M-019

Android アプリケーションのオブジェクトと GC の関係に関する一考察
A Study on Relation between Application Objects and GC

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

1. はじめに

Android はスマートフォンを始めとしたモバイルデバイスに広く採用されている．Android で動作するアプリケーションは仮想マシン Android Runtime (ART)上で動作し，メモリ管理等は ART が行う．ART のメモリ管理機能の一つに，空きメモリ量が減少するとどこからも参照されていないオブジェクト(ゴミオブジェクト)を発見し，そのメモリを開放する Garbage Collection (GC)がある．GC の処理はアプリケーションスレッドを一時的に止めてしまう Stop The World (STW)を引き起こす．GC 処理はアプリケーション性能やユーザの操作性に悪影響を与える[1]ため，GC の発生は少ないことが望ましい．この停止時間を短縮するため，世代別 GC[2]が提案されている．世代別 GC はオブジェクト群を近い将来にゴミとなると予想されるものと，そうでないものに分け，積極的に前者を探索することで探索範囲を小さくし，停止時間の短縮と GC のスループットの向上を図っている．近い将来ゴミになるか否かの予想は，“多くのオブジェクトは生成されてすぐゴミになる”との経験に基づく仮説に基づき，誕生してからの時間(年齢)が短いオブジェクトは近い将来ゴミになりやすいとみなし，時間が長いオブジェクトはゴミになりづらいとみなしている．探索範囲を効率的に削減するためには，オブジェクトの生存期間を正確に推定することが重要だと考えられる．

本稿では，Android のアプリケーションヒープを解析し，オブジェクト毎のサイズや寿命についての調査を行い，効率的な GC を行うための考察を行う．

2. ART の GC

ART の GC アルゴリズムとして Mark and Sweep (MS)をベースとした Concurrent Mark and Sweep (CMS)が採用されている．MS GC の処理は生きているオブジェクトにマークをつけるマークフェイズと，マークのついていないオブジェクト(ゴミオブジェクト)を回収するスイープフェイズに分けられる．ART の CMS は最初に，マークフェイズで ART が直接参照しているオブジェクト(ルートオブジェクト)にマークをつける．次のマークフェイズでは，ルートオブジェクトからの参照を再帰的に辿り，到達可能なオブジェクト全てにマークをつける．CMS のマークフェイズはアプリケーションスレッドと並行で処理が行われる．よって，マークフェイズ中に参照関係に変更が生じる場合がある．この参照関係の整合性を確保するために，次のリマークフェイズではアプリケーションスレッドを停止(STW)し，変更箇所を再度マークする．その後，スイープフェイズでゴミオブジェクトの回収が行われる．STW となるリマーク処理はマーク時の参照関係の差分のみをチェックするため，マーク処理全体を STW とする MS よりも CMS の方が STW 時間は短くなる．

また，CMS にも世代別 GC の考えが適用されている．CMS では，まず前回の GC 以降に誕生した(年齢が若い)オブジェクトのみを対象とした GC (Sticky GC)を行い，これにより十分なメモリが得られなかった場合は若くないオブジェクトも含む GC (Partial GC)を行う仕組みとなっている．前章の仮説により，Sticky GC では短い時間にて多くのメモリを得られると期待できる．

CMS では，年齢のみに基づき近い将来ゴミになるか否かの予想を行っているが，他の情報によりこの予想の正確さを向上させることができれば GC のスループットを向上させることが可能であると期待できる．

3. Android アプリケーションのオブジェクト特性

本章で一般的な Android アプリケーションのヒープを解析し，オブジェクトの特性と寿命の関係について考察する．

3.1 解析手法

我々は Android OS のソースコードを改変することにより ART GC のモニタリングシステムを開発した．このモニタリングシステムは CMS のマークフェイズにてオブジェクトを再帰的に辿る処理を記録する．マークされたオブジェクトはゴミでは無いことから GC によって回収されなかったオブジェクトが記録からわかる．マーク処理は ART の /art/runtime/gc/collector/mark_sweep-inl.h を修正することによりモニタリングが可能である．

