M-020

Android アプリケーションシステムの加速環境による高速なアプリケーション観察
Accelerated Monitoring Environment for Android Application Systems

福田 翔貴† 濱中 真太郎† 栗原 駿† 小口 正人‡ 山口 実靖†
Shoki Fukuda Shintaro Hamanaka Shun Kurihara Masato Oguchi Saneyasu Yamaguchi

1. はじめに

スマートフォンやタブレット用 OS として高いシェアを持つ Android OS 向けに制作されるアプリケーションは膨大な数に上っており，アプリケーションの動作の観察はセキュリティ上の理由や消費電力の推定などのために重要な事項の一つとなっている．しかし，アプリケーションを実際に動作させる動的解析による動作観察には長い時間を要するという問題点があり[1]，この短縮が重要な課題となっている．

我々は既存研究[2]にて，スタンドアロン型アプリケーションの動作観察を前提とした端末内部時計を加速する手法を提案した．しかし，現在のスマートフォン用アプリケーションにはネットワークを利用しサーバと一体となり運用されているものも多い．本稿では，Android OS に加えてサーバ用 OS として Linux に焦点を当て，動的解析手法を用いつつ短時間で高精度なアプリケーション動作観察を実現するために，サーバ・クライアント双方が認識する時間の流れを実際よりも速くできる環境の構築方法を提案する．

2. 提案手法

本章で，アプリケーションの動作観察にかかる時間を短縮するため，Linux カーネルを改変しシステムが認識する時間の流れを速くする方法を提案する．

Android OS や Linux OS に含まれる Linux kernel のカーネルソースファイル `time/timekeeping.c` 内には，時刻の更新に関する関数 `timekeeping_get_ns` がある．同関数内では `cycle_t` 型の変数 `cycle_now` が宣言され，ここにはクロックソースより取得した値が格納されており，時刻が進むにつれて増加している．本稿の実装ではこの `cycle_now` の増加スピードを 2 倍にすることにより，実時間比 2 倍速の加速環境を実装した．

3. 評価

提案手法をクライアント端末（Android OS）とサーバ計算機（Linux）に導入し，実端末で動作させて評価を行った．詳細な実験環境は表 1 および表 2 の通りである．

3.1 静的動作アプリケーションにおける評価

本節にて，静的な動作のクライアントアプリケーションを用いた評価を示す．動作評価を自作のアプリケーションをクライアントおよびーバに導入して行った．サーバは Web サーバとして振る舞い，コンテンツとその更新情報を提供する．サーバは，コンテンツを 5 分に一度更新する．

これらのアプリケーションを通常システムおよび提案システム上で動作させ，HTTP GET の発行と既存研究にて重

† 工学院大学大学院 工学研究科 電気・電子工学専攻
‡ お茶の水女子大学 理学部 情報科学科

表 1. 使用端末（クライアント）

Device Name	Nexus 7 (2013)
OS	Android 5.0.1 (aosp) with modified kernel
CPU	Qualcomm Snapdragon S4 Pro, 1.5 GHz
Memory	2GB

表 2. 使用端末（サーバ）

Device Name	ASUS X200MA-B
OS	CentOS 6.8 with modified kernel
CPU	Celeron Dual-Core N2830, 2.16GHz
Memory	4GB

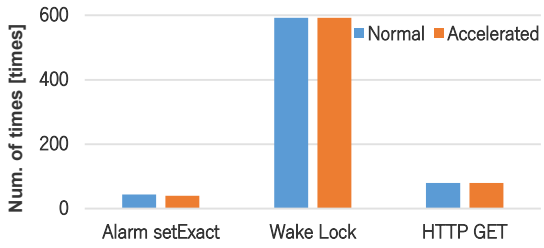

図 1. Alarm, WakeLock および HTTP GET 回数の比較

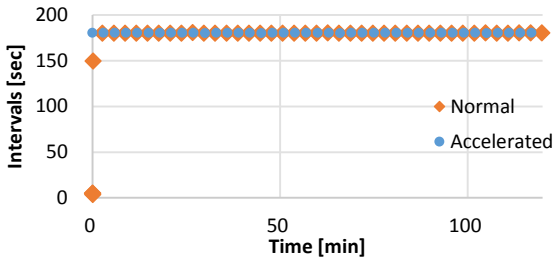

図 2. Alarm 発行間隔の比較

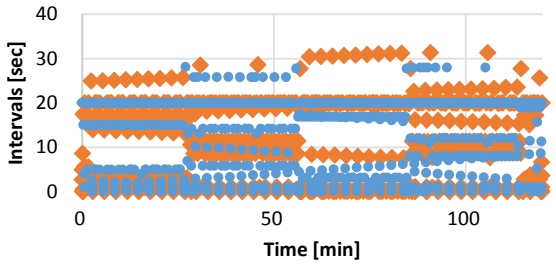

図 3. WakeLock 間隔の比較

要性が示された Alarm セットと WakeLock の時刻を記録した．

加速状態における時計加速倍率は 2 倍とし，実験はスクリーン点灯状態かつ無操作状態で端末内時間にて 2 時間（加速状態では実時間 1 時間）行った．

実験結果を図 1～図 4 に示す．図 1 は Alarm セット，WakeLock 実行，HTTP GET それぞれの回数を通常状態と加速状態で比較した図である．図 2～図 4 では，Alarm セット，WakeLock 実行，HTTP GET それぞれの間隔を表している．

