とある金融機関システムにて、
カーネル初心者が
Linuxカーネルコードと格闘しながら
eBPFでオブザーバビリティを高めた話

2024/09/06

Daiki Tamura

Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

自己紹介

名前：田村 大樹／Daiki Tamura

所属：日鉄ソリューションズ株式会社
技術本部システム研究開発センター

担当：クラウドネイティブ技術に関する研究開発、事業支援、人材育成
SREとしてのプロジェクト参画

好きなこと：
・仕組みを深ぼること
・学んだ技術を他者に伝えること

趣味：オーケストラ（クラリネット）

Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

はじめに

- 本発表の内容は、エンジニア個人の見解であり、所属する企業の公式見解ではありません。

- NS Solutions、absonne（ロゴ）、emerald（ロゴ）、CloudHarborは、日鉄ソリューションズ株式会社の登録商標又は商標です。
- Oracle、Java、MySQL及びNetSuiteは、Oracle Corporation、その子会社及び関連会社の米国及びその他の国における登録商標です。NetSuiteは、クラウド・コンピューティングの新時代を切り開いたクラウド・カンパニーです。
- その他本文記載の会社名及び製品名及びロゴは、それぞれ各社の商標又は登録商標です。

Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

オンラインセッションで紹介した話

背景：とあるシステムの性能改善チームに参画したときの話

- ミリ秒単位の厳しいレイテンシ目標
- カーネル内部についても詳しく分析したい

