カーネルソフトウェア開発支援ツール
―動作情報トレースによるカーネルソフトウェアデバッグの圧倒的効率化―

1. 背景

カーネルソフトウェアとは OS（Operating System）の中で直接動作するようなソフトウェアである. OS の中核機能をカーネルと呼ぶため, 本プロジェクトでは OS をカーネルと呼び, カーネルの中で動作するソフトウェアをカーネルソフトウェアと呼ぶ. このカーネルソフトウェアにはハードウェアの制御を行うデバイスドライバやファイルへの操作を提供するファイルシステムなどがあり, いずれも OS の基本動作を担う重要なソフトウェアである. 特にデバイスドライバはハードウェアの多様化に伴い年々増加の傾向が見られ, 報告されるバグの件数も増え続けている. このようなバグは単なるソフトウェアクラッシュに留まらず, カーネルを巻き込んだシステム全体のクラッシュにつながる.

よってカーネルソフトウェアは開発段階において入念なデバッグが必要とされる. 従来, ハードウェアベンダーなどは自社専用のデバッグ, テストツールを開発し利用してきたが, 実際の開発作業において対話的に効率よくデバッグを行えるような物は少なかった. そのためカーネルソフトウェアの開発者は依然として古典的なデバッグ手法に頼りきっている場合が多い.

2. 目的

本プロジェクトではカーネルソフトウェアのデバッグ, テストを効率的に行う支援ツールを開発する. ユーザモードソフトウェアのデバッグに用いられていた Timeless Debugging というデバッグ手法をカーネルソフトウェアのデバッグにも採用することで, デバッグ効率を格段に上昇させることを目標とした. 結果として, カーネルソフトウェアの動作をデバッグ対象部分のみ詳細に記録することで, Timeless Debugging を行うことができるようになった. また記録された情報をもとに, デバッグだけでなく Valgrind の memcheck が行っていたメモリリーク自動検出の実装にも成功した. これによりカーネルソフトウェアのデバッグを非常に効率的に行うことができるようになった.

3. 開発の内容

本プロジェクトは Kernel Analysis Platform（以下 KAP）という名前で, カーネルソフトウェアのデバッグ, テスト, 解析を支援するプラットフォームになっている. 本プラットフォームはデバッガ, テストツールのユーザインタフェースを実装した KlareDbg, kvalgrind の 2 つ, およびカーネルソフトウェアの動作情報を記録するバックエンドである QEMU 改, K2E の 2 つ, 計 4 つからなる. バックエンドはそれぞれ仮想マシンで OS ごとエミュレートさせる方式（QEMU 改）と, 実機を利用しつつ解析対象のカーネルソフトウェアのみエミュレート（インタプリタ実行）させる方式（K2E）の 2 つを取っている（図 1）.

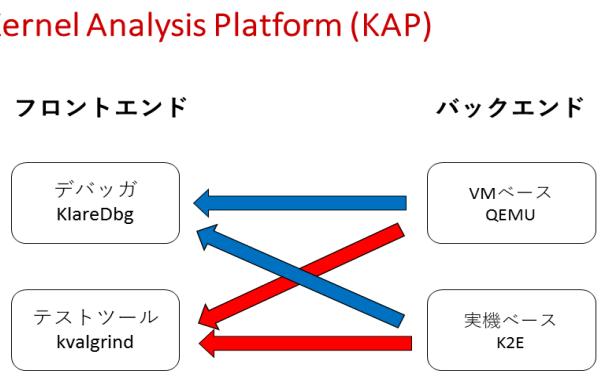

図 1 Kernel Analysis Platform の全体像

図 2 に KlareDbg の外観を示す. 画面上がデバッグ中のカーネルソフトウェアの命令ダンプであり, 開発者は上下カーソルキーによってこのダンプを縦横無尽に動き回る. そしてそれに追従する形で画面下のレジスタの値がその時の状態に復元される. このようなことが可能なのは, バックエンドの QEMU 改および K2E が命令ごとの動作記録を取り, フロントエンドの KlareDbg が再生しているためである. kvalgrind も同様にこれらの動作記録をもとにメモリリークの検出を行う.