結果より，実時間比 2 倍速の加速状態においても通常状態と同様の結果が得られ，提案手法が有効であることを確認できる．

3.2 動的動作アプリケーションにおける評価

前節と同様に自作のアプリケーションを用いて動作評価を行った．クライアントアプリケーションは新着情報があったときにのみコンテンツ取得を実行する．新着情報が無いときは，最初は n 分に再度新着情報の確認のための接続を行う．n は最初は 1 であり，新着情報が無い場合は倍増させ，最大 16 分まで増加させる．

実験結果を図 5～図 8 に示す．実時間比 2 倍速の加速状態においても，通常状態と同様の結果が得られ，提案手法が有効であることを確認できる．

4. おわりに

本稿では，スマートフォンアプリケーションの動作観察における所要時間が長い問題を紹介した．そして，Linux カーネルの時間管理実装を改変し，Android OS と Linux OS の双方で動作するアプリケーションが認識する時間の流れを実時間より速くし，短い時間でのアプリケーションの動的観察を可能とする手法を提案した．

提案手法を実装しベンチマークアプリケーションと実アプリケーションを用いて評価したところ， Alarm セット，WakeLock 実行，HTTP GET の観察に関して，加速された環境でも通常環境と同等の観察結果を得ることができ，提案手法が有効に機能することが確認された．

今後はより多くのアプリケーションを用いての評価，提案手法の適用可能範囲の調査を行っていく予定である．

謝辞

本研究は JSPS 科研費 25280022, 26730040, 15H02696 の助成を受けたものである．

本研究は，JST，CREST の支援を受けたものである．

参考文献

[1] Shun Kurihara, Shoki Fukuda, Ayano Koyanagi, Ayumu Kubota, Akihiro Nakarai, Masato Oguchi and Saneyasu Yamaguchi: “A Study on Identifying Battery-Draining Android Applications in Screen-Off State”, 2015 IEEE 4th Global Conference on Consumer Electronics (GCCE 2015), 2015.

[2] ShokiFukuda, Shun Kurihara, ShintaroHamanaka, Masato Oguchi and Saneyasu Yamaguchi: “Accelerated Application Monitoring Environment of Android”, 2016 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW 2016), 2016.

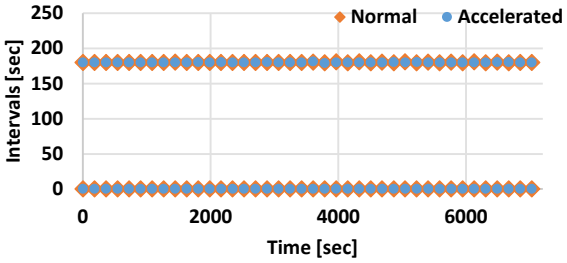

図 4. HTTP GET 間隔の比較

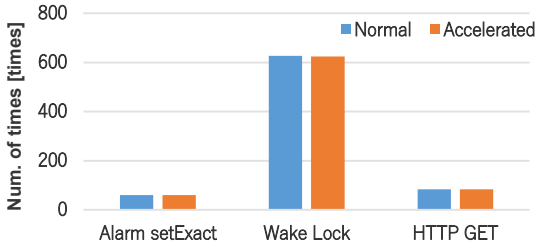

図 5. Alarm，WakeLock および HTTP GET 回数の比較

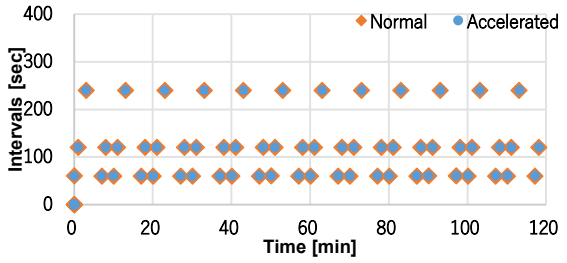

図 6. Alarm セット間隔の比較

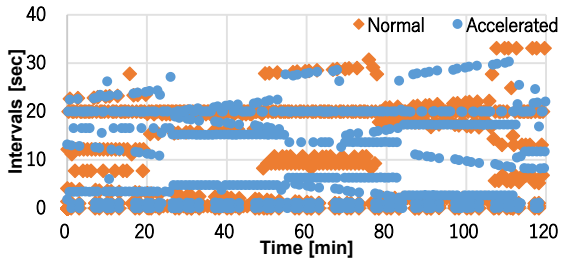

図 7. WakeLock 実行間隔の比較

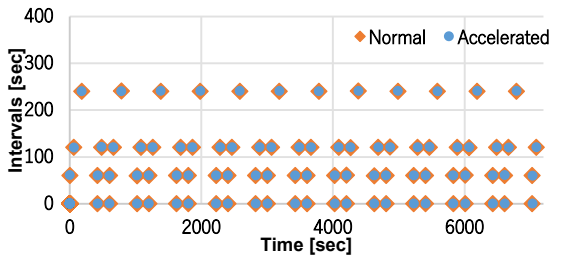

図 8. HTTP GET 間隔の比較