B-007

マルチサーバ型 OS のドライバ即時復旧による可用性の向上

Quick Recovery of Failed Drivers on Multi-server Operating Systems to Improve Availability

尾崎亮太† 日高宗一郎†
児玉和也† 丸山勝巳†
Ryota Ozaki,† Soichiro Hidaka,† Kazuya Kodama† and Katsumi Maruyama †

概要 オペレーティングシステム（OS）の障害としては，デバイスドライバの故障に起因するものが特に多い．そのため，特にドライバ故障を考慮した耐故障機能を OS へ付加することが OS の信頼性および可用性向上には重要である．この論文では，マルチサーバ型 OS の可用性を向上する高速かつ軽量にドライバを障害から復旧する手法を提案する．ドライバエラー即時検出およびデバイス初期化スキップにより，ドライバ復旧時間を短縮する．

1. はじめに

近年では，故障要因の多いドライバ障害を考慮した耐故障機能を OS へ付加する手法が注目されている[1),3)]．現在提案されているシステムは，メモリ保護機構などでドライバを論理的に隔離し，障害のあったドライバのみを再起動することで，OS 全体を止めることなくドライバを復旧する．しかしながら，無限ループやデッドロックなどによりドライバが反応しなくなる無反応障害には，単純なタイムアウト待ちが必要となるため障害検出が遅れる．またデバイス初期化を含むドライバ初期化処理も復旧が遅れる要因となる．既存システムではこの遅れは数秒になり，人間に知覚できる程度の時間の OS 動作停止を引き起こす．

そこで本研究では，デバイスドライバのエラーを即座に検出する手法，ドライバ再起動時間を短縮する手法を提案する[2)]．正常なドライバは単純な処理を繰り返し，一定の短い時間でその処理を完了するという特徴に着目し，一定期間を越えても 1 回の仕事を完了しないドライバをエラー状態とみなし，ドライバを復旧処理へ移行させる．また，ドライバ障害時にデバイス状態が正常か否かを判定し，可能ならデバイス初期化をスキップすることでドライバ再起動を短縮する．これにより，ドライバ復旧時間を短縮させることにより OS の可用性向上を目指す．

以降，本論文は次のように構成される．2 章で OS 部分復旧手法およびその問題点について，3 章で問題点を克服する提案手法について述べ，4 章で本論文をまとめる．

2. OS 部分復旧手法

2.1 概　　要

ドライバ故障はドライバ内のエラーを発生させ，ドライバ障害につながる．ドライバ障害は OS 内の他の部分へ波及し，最終的にアプリケーションに対するサービスが停止する．そのため，ドライバ故障に起因する OS 障害を防止するには，まずドライバエラー波及を食い止めなければならない．

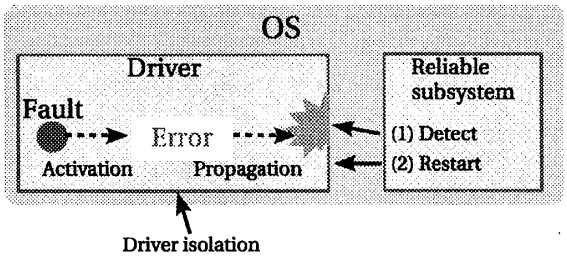

図 1 ドライバ隔離と信頼サブシステム
Fig. 1 Driver isolation and the reliable subsystem.

OS 部分復旧手法では，メモリ保護機構や型安全性言語などを用いてドライバを OS 本体から論理的に隔離する（図 1）．また信頼サブシステム（reliable subsystem）が OS 本体とドライバ間の入出力の監視やドライバ状態の監視を行ない，障害につながるエラーが検出されたドライバを再起動する．障害のあったドライバのみを再起動させることで，OS を稼働したままドライバ障害から復旧する．これにより OS 全体を再起動させる場合に比べ，障害期間を短く抑えることができる．

2.2 ドライバ隔離手法

Nooks[3)] は，Linux カーネルの拡張機能をカーネル本体から分離することで，OS の信頼性を高める OS サブシステムである．拡張機能実行時にカーネル内の他のメモリ領域を書込み不可能にすることで，拡張機能障害のカーネル本体への波及を防ぐ．クラッシュした拡張機能は，復旧エージェントにより自動的に再起動される．

MINIX 3[1)] は，マイクロカーネル＋マルチサーバ構成の OS である．ドライバは個別のユーザプロセスとして実行されるため高い耐障害性をもつ．一般的なユーザプロセスと同様に，ドライバの障害が他へ波及しない．またドライバ再起動はユーザプロセス再起動という形で実現できる．

いずれのシステムもエラーを検出した後，OS 全体を止めることなくドライバを再起動させることでサービスを継続し OS の信頼性を向上している．

2.3 ドライバ復旧遅延

ドライバ復旧が遅れる要因となるドライバエラー検出遅延と

† 国立情報学研究所, National Institute of Informatics

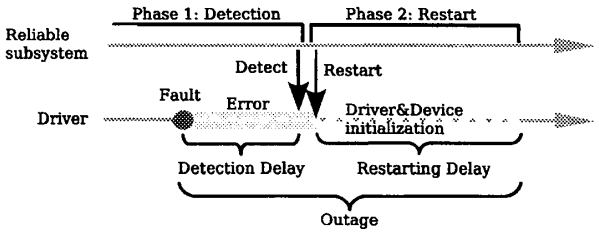

図 2 ドライバ復旧遅延
Fig. 2 Recovery delay of a failed driver.

