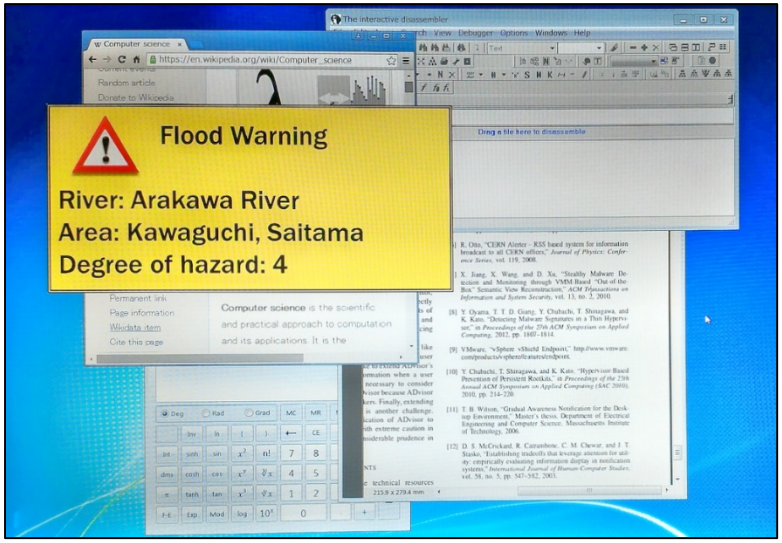
BitVisorによる OSの見かけ上10倍速実行

大山恵弘
筑波大学 学術情報メディアセンター

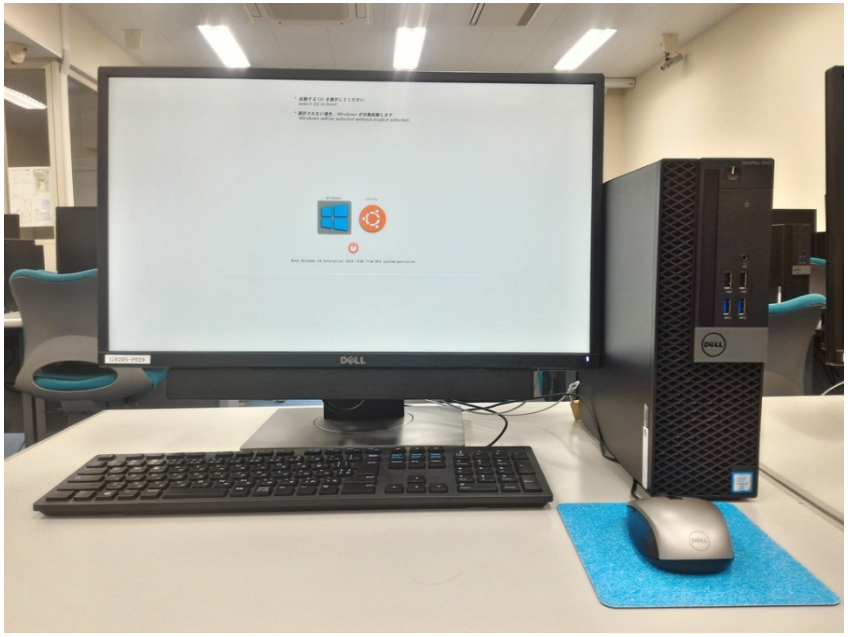
やってきたこと

- BitVisorベースのちょっと変なハイパバイザを学生とともに考えてきた
  - 災害警報を表示するハイパバイザ
  - 青少年に見せられない部分を黒塗りするハイパバイザ
  - 広告を表示するハイパバイザ
  - 旧バージョンのSSL通信に警告するハイパバイザ
  - アクセスしたディスクブロックやネットワークパケットからマルウェア文字列を検出して警告するハイパバイザ



BitVisorに関してやっていること

- 筑波大学の教育用システムの全端末（1000台以上）で動くvThriiの運用
  - BitVisor Summit 6で発表した
    - 「大学の教育研究用端末上でのベアメタルハイパバイザの運用」
  - 自分の部署の月例会で，ほぼ毎月，vThriiをどう安定的に運用するかの議論を続けている



今日の話

- ゲストOSの時間の流れを速くするための，BitVisor ベースのハイパバイザ
  - OSが認識する時間の流れる速度を変えるだけ
  - 計算の実際の速度は従来通り
  - OSが計時に用いるTSCの増加幅を増やして実現
    - TSC = Time Stamp Counter
    - RDTSC命令が返す値を書き換える

作ったシステム

- あるキーを押すと，ゲストOSが認識する時間の流れる速度が10倍になる
  - ソフトウェアの状態が10倍程度速く変化する
- 別のキーを押すと，1倍に戻る
- 10倍→1倍→10倍→1倍→...と，何度でも繰り返せる