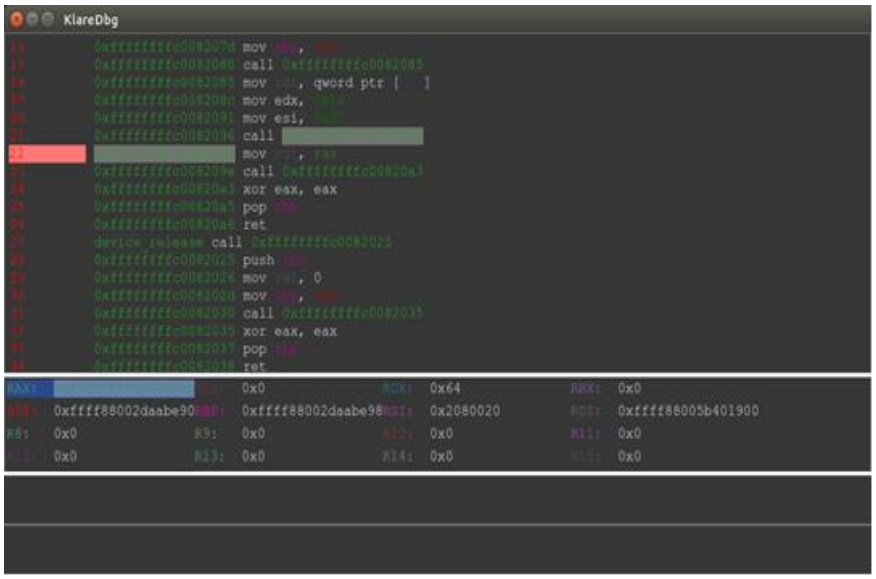

図 2 KlareDbg のインタフェース

続いてバックエンドの QEMU 改のシステム構成を図 3 に示す. こちらは仮想マシンで OS ごとエミュレートさせる方式であり, 実際のハードウェアが無くてもデバイスドライバのデバッグなどが可能である. QEMU は通常, 動的バイナリ変換という手法を用いてゲスト OS 全体をエミュレートする. これはゲスト OS 実行時, Just In Time で実行バイナリを中間表現に変換し, ホスト OS 用の実行バイナリに変換して実行する方式である. 本システムではこの変

換フェーズに改造を施し，動作記録用の余分な命令を挿入してホスト OS 向けバイナリに変換するようにした．これによりカーネルソフトウェア実行時のみ，レジスタやメモリへの読み書き動作を記録としてデータベースに保存し，KlareDbg や kvalgrind といったフロントエンドから読み込むようになっている．

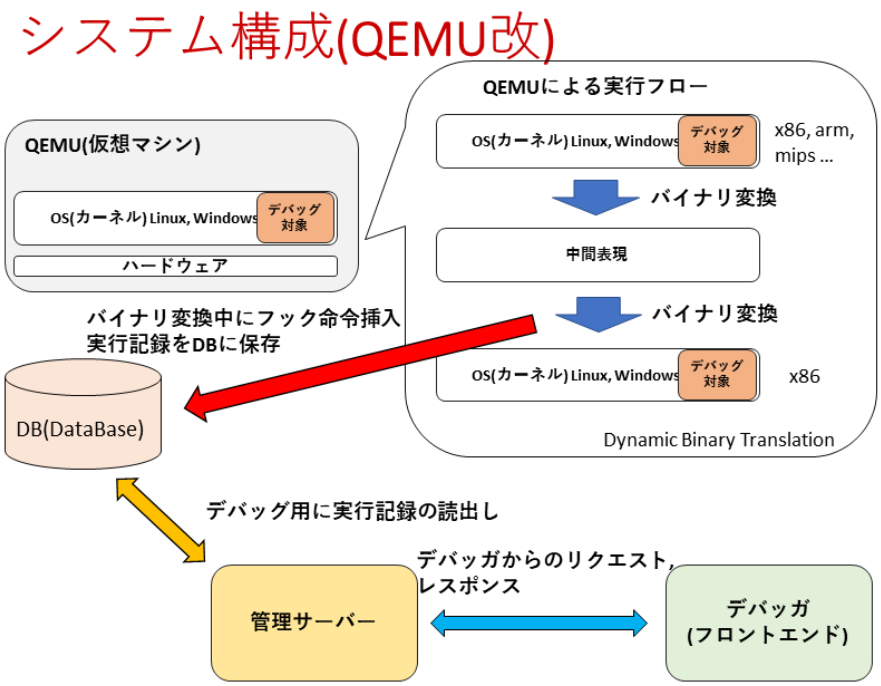

図 3 QEMU 改によるバックエンドの構成

次に実機を利用しつつ，カーネルソフトウェアのみエミュレートさせる K2E のシステム構成を図 4 に示す．K2E では BitVisor というハイパーバイザを改造することで，通常実行はネイティブとほぼ同じ速度で行い，デバッグ対象のカーネルソフトウェアが実行された時のみ内部のインタプリタが代わりに実行を代替する．これによりデバッグ対象ソフトウェアのみを詳細にステップ実行し，QEMU 改同様，レジスタやメモリへの読み書き動作を記録することができる．加えて K2E には C++の標準ライブラリが全て移植されており，拡張プラグインの記述にオブジェクト指向を用いることができる．拡張プラグインはカーネルソフトウェアのカスタムなテストの記述に用いることができる．