ドライバ初期化遅延について述べる（図 2）.

2.3.1 ドライバエラー検出遅延

既存の手法はドライバ復旧に大きな遅れを発生させる可能を残す．不正なドライバ入出力やドライバのメモリ保護違反などのエラーによる一般的な障害は信頼サブシステムが即座に検出可能であるが，無限ループやデッドロックなどのエラーによる障害（無反応障害）は信頼サブシステムの定期的な監視を必要とする．監視間隔が長いとその分だけエラー検出が遅れ，結果としてドライバ復旧も遅れる．しかし，監視間隔を短くすると単位時間あたりの監視回数が増え，監視オーバヘッドが増加する．

2.3.2 ドライバ初期化遅延

ドライバ初期化は，プロセス生成などのソフトウェア処理とハードウェアリセットなどのデバイス初期化で構成される．特にハードウェア処理はソフトウェア処理に比べ処理時間が長くなるため，ドライバ復旧遅延の大きな要因となる．

3. 提案手法

本手法は，マルチサーバ型 OS を前提に設計する．ドライバは個別のユーザプロセスとして実行され，ドライバへの処理要求や割込みはプロセス間通信（Inter Process Communication: IPC）でドライバへ通知される．

3.1 軽量なドライバエラー検出手法

短い監視間隔でも，監視オーバヘッドが増加しない軽量なドライバエラー検出手法を提案する．

本手法は，デバイスドライバの正常動作である（a）一連の動作を繰り返す（b）一定の短い時間で処理を終える，という特徴を基に設計する．一定期間を越えても 1 回の仕事を完了しないドライバをエラー状態とみなし，ドライバを復旧処理へ移行させる．マルチサーバ型 OS のドライバは通常（i）要求・イベント待ち（ii）実処理（iii）返答，の処理サイクルを繰り返す．そのため仕事の開始と終了はそれぞれ IPC の受信処理と送信処理とみなすことができる．受信処理の後に一定期間経っても送信処理が行なわれなかった場合，無限ループやデッドロックなどのエラーと判断する．この判断（以下エラー判定）は，ドライバのタイムクォンタム全消費時，タイマ割込み時，プロセス切替え時などのタイミングで行なう．IPC の監視やエラー判定はカーネル内で行なわれるため，エラー検出処理コストは小さい．

表 1 は，提案手法をの IPC による Heartbeat を用いた

表 1 MINIX 3 の手法との比較
Table 1 Comparison with the original.

	チェック方法	ドライバ修正	対タスク数コスト増加比
MINIX 3	IPC	必要	大
提案手法	カーネル内	不要	小

MINIX 3 の手法を比較したものである．MINIX 3 の手法は監視間隔毎に $IPC \times 2 \times n$（n：タスク数）の処理が必要となる．一方，提案手法では，IPC 毎に代入文 1 回およびエラー判定毎の処理がカーネル内で行なわれる．

3.2 ドライバ高速再起動

ドライバエラーが起きたとしても，デバイスも同時に異常状態になるとは限らない．デバイス初期化を行なわず，ドライバ（ソフトウェア）の初期化のみ行なうだけでドライバ復旧できる可能性も存在する．

デバイスが正常か否かは，ドライバがデバイスにどのような操作を行なったかで判断できる．そこで，ドライバが発行する I/O 命令を監視し記録する．ドライバ障害時に履歴を参照し，不正な I/O を発行していた場合はデバイスが異常状態であると判断する．本研究では，プロセッサ仮想化技術を利用し，ドライバが発行するすべての I/O 命令を監視する．

上記のように判断した結果をドライバ再起動に反映させ，可能ならばドライバ初期化時にデバイス初期化をスキップすることでドライバ高速再起動を実現する．

4. おわりに

マルチサーバ型 OS を対象にドライバ復旧遅延を短縮する手法について提案した．軽量なドライバエラー検出は，無反応障害を素早く検出し，またドライバ高速再起動は，デバイス初期化をスキップすることでドライバ再起動を短縮する．これにより障害期間が短くなり，OS の可用性が向上する．今後は提案システムを MINIX 3 上に実装，評価を行なうことで提案手法の有効性を示す予定である．

参考文献

1) Herder, J.N., Bos, H., Gras, B., Homburg, P. and Tanenbaum, A.S.: Failure Resilience for Device Drivers, Proc. *the 37th International Conference on Dependable Systems and Networks (DSN '07)*, pp.41–50 (2007).
2) Ozaki, R., Hidaka, S., Kodama, K. and Maruyama, K.: Quick and Lightweight Detection of Anomalous Drivers in Multi-server Operating Systems to Improve Availability, Proc. *the 37th International Conference on Dependable Systems and Networks (DSN '07)*, fast abstract, pp.378–380 (2007).
3) Swift, M.M., Bershad, B.N. and Levy, H.M.: Improving the Reliability of Commodity Operating Systems, Proc. *the 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, pp.207–222 (2003).