目的：「スケジュールレイテンシ」の仮説を、本番環境で検証したい

アプローチ：eBPF技術を使って軽量に計測しよう

だが、しかし！

ツール入れて動かせば終わり、なんていう穏やかな話ではなかった・・・

Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

アジェンダ

- スケジュールレイテンシとは
- eBPFとは
- 実装に向けた技術課題と解決
- 結果と考察
- まとめ

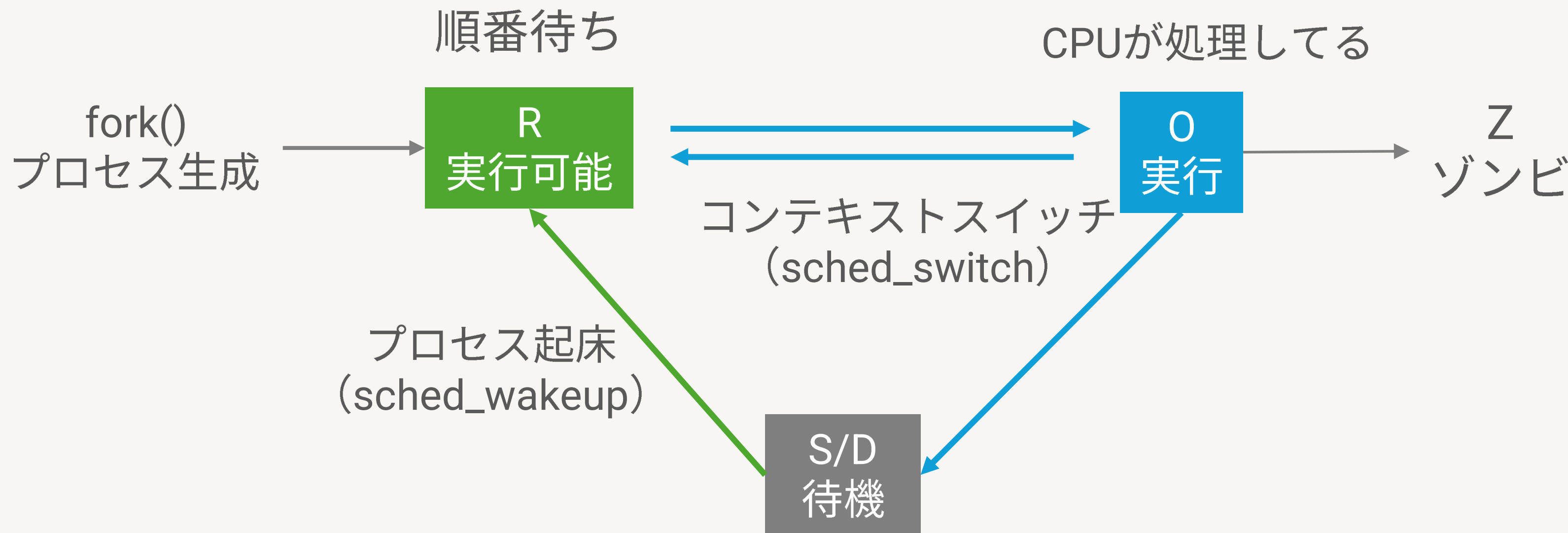
多少重複する部分もありますが、

本日は主に、オンラインセッションのほうで省略していたポイントについて補足をします

Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

スケジュールレイテンシとは

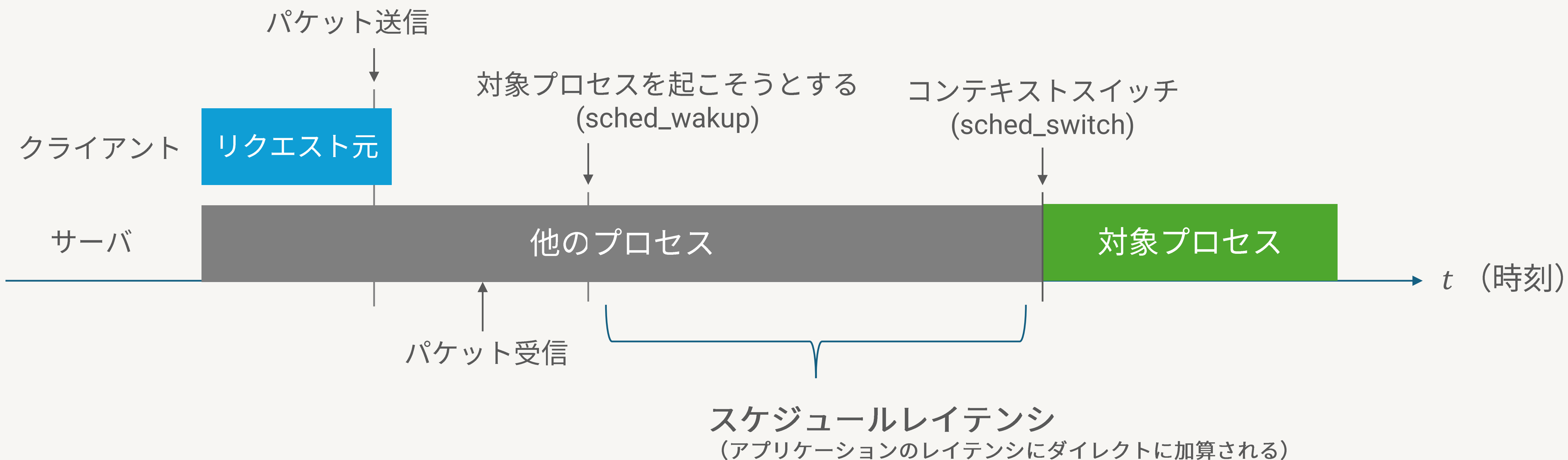
あるプロセスが起床してから実行状態になるまでの遅延時間



- プロセス起床（sched_wakeup）：寝ている（待機中の）プロセスを起こして実行可能状態にする
- コンテキストスイッチ（sched_switch）：CPUが実行中のプロセスを切り替える

Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

アプリケーションレイテンシとの関係



Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

通常のスケジューレイテンシの分析

- perfコマンド
  - `perf sched record` でイベントを記録
  - `perf sched latency` や `perf sched timehist` 等で詳細に分析