上記 2 つのバックエンドにより，実際のハードウェアが存在しないデバイスドライバには QEMU 改を用い，存在する場合は K2E で高速にデバッグするといった選択が可能である．

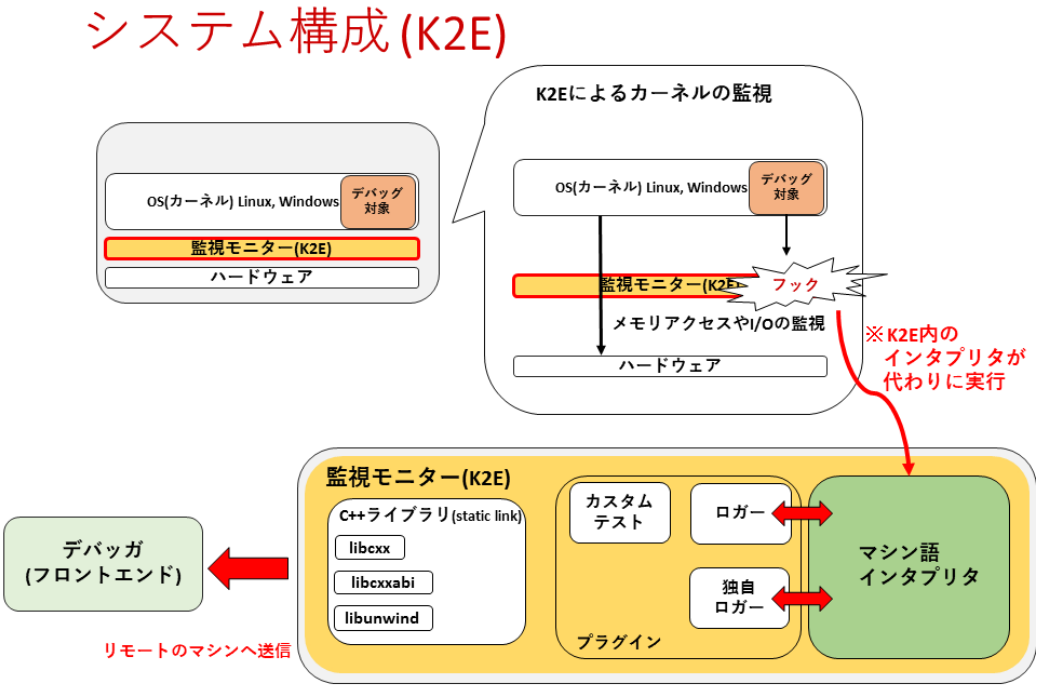

図 4 K2E によるバックエンドの構成

4. 従来の技術(または機能)との相違

カーネルソフトウェアのデバッグ，テストツールはいくつか存在するが，いずれも Timeless Debugging のような縦横無尽なデバッグは行えず，またテストもカーネル固有の設定が必要な場合が多い．本プロジェクトのデバッガ，テストツールは，仮想マシンやハイパーバイザを用いることで，カーネルやアーキテクチャに依存しない柔軟な解析機構を提供する．さらに，開発者が容易にテストを拡張できるよう，プラグイン機構を提供することで，ニーズにあわせたテスト手法を選択することができる．これによりカーネルソフトウェア開発にかかる労力が大幅に低減される．

5. 期待される効果

カーネルソフトウェアのデバッグやテストが効率的に行えるようになることで，ハードウェアベンダーなどにおけるデバイスドライバ等の開発コストが下がり，より多くの安定した組込機器の登場が期待される．また既存のソフトウェアに対しても，本プロジェクトの成果物を利用することで多くのバグが発見できる可能性もある．

6. 普及(または活用)の見通し

本プロジェクトはオープンソースプロジェクトとして，常に機能追加，バグ修正による安定化を図っていく．ある程度の安定版ができればそれを都度リリースし，世界中の開発者に対して普及を目指していく．

7. クリエータ名（所属）

木村 廉（神戸大学）

（参考）関連 URL

- KlareDbg
  https://github.com/KernelAnalysisPlatform/KlareDbg
- kvalgrind
  https://github.com/KernelAnalysisPlatform/kvalgrind
- K2E
  https://github.com/RKX1209/k2e_public