性低下が問題となる．

以上の背景から，障害情報収集と迅速なシステム復帰の両立を可能とする高信頼システムの開発を進めている．本稿では，その高信頼システムの概要について述べ，そこで利用する組込み用の仮想計算機モニタ（VMM）の開発を中心に述べる．

2 高信頼システムの設計

2.1 課題

汎用 OS がクラッシュした場合，汎用 OS の動作は保証されない．そのため，汎用 OS とは別に，障害情報を収集するためのソフトウェアが必要となる．また，このソフトウェアを汎用 OS のクラッシュから保護するため，それぞれ独立していなければならない．これらより，このソフトウェアを VMM によって実現する．

2.2 VMM の要件

高信頼システムで利用する VMM が満たすべき要件は，以下の通りである．

(1) クラッシュしたゲスト OS とサービスを再開するゲスト OS が別の VM として動作

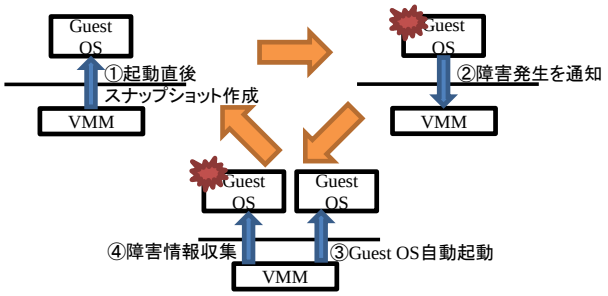

図 1: 高信頼システムの処理概要

(2) パフォーマンス低下を軽減

(3) 障害情報の収集が可能

障害情報を収集するためには，クラッシュしたゲスト OS が使用していたメモリ，レジスタなどの情報を保持しておかなければならないため，(1) が必要となる．迅速なサービスの再開と障害情報の収集を同時に実行しているとき，システム全体のパフォーマンスが低下し，サービス再開までに時間を要するようになってしまうため，(2) が必要となる．

2.3 処理手順

図 1 に高信頼システムの処理概要を示す．

(1) 正常起動時，障害発生後のゲスト OS の高速起動を可能とするために，スナップショットブート用のスナップショットを取得する．このスナップショットには，ゲスト OS 起動直後のメモリ，レジスタ値を保存する．

(2) 障害発生後，ゲスト OS は障害発生を VMM に通知する．

(3) スナップショットを用いて，ゲスト OS を自動起動する．また，別の VM として立ち上げることで，障害情報の損失を防ぐ．

(4) VMM が障害情報を収集する．メモリ，デバイス，CPU，ストレージの情報を収集する．収集した情報は，記憶媒体に保存する．

(5) 障害情報収集後，(1) に戻る．

以上の手順で，サービスの高速な復帰と，障害情報の収集を行う．

A virtual machine monitor for dependable embedded systems

Akira FURUTANI†, Ryosuke YAMAMOTO†, Yoshiaki KATAYAMA‡, Takehisa MIZUGUCHI‡, Shuhei AKETA† and Koichi MOURI†

†Ritsumeikan University

‡Mitsubishi Electric Corporation

3 組込み用 VMM の開発

2.2 節で述べた要件のうち，(1)，(2) の構築を行ったので，それについて述べる．高信頼システムは，既存の VMM を改変し，障害発生後の処理を行う VMM を構築する．また，改変する VMM として，ソフトウェアの規模が小さく，拡張が容易である T-Visor [1–3] を使用する．この T-Visor では，ゲスト OS として Linux を動作させることと，複数のリアルタイム OS による OS 間通信が可能であることが確認されている．ただし，複数の Linux をゲスト OS として動作させることはできない．また，T-Visor は，コアを 1 つしか使用できないため，マルチコアで動作することができない．以上のことから，T-Visor を改変し，複数のゲスト OS(Linux) を動作させることと，T-Visor のマルチコア化を行った．

3.1 複数ゲスト OS の動作

複数のゲスト OS を動作させるためには，VMM のスケジューラによって VM を切り替える必要がある．T-Visor に実装されているスケジューラは，FIFO，ラウンドロビン，リアルタイム OS 用スケジューラがある．この中で，横取り不可能な FIFO やリアルタイム用のスケジューラは，複数のゲスト OS(Linux) を動作させることに適していない．そこで，ラウンドロビンスケジューラを用いてゲスト OS の切替えを行うこととする．T-Visor の実装では，タイムスライスごとに切り替えるような処理にはなっていなかったため，タイムスライスごとに切り替えるように改変を行った．

3.2 マルチコア化

T-Visor をマルチコア化するにあたり，必要となる拡張を以下で述べる．

(1) Application Processor(AP) の起動と初期化
(2) ゲスト OS における CPU コアの電源管理を仮想化
(3) シングルコアを想定して作られた箇所のマルチコア化

(1)，(2) の実装を行ったため，それについて述べる．

3.2.1 AP の起動と初期化

CPU の起動やリセットなどの方法は，CPU ではなく，使用するボード（SoC）に依存する．T-Visor の動作ボートである Cubieboard2 では，AP の開始直後の実行アドレス（エントリポイント）を設定した後に，AP を起動させる．

AP の初期化では，CPU ごとの割込み優先度の設定，例外や割込みの VMM へのルーティング，物理タイマのアクセス権限の変更，ベクタテーブルのベースアドレスを設定などを行う．また，Bootstrap Processor(BSP) と AP では，スタック領域を共有できないため，AP 用のスタック領域を作成する．

3.2.2 CPU コアの電源管理を仮想化

OS によるセカンダリコアのブート，リセット，サスペンドなどの要求は，物理アドレスに直接アクセスされ，電源状態が変更される．そのため，ゲスト OS における CPU コアの電源管理を VMM によって仮想化する必要がある．そこで，電源管理のためのインタフェースとして定義されている Power State Coordination Interface(PSCI) の呼出しをトラップし，エミュレートする．エミュレート関数の中では，CPU_ON と CPU_OFF のみエミュレートを行う．

4 動作検証

複数のゲスト OS の動作，T-Visor のマルチコア化について動作検証を行った．複数ゲスト OS の動作については，2 つのゲスト OS を起動させ，2 つの LED を各々点滅させることで検証を行う．その結果，各 LED が交互に点滅したことから，2 つのゲスト OS が動作していることが確認できた．

マルチコア化については，AP が起動し，初期化処理を行い，電源管理をエミュレートできているか検証を行う．その結果，AP の起動，初期化が行われていることを確認でき，BSP の動作に悪影響を及ぼしていないことも確認できた．また，PSCI 関数のエミュレートが行われていることを確認した．

5 おわりに

本稿では，組込みシステムにおける障害情報収集と迅速なシステム復帰の両立を可能とする高信頼システムの概要について述べ，そこで利用する組込み用 VMM の開発について述べた．VMM では，複数 OS の切替えとマルチコア対応を実現した．

参考文献

[1] garasubo: T-Visor, https://github.com/garasubo/T-Visor (2016).

[2] 島田 工, 矢代 武嗣, 越塚 登, 坂村 健: 組込みコンピュータにおけるハイパーバイザを用いたリアルタイムシステム構成の提案.

[3] Takumi Shimada, Takeshi Yashiro, Noboru Koshizuka, and Ken Sakamura: A real-time hypervisor for embedded systems with hardware virtualization support, TRON Symposium (TRONSHOW) .