- 多少の負荷がかかるため(※) 、本番環境での実行は躊躇

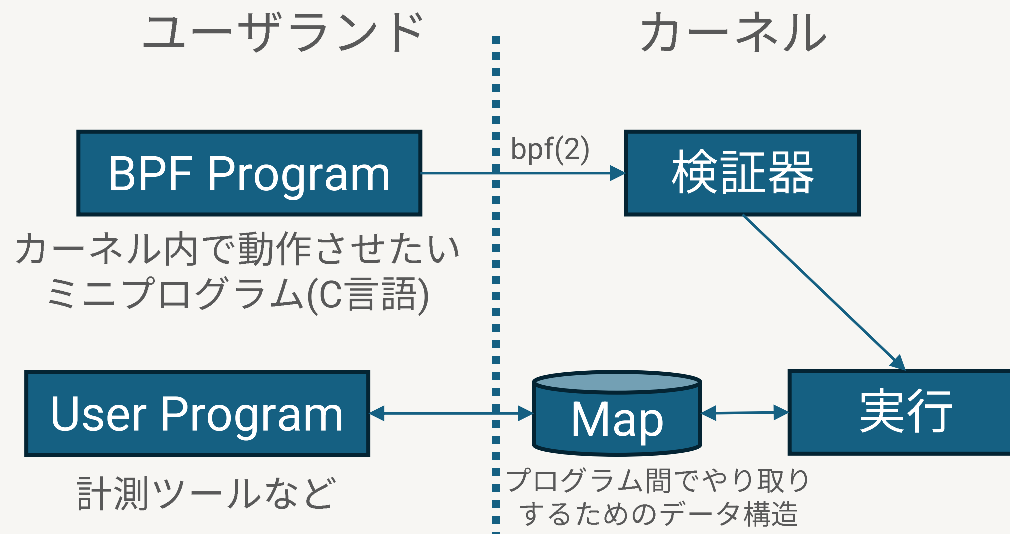
eBPF技術によって、軽量に計測できないか？

※参考： https://www.brendangregg.com/blog/2017-03-16/perf-sched.html

Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

eBPFとは

- Linuxカーネルの拡張性を高める汎用実行エンジン
  - ミニプログラムを作成し、カーネル内で動作させることができる
- イベント駆動型で、様々なイベントに対してフックできる
  - システムコール、カーネル関数、ネットワークパケットなど、様々なイベントをきっかけに動作
- 近年、クラウドネイティブ界隈で注目されている技術の1つ
  - 主にネットワーク、可観測性、セキュリティの領域での応用が多い



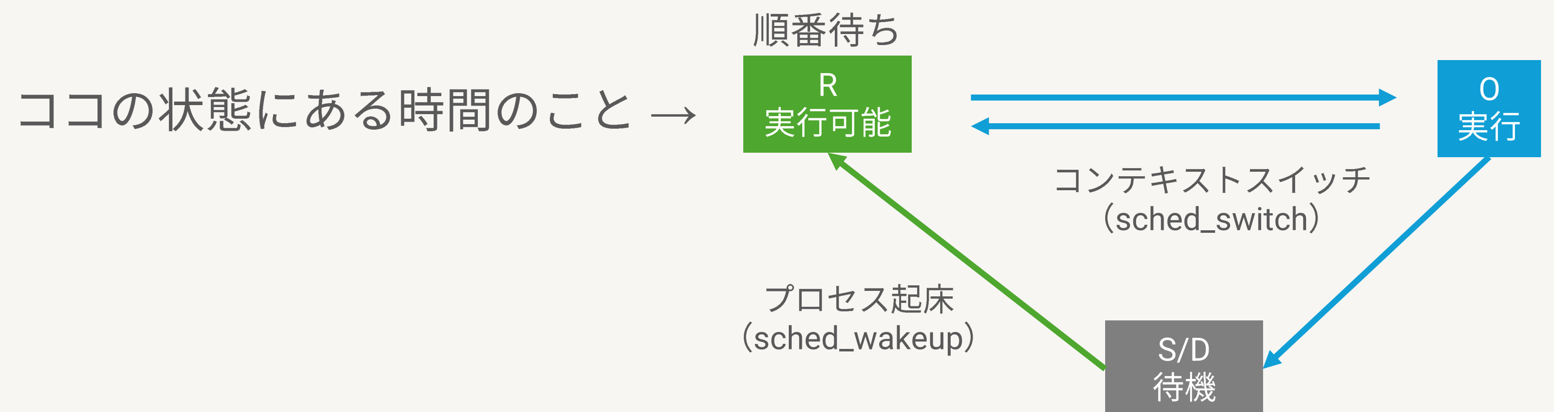
（参考）eBPFの基礎が学べるハンズオン
https://killercoda.com/tamura-daiki-dc3/course/bpf

Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

既存ツールの探索

- BCC: https://github.com/iovisor/bcc
  - カーネル内の様々なイベントを観測するためのコマンド群
  - eBPFで実装されている

- runqslower
  - BCCに含まれるコマンドの1つ
  - 「ランキュー」に滞在する時間が長かったイベントを検知して標準出力



Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

runqslowerの仕組み

①対象プロセスを起こそうとする
(sched_wakup)

