


(様式 5)



2018 年 12 月 10 日
Year Month Day

学位（博士）論文要旨

(Doctoral thesis abstract)

論文提出者 (Ph. D. candidate)	工学府博士後期課程 電子情報工学 専攻 (major)
	平成 28 年度入学(Admission year) 学籍番号 16834305 氏名 高山 献 (student ID No.) (Name)
主指導教員氏名 (Name of supervisor)	並木美太郎 
論文題目 (Title)	オペレーティングシステムのソフトウェアバグからアプリケーションの実行を保護する手法に関する研究
論文要旨 (2000 字程度) (Abstract(400 words)) ※欧文・和文どちらでもよい。但し、和文の場合は英訳を付すこと。 (in English or in Japanese)	
<p>コンピュータシステムの信頼性は主に、狭義の信頼性 (Reliability), 可用性 (Availability), 保守性 (Serviceability), 保全性 (Integrity), 機密性 (Security) で評価され、一般的には故障間隔, 稼働率, 修理時間などで評価される。オペレーティングシステム (OS) はマシン全体を管理するため, OS の信頼性が高いことは, その上で稼働するアプリケーションプログラム (App) の信頼性を担保するための必要条件である。</p> <p>OS のソフトウェアバグはシステム全体の信頼性を低下させる要因である。例えば, OS カーネル内でバッファオーバーフローが発生すると, 不正なアドレスにメモリ書き込みを行ってしまうこともある。OS 内のメモリオブジェクトが破壊されると, App のユーザ空間のメモリを破壊したり, 破壊されたメモリオブジェクトを参照した場合に OS がクラッシュしたりする。障害が発生した OS での実行を続ければ, エラーは更に拡大し App や OS が意図しない挙動をするようになる。エンドユーザにとって, OS のバグを解決する現実的な方法は OS の再起動である。OS の再起動によって破壊された実行状態を捨てて, 新しい OS の実行状態を作り出すことができる。またバグを修正するためのアップデートを適用するためにも, 再起動は有効である。</p> <p>OS の実行状態の中には App の実行状態も含まれており, OS の再起動によって稼働していた App の実行状態も失われる。Checkpoint によって再起動前の段階で App の実行状態を保存することができていれば, OS 再起動の後に App の実行を再開することができる。しかし</p>	

Checkpointの取得頻度が高ければランタイムオーバーヘッドが増し、低ければ再開したAppの実行状態が古いため障害発生直前の状態になるまで時間がかかるようになるというトレードオフが存在する。OSカーネルのアップデートをOSカーネルの実行中に適用するMemory Patchingでは、Appの実行を停止すること無くアップデートが適用でき、OSの障害を引き起こすようなバグを修正することも可能である。しかしすべてのアップデートで利用可能なわけではなく、コードのみ修正を行うものは得意であるものの、データ構造の修正を伴うものには対応していないこともある。

本論文では、Appの実行環境を抽象化したプロセスという概念は歴史的に変わっていないことに着目し、Appの実行に必要なプロセスの実行状態を *Essential Context* として定義する。Essential Contextを用いることで、OSは再起動後もAppの実行を継続することができる。まず、信頼できないOS上で動作するAppのEssential Contextを作成する手法、*ShadowBuddy*を提案する。ShadowBuddyはOSよりも強いハードウェア権限を持ち、OSのエラープロパゲーションからAppの実行状態を保護しつつ、AppとOSの挙動を監視してEssential Contextを更新する。ShadowBuddyでは保護のためにランタイムオーバーヘッドが発生し、Appのスループットが低下する。そこでOSカーネルのバグの修正のために行われるアップデート時に絞り、ランタイムオーバーヘッドをなくしつつサービスダウンタイムを低減する最適化手法 *Dwarf*を提案する。両手法の提案によって、OSのソフトウェアバグによってシステム全体の信頼性が低下することを防ぐことができた。Appの実行状態が破壊されることを防ぐことによって保全性を高め、ダウンタイムを短縮することによって可用性と保守性を高めることができる。両手法は仮想マシンモニタ (VMM) のXenと広く用いられているOSのLinuxに対して実装を行い、OS再起動時にもAppを継続して実行できることを示す。

(英訳) ※和文要旨の場合(400 words)

Dependability is mandatory in current computer systems. It consists of 5 factors: Reliability, Availability, Serviceability, Integrity, and Security. Operating systems (OSs) are crucial to achieving high dependability because they are the primary software to manage underlying hardware and application programs (Apps).

Software bugs at the OS-level significantly affect the dependability, and are more serious than ones at App-level because App-level bugs affect only the App while OS-level bugs infect all Apps running on it. OS reboots are commonly the only solution for most end-users to recover from OS-level bugs because neither its cause nor recovery method are hardly determined in most cases. OS reboots are also used for OS-level bug fixes to launch newer OS instance.

The restarted App loses its context, such as its register values, process control block, and user-space memory, on the rebooted OS. Checkpointing is one solution to keep App's context, but it causes tremendous runtime and memory overhead to take checkpoints continuously to prepare for occurring failures in the OS. Moreover, the restarted App from checkpoint does not have the latest status in most cases because we cannot take App's latest checkpoint just before the failure. We need to redo the execution to the point manually and carefully. Dynamic patching overwrites running OS execution flow to apply the OS kernel updates. However, it is mainly limited to small code fix because not all updating memory objects are definitely and

automatically solved.

This dissertation introduces *Essential Context* which has the latest status of the running App process, focusing on the fact that process, the abstraction of computer resource, has almost unchanged in decades. With Essential Context, we can restart from the App context just before the fault. Based on the concept of Essential Context, we first introduce *ShadowBuddy*, a technique to create Essential Context outside the untrusted OS. It protects the user space memory of the target App from illegal kernel writes to preserve the intact memory context, and monitors system calls executed by the App and device I/Os conducted by the OS to take running OS status from outside of the OS without believing OS kernel memory objects. The rebooted OS can restore the App status just before the OS fault from Essential Context. *Dwarf*, presented secondly, is an optimization of ShadowBuddy for OS kernel updates. Dwarf constructs Essential Context from memory objects in the OS kernel. When updating an OS kernel, we trust the running OS status because it is conceivable that there is no failure in the OS. Essential Context keeps the latest running App status, so that we can run Apps even on the untrusted OS.