3.2 測定方法

測定端末(Nexus7 2013)にアプリケーションをインストールし，インストールしたアプリケーションを利用し GC の動作を観測した．測定対象アプリケーションは Google Play Store 2016 年 6 月 6 日時点でのカテゴリ毎の無料アプリケーションランキングでトップのものを利用した．測定端末でサポートされていないアプリケーションと測定端末で安定

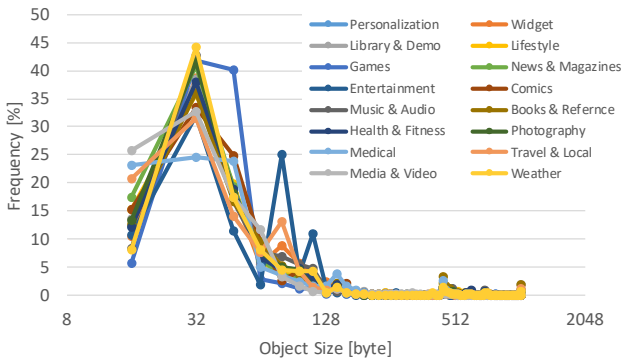

図 1 オブジェクトのサイズ分布

† 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

した動作が確認できないものを除外し，合計 16 アプリケーションのモニタリングを行った．

CMS の GC には Sticky GC と Partial GC があるが，全てのオブジェクトに対して公平に生成と寿命を観察するために，本測定は Partial GC のみが行われるように OS を改変し行った．

3.3 測定結果

図 1 はオブジェクトのサイズ分布を示している．図より，全てのアプリケーションにおいて小さいオブジェクトの数が多く，オブジェクトのサイズ分布はアプリケーションに依らないことが分かる．図 2 と図 3 は，測定対象アプリケーションのうち，GC が 10 回行われた 5 アプリケーションにおけるオブジェクトのサイズとオブジェクトの寿命の関係を示している．縦軸は“経験した GC(マークされ回収されなかった GC)の回数”と示しており，本稿ではこれは寿命と定義する．寿命 5 の場合，オブジェクトは GC が 5 回行われた時点でゴミとならなかったことを示している．

表 1 は 5 アプリケーション全てにおいて平均寿命が 10(最大)であるクラスを示している．Android フレームワーク関連のオブジェクトはアプリケーションに依存せず寿命が長いことがわかる．

4. 考察

2 章で述べたとおり，CMS の寿命の推定は前回の GC 以降に割り当てられたオブジェクトであるか否かにより行われる．Android アプリケーションは共通のフレームワークを用いて動作することから，フレームワーク共通のオブジェクト特性があると考えられ，前章の調査でもアプリケーションによらず長い寿命を持つオブジェクトがあることが確認できた．この特性を用いれば，より正確にオブジェクトの寿命を推定できると期待できる．例えば，寿命が長いと予測されるオブジェクトが生成されたときは最初から Sticky GC の対象外とすれば，効果的に探索領域を削減でき Sticky GC の停止時間を短縮しスループットを向上できると期待できる．

表 1. 寿命が長いオブジェクトのクラス名

android.app.ActivityManagerProxy	android.security.keystore.AndroidKeyStoreProvider
android.app.ActivityThread	android.view.accessibility.AccessibilityManager
android.app.Instrumentation	android.view.accessibility.IAccessibilityManager
android.app.ResourcesManager	android.view.IWindowManager
android.content.pm.IPackageManager	android.view.ViewConfiguration
android.hardware.display.DisplayManagerGlobal	android.view.WindowManagerGlobal
android.hardware.display.IDisplayManager	com.android.internal.logging.AndroidHandler
android.os.AsyncTask	com.android.internal.os.AndroidPrintStream
android.os.Environment	com.android.internal.os.LoggingPrintStream
android.os.ServiceManagerProxy	com.android.internal.os.RuntimeInit
android.security.keystore.AndroidKeyStoreBCWorkaroundProvider	