②コンテキストスイッチ
（sched/sched_switch）

PID: 100 のプロセス
を起こしたい

PID: 100 のプロセスにス
イッチします

他のプロセス

対象プロセス

t （時刻）

呼ばれる

呼ばれる

eBPFプログラム①

eBPFプログラム②

通知

ユーザランドプログラム

PIDをkeyにして時刻を保存

PID:100が起きた時刻を参照

Key=100, value=15:04:01.123

PID:100のスケジュールレイテンシは2msでした

Map (Key-Value ストア）

詳しくはオンラインセッションを御覧ください

Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

今回ケースに対する技術的な課題

- 分析のために情報を追加する必要があった
  - CPU番号も合わせて取りたい
  - タイムスタンプの精度を上げたい
  - 目的のイベントのみに絞りたい
  - アプリケーション側のスレッドとの対応を分かりやすくしたい

- これらを解決すべく、runqslowerの追加実装を試みる

Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

今回ケースに対する技術的な課題

- 分析のために情報を追加する必要があった
  - ① CPU番号も合わせて取りたい
  - ② タイムスタンプの精度を上げたい
  - ③ 目的のイベントのみに絞り
  - ④ アプリケーション側のスレ

オンラインセッションでは、
時間の都合上、これだけピックアップ

詳しくはオンラインセッションを御覧ください

- これらを解決すべく、runqslowerに追加実装を試みる

Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

今回ケースに対する技術的な課題

- 分析のために情報を追加する必要があった
  - ① CPU番号も合わせて取りたい
  - ② タイムスタンプの精度を上げたい
  - ③ 目的のイベントのみに絞りたい
  - ④ アプリケーション側のスレッドとの対応を分かりやすくしたい

