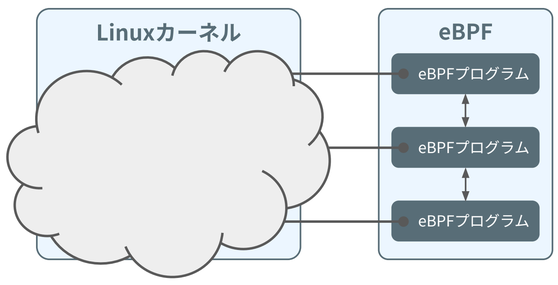
eBPFプログラムの機能正当性検証に向けた検証フレームワークの提案

神 拓己　五十嵐 淳
京都大学

研究の概要

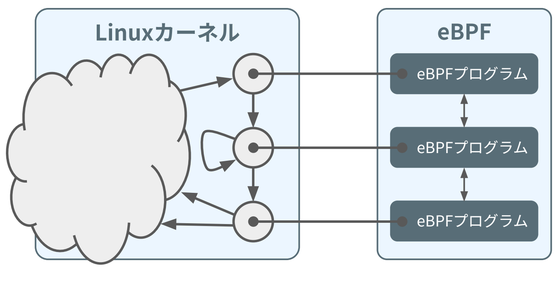
- Linuxカーネルと**密に**結合するeBPFプログラムを検証するためのフレームワークの提案
- 本ポスターではsched_extという技術に着目

既存研究



Linuxカーネルの状態を**モデル化しない**

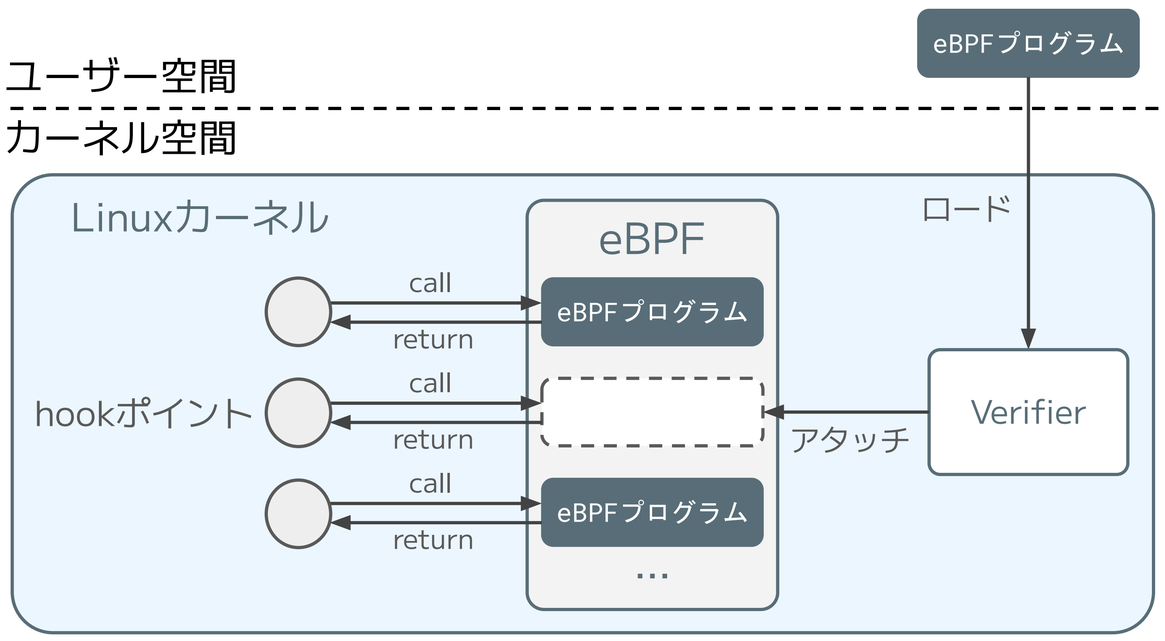
本研究



Linuxカーネルの状態を**一部モデル化**

eBPFとは？

- Linuxカーネルの機能を**安全に拡張**するためのイベント駆動形の**カーネル拡張技術**
- eBPFプログラムのロード時に静的に検証が行われるため安全（無限ループがない、メモリ範囲外アクセスがない等）
- 検証を可能にするため、プログラムの表現能力に制限がかけられている→**正当性検証へのハードルも小さい**



sched_extとは？

- eBPFでスケジューラを実装できるようにする**超新しい**技術
- スケジューラを小さな処理に分割し、それぞれの処理に対応するeBPFプログラムが協調して動作することでスケジューラの動作を実現

select_cpu	タスクが起床したときにタスクをどのCPUに紐付けるかを選択するために呼び出されるeBPFプログラム
runnable	タスクが実行可能状態になったときに呼び出されるeBPFプログラム
update_idle	CPUがidle状態になるときに呼び出されるeBPFプログラム

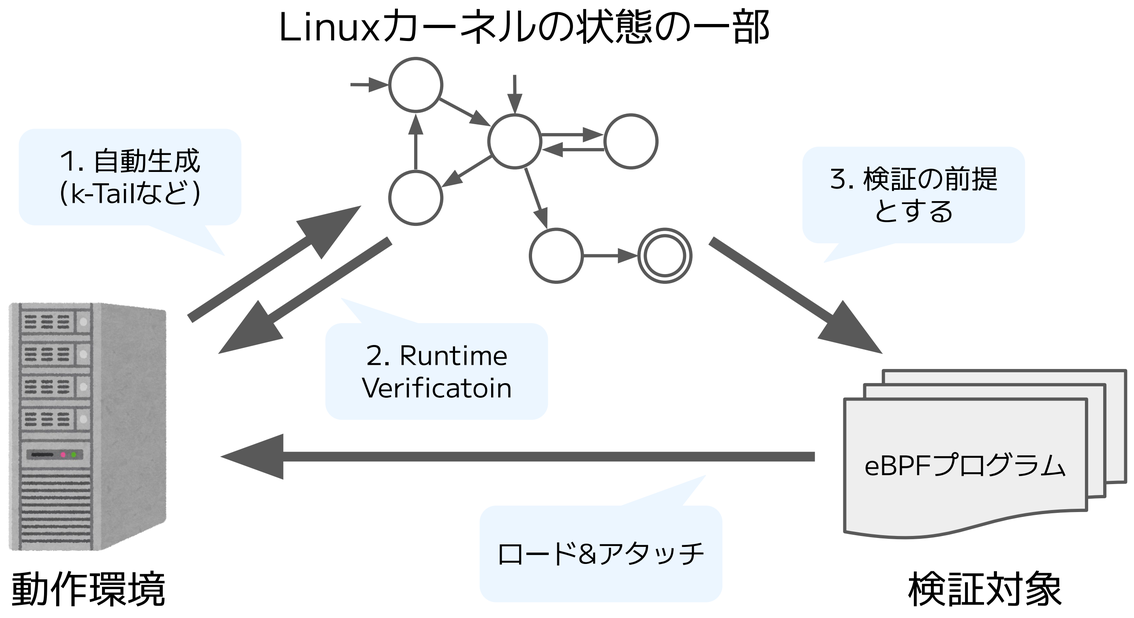
検証したい性質と問題点

複数のeBPFプログラムが**同時に**動作することで実現される性質を検証したい

- 例）実行可能タスクが存在しているのに、CPUがidle状態に遷移してしまわないか（★）
- 例）常に最高優先度のタスクにCPUリソースを割り振れているか

eBPFプログラム間のやりとりや
カーネルの状態のモデル化が必要

フレームワークの全体像

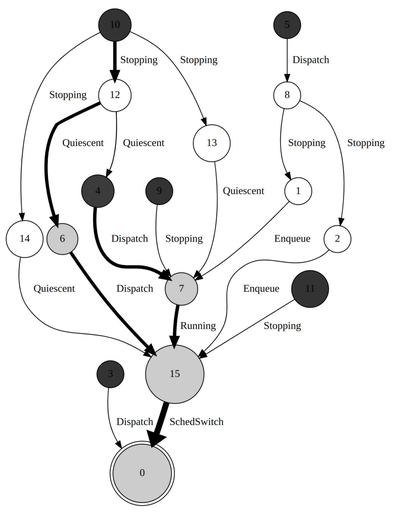


1. カーネルイベントの発生履歴をもとに、k-Tailなどの学習アルゴリズムを用いてオートマトンを**自動生成**
2. オートマトンが正しいことを**Runtime Verification**によって実行時に検証
3. カーネルを近似したオートマトンを**前提に**eBPFプログラムの検証を行う

eBPFプログラムの呼び出し順序についてのオートマトン学習の結果の例：

```
stopping
quiescent
dispatch
running
sched_switch
-----
stopping
sched_switch
-----
dispatch
sched_switch
-----
[...]
```

BPFプログラムの
呼び出し履歴



BPFプログラムの呼び出し順序
を表現したオートマトン

本フレームワークの特徴

- カーネルの複雑な内部状態をオートマトンで近似することで、**カーネルと密に連携するeBPFプログラムの検証を可能にする**
- カーネルの内部状態は複雑でモデル化するのが難しいため、学習アルゴリズムを用いてモデル化を手助け
- **カーネルで提供されている技術を有効活用**
  - トレーシング技術（Tracepoint, kprobe, etc..）
  - 実行コンテキストでイベント履歴を区切ることでモデルの学習の精度を向上

現状と今後の展望

- 現状は、学習したオートマトンをもとにSpinを用いて性質★を検証している
- 今後はeBPFのバイトコードやソースコードから必要な情報を取り出し、自動で検証できるようにする
- その他の検証手法の適用を試みる（KLEE, Frama-C, Uppaal, etc..）