まずはデモ動画を見ていただくのがいいと思います

他に可能な操作

- あるキーを押すと，N倍速がN+1倍速になる

- あるキーを押すと，N倍速がN-1倍速になる
  - ただしNは1未満にはしない

- 2倍や60倍にする
  - 60倍速にしてもOSやアプリは落ちないことを確認ずみ
  - 画面遷移速度や動画再生速度は60倍にはならない
    - 意図通りの速度になるかどうかは，処理がCPUなどの資源バウンドか，待ち時間バウンドかによる

動機

- 時間の流れを速く（遅く）できる利点は多い
  - 実行の様子を短時間で確認できる
  - 変化が速すぎて人間がついていけない処理を，低速にできる
  - 長い待ち時間をスキップできる
  - 長い時間の経過がトリガになって実行される処理（マルウェアの感染処理など）を，すぐに実行させることができる

  - まじめな研究の話をしますと...
    - TSCを用いたマルウェアによる解析回避処理を無効化できる

考えられる拡張

- 速度変更のトリガを，キー入力ではない何かにする
  - 特定の時刻になったら
  - 特定のアプリケーションが立ち上がったら
    - 例： たるすぎるゲーム
  - 特定の文字列が画面に表示されたら
    - 例：「5秒後にスキップして動画へ進めます」
  - PC付属のカメラが認識する顔が正規ユーザの顔でなかったら
    - 端末を不正利用したユーザが驚いて退散する

実装

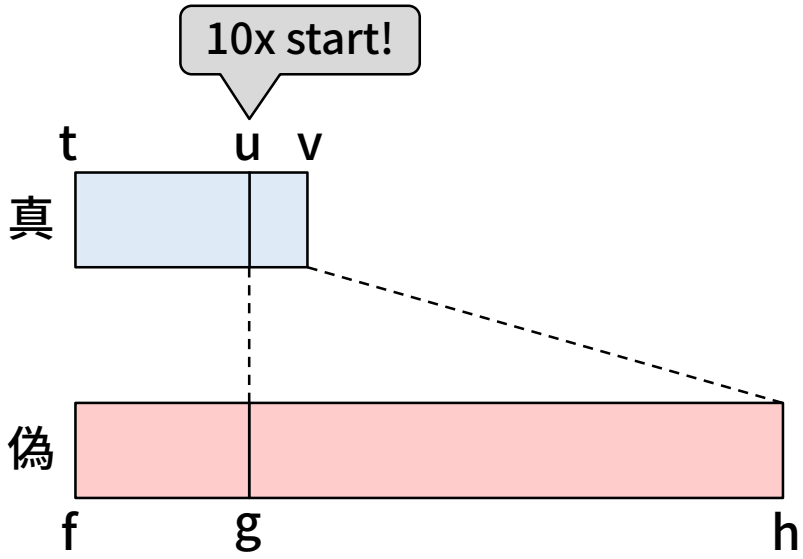
- ゲストOSによるRDTSC命令の実行をBitVisorがフック
  - VT-xの機能を利用する
    - Intelマニュアル Appendix C VMX Basic Exit Reasons を参照
    - Exit Reason第16番がRDTSC
    - （Exit Reason第51番がRDTSCP）
  - BitVisorの初期化処理で，これらの命令でVM exitが発生するようにしておく
- 返すTSCの値を偽装する

RDTSC命令

- TSCを読む命令
  - TSC上位32ビット → EDXレジスタ
  - TSC下位32ビット → EAXレジスタ
  - TSCとは？
    - x86 CPU内にある64bitのカウンタ
    - 一定時間ごとに1増えたり，1クロックごとに1増えたりする
      - 最近では，CPU周波数の変化や命令実行のストールには関係なく，一定のペースで増加
- OSカーネルが経時に標準的に用いる
  - LinuxやWindowsなどの多くのOSで，HPETやPITより優先
- 一般のユーザプログラムが経時に用いることもできる

TSCの値の偽装

- RDTSCの最初の実行：
  - その時点での真のTSCの値tを，
    その時点での偽のTSCの値fとする
    - 最初は両者は一致
- 10倍速開始キーの入力：
  - その時点での真のTSCの値uと
    偽のTSCの値gを記憶
- 10倍速中のRDTSCの実行：
  - その時点での真のTSCの値vを取得
  - その時点での偽のTSCの値 h = g + (v - u) * 10 を返す



CPUIDが返す値の偽装

- 一部のCPU機能のサポートを示すビットを落とす
  - TSC deadline:
    - TSCが所定の値になったら割り込みを発生させる機能
    - 最近のLinuxが計時に使っている？
    - RDTSCを仮想化したのに，TSC deadlineを仮想化しないと，TSCの辻褄が合わなくなる
      - OSが固まる
      - しかし，TSC deadlineの仮想化の実装は面倒 → 実装するのやめた
  - RDTSCP:
    - out-of-order実行されないことが保証されたRDTSC
    - 対応はできそうだが，実装量を極限まで最小化するために偽装
- TSC adjustやinvariant TSCのビットはそのまま

ブート時のTSC calibration

- Linuxはブート時に，1秒あたりのTSCの増加幅を，実際の測定値から求める
  - PITで測った一定時間内のTSCの増加幅を測定
- ブート中は，TSCもPITも偽装しないようにする
  - 1倍速時のPIT速度とTSC速度の比を正しく認識させる

```
[    0.000000] tsc: Fast TSC calibration using PIT
[    0.000000] tsc: Detected 3192.973 MHz processor
[    0.000064] Calibrating delay loop (skipped), value
calculated using timer frequency.. 6385.94 BogoMIPS
(lpj=12771892)
```

マルチコア対応

- TSCはコアごとに存在し，それらの値は異なる
- よって，今回の実装でも，コアごとに，真のTSCの値と偽のTSCの値を管理

ソースコードの修正

- 修正した（というか追加した）行数： 約230行
  - 極限まで手抜き実装した結果，ここまでコンパクトに

- 修正したファイルと行数
  - core/cpu_emul.c　　5行
  - core/io_iohook.c　　22行
  - core/vt_init.c　　4行
  - core/vt_main.c　　約200行

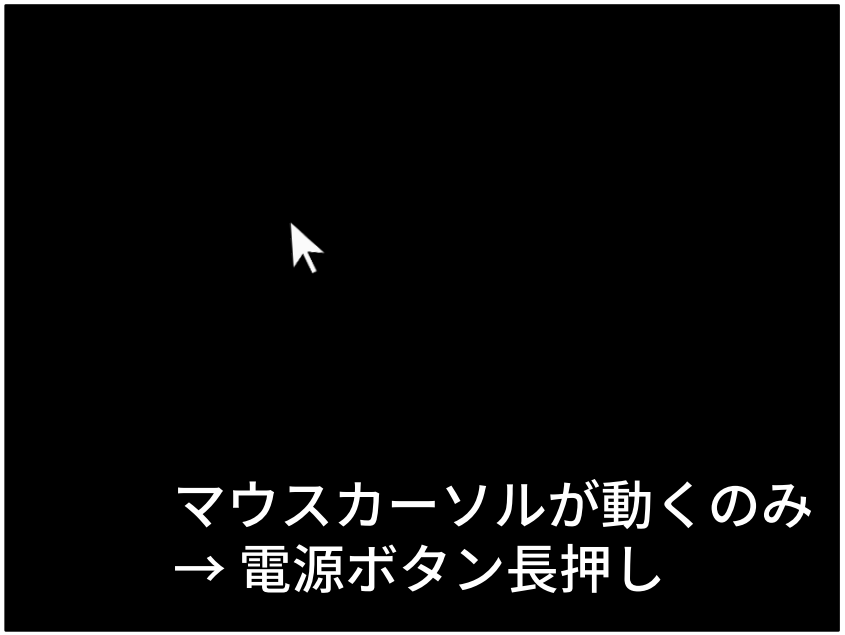
ゲストOS側で必要な修正

- OSの設定でインターネット時刻との同期をオフにする
  - これをしないと時刻が頻繁に前に戻り，時計アプリケーションが固まる

- Linuxのブートオプションに `tsc=reliable` を付ける
  - 時間の源（clocksource）として常にTSCを使うようになる
  - これがないと，OSがTSCのおかしさを検出し，clocksourceをHPETなどに切り替える
    - Linuxは，TSCの値の信頼性をPITを使って定期的に検査している（ような気がする）

Windowsゲストの加速/減速

- まだ成功してしない
- 10倍速キーを押した瞬間，画面真っ黒
- 以降は何をしても反応しない
- `tsc=reliable` に相当するオプションを与えていないのが原因か
  - 探したが，多分Windowsにはそういうオプションがない



人間の操作に与える影響

- キー操作が難しくなる
  - リピート開始までの時間が1/10
  - リピート速度が10倍

議論

- タイムスライスやコンテキストスイッチはどうなる？
  - 実質的なタイムスライスはおそらく1/10に変化
  - よって，実質的なコンテキストスイッチ頻度は10倍に変化
    - コンテキストスイッチのオーバヘッドが際立つ

- 特定のプロセスだけ10倍速にすることはできるか？
  - わからない
    - どのRDTSCを偽装するかは難しい問題
    - そのプロセスのスケジューリング中だけ，カーネルが実行するRDTSCを偽装するようなことが可能か？
  - clock_gettimeやgettimeofdayをどう偽装するのか問題もある

関連システム

- エミュレータの加速/減速機能
  - この機能を持つエミュレータは多い
  - ベアメタルハイパバイザかエミュレータかという差分はあるが，やっていることは同じ
- 加速/減速機能を持つ音楽や動画の再生アプリ
  - ポッドキャストアプリとか
- HyperSlow [Okamura et al., '12]
  - 本研究の前身
  - 仮想時間の流れを極端に速くしてマルウェアを動けなくする

うさみみハリケーン 加速/減速機能

今後考えたいこと

- 有用な目的への適用
  - マルウェアの解析や制御
  - ソフトウェア開発の高速化
- タイミングベース攻撃との相互作用の考察
  - MeltdownやSpectreなど
- もっとちゃんと作り込む
  - ハイレベル国際会議に通るならやりますけど...