本日はこちらの紹介

- これらを解決すべく、runqslowerに追加実装を試みる

Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

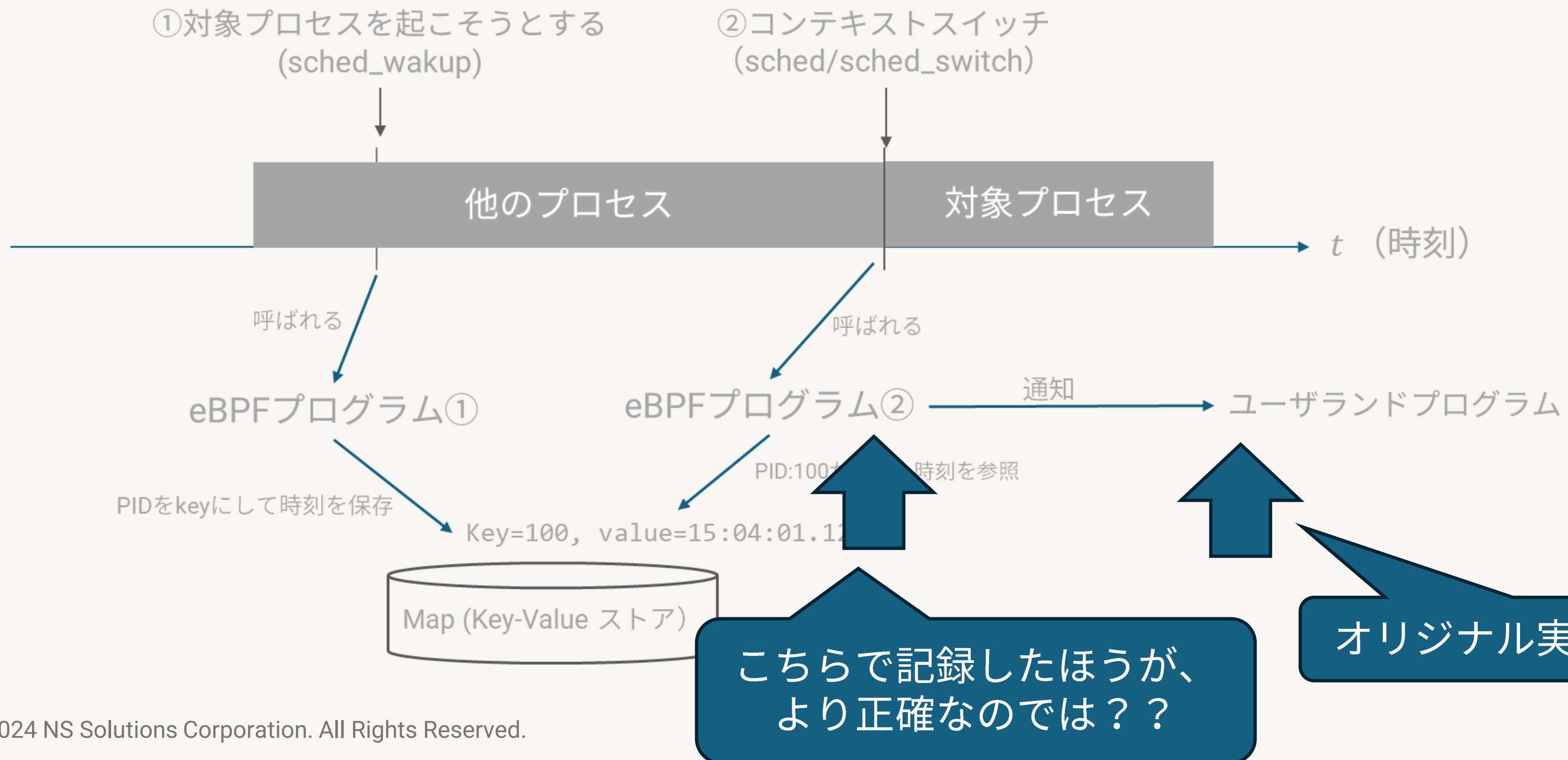
② タイムスタンプの精度を上げたい

- オリジナルは「秒」単位 → マイクロ秒以上にしたい

Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

② タイムスタンプの精度を上げたい

- オリジナルは「秒」単位 → マイクロ秒以上にしたい
- タイムスタンプを打つ場所：



Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

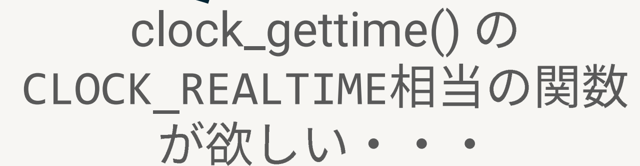
そこまで単純な話ではなかった

汎用性の高いeBPFとはいえ、カーネル内の情報を自由に参照できる訳ではない

情報源は、基本的に以下の３つ

- イベントからeBPFプログラムに渡される引数
- 他のeBPFプログラム等がMapに保存したデータ
- ヘルパー関数として用意されたもの

この中から、時刻に関する関数を探すしかない



だがしかし、CLOCK_MONOTONICまたはCLOCK_BOOTIME相当の関数しか用意されていない・・

2つのイベントの時刻差を計算するには問題ないが、
eBPFプログラム内で正確な現在時刻を得るには難しい

Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

結局どうしたか

- タイムスタンプをナノ秒単位で記録するように修正

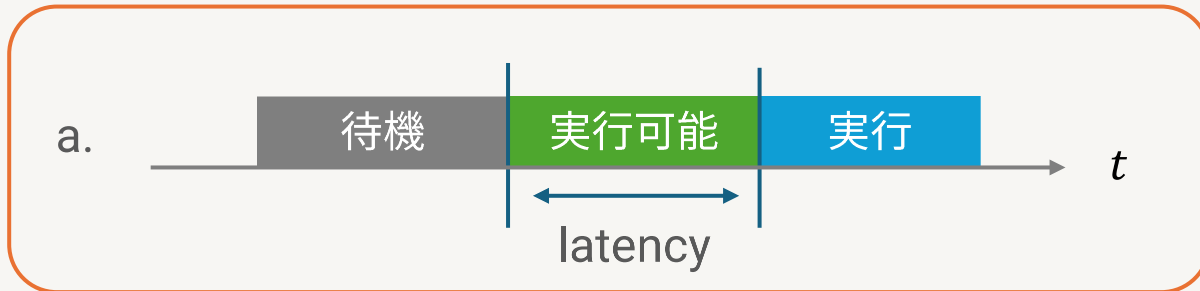
- 記録する場所は、ユーザランド側で受け取った時点で妥協
  - 事前の実験で、約40〜50μ秒程度の遅延がありえることを確認

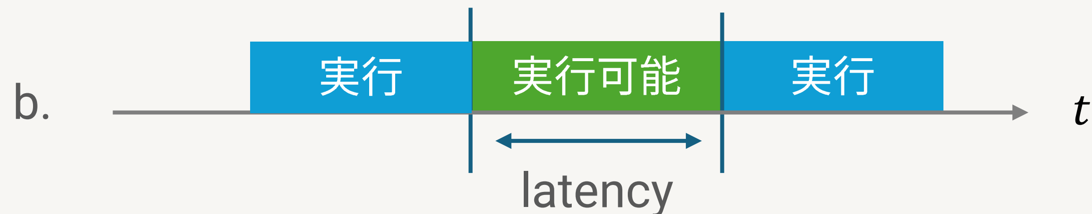
Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

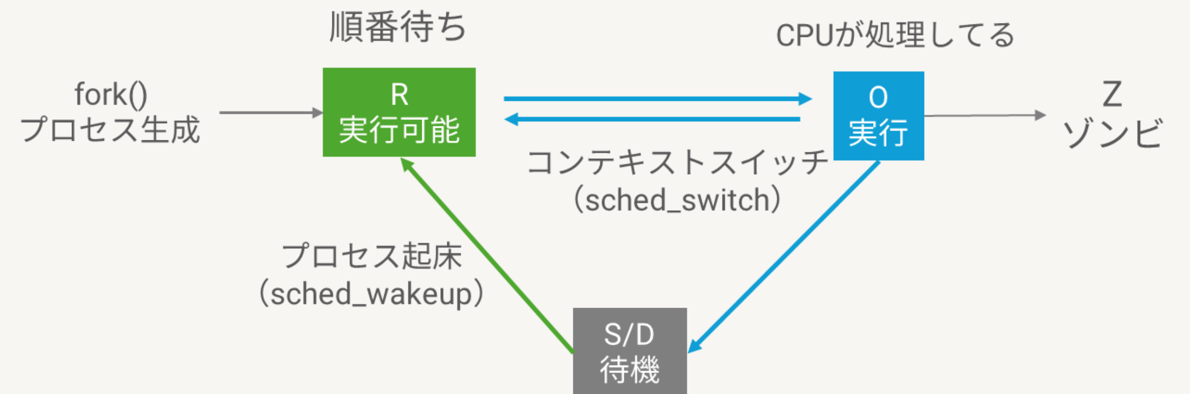
③ 目的のイベントのみに絞りたい

- runqslowerは、ランキューの滞在時間を計測するツール
- ソースコードをよく見ると、2種類のレイテンシを計測していた

→ 今回の計測対象としているほうを残して、削除





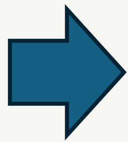


Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

④アプリケーション側のスレッドとの対応を分かりやすくしたい

- カーネル内では、OSスレッドもプロセスも、
  どちらも同じtask_struct構造体で表現される

- 各スレッドを、アプリケーションがどのように使用しているか、
  カーネル側からは分かりにくい
  - コマンド名(comm)はどのスレッドも同じ値
  - スレッドID(pid)くらいしか違いが見えない



今回の対象アプリはJavaだったため、
jstackコマンドを使用し、
スレッドIDとJavaスレッド名をマッピングして出力

Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

今回ケースに対する技術的な課題

- CPU番号も合わせて取りたい → がんばった
- タイムスタンプの精度を上げたい → マイクロ秒で出力
- 目的のイベントのみに絞りたい → コードから除去
- アプリケーション側のスレッドとの対応を分かりやすくしたい
  → スレッドダンプの情報とマッピングして出力

Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

結果

✓ 技術課題をクリアし、runqslowerの改良版を製作完了

✓ 負荷試験にて、アプリに性能影響ないことを確認
✓ デーモン化、ログローテ等、細かな実装も完了

◎ 本番環境での動作も問題なく完了し、データが取れた
非常に緊張した・・その割に、あっけなかった

○ 一部は仮説通り、スケジュールレイテンシが影響してそうに見えるものも見受けられた

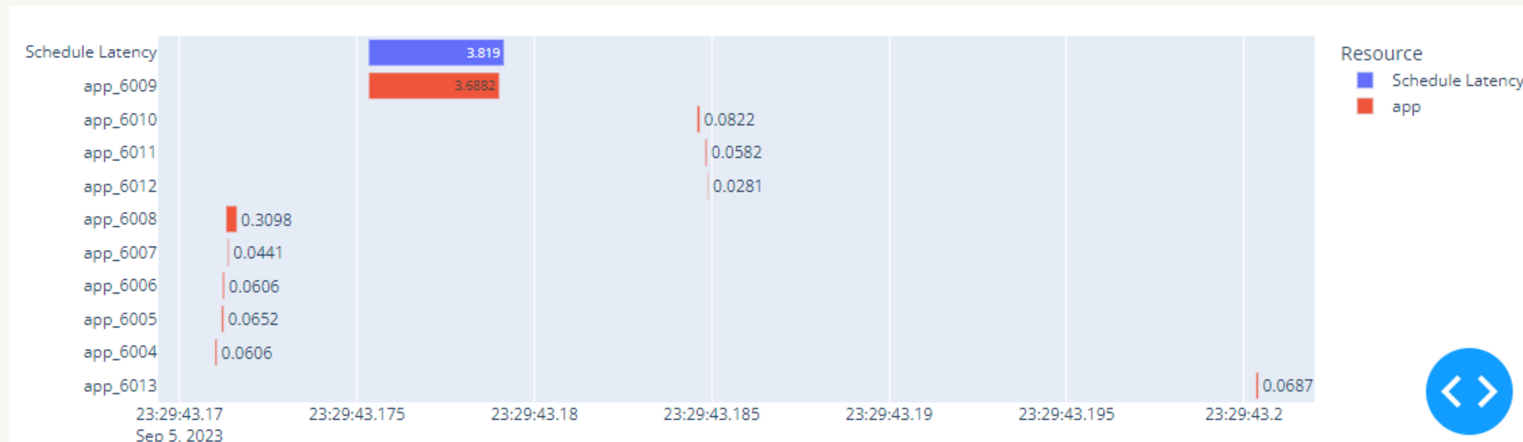
△ 分析が難しく、活用方法の模索が今後の課題

Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

考察：eBPFプログラミングの難しさ

- カーネルに関する深い知識の必要性（オンラインセッションで主に紹介）
  - カーネルコードを読み解く場面も出てくる
  - 現在は生成AIが助けになる

- アプリケーションとカーネルの情報紐づけの難しさ
  - 紐づける手がかりが、タイムスタンプを見比べるしかないのが辛い
  - 可視化・分析するツールを自作しながら頑張った



▲スケジュールレイテンシとアプリケーションレイテンシの対応の可視化イメージ

Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

まとめ

- Linuxカーネルの動きを紐解きながら、システムのObservabilityを高めてみた
  - 今回作成した改良版のコントリビュートも検討したい

- eBPF技術は、Linuxカーネルの拡張性を高めることが可能
  - とはいえ、何でも自由に実行可能な訳でもない
  - アプリケーションイベントの紐づけ方が課題
  - Linuxカーネルの知識が必要となる

- 生成AIやeBPFを活用して、Linuxカーネルと仲良くなろう！

Copyright ©2024 NS Solutions Corporation. All Rights Reserved.

【包括的クラウドネイティブ化支援サービス】CloudHarborのご紹介

- これまでご説明した知見を集約し、包括的クラウドネイティブ化支援サービス CloudHarborとして、事業化を推進しています。
- ご興味のあるお客様は、是非ご連絡ください。

CloudHarbor（クラウドハーバー）提供イメージ

Base　Option

コンサルティング
- 活用戦略アセスメント
- モダナイ戦略アセスメント
- 組織＆プロセス変革
- CNCoE運営支援

伴走型要員支援

Dev　Sec　Ops

クラウドネイティブプラットフォーム

※記載コンポーネントは構成の一例となります

コミュニケーション基盤	クラウドネイティブ開発基盤	クラウドネイティブ稼働基盤
・コラボレーション ・バックログ管理 ・チャット	・リポジトリ ・ビルド ・デプロイ ・解析 ・自動構築 ・統合開発者ポータル	・プロダクション環境 ・ステージング環境 ・オブザーバビリティ ・セキュリティ ・認証認可

教育・育成
- クラウド入門
- クラウド応用
- クラウドネイティブ
- DX・業務効率化

クラウドネイティブガイドライン提供
- 開発ガイドライン
- 活用ガイドライン
- 運用ガイドライン

クラウドリソース提供

powered by aws　Microsoft Azure　Google Cloud　ORACLE Cloud　absonne

クラウドネイティブ統合運用センター

emerald × SRE

- また事業推進に当たり、研究開発・事業推進の両面でキャリア採用募集中です！
- こちらもご連絡お待ちしています！

Copyright ©2024 NS Solutions Corporation. All Rights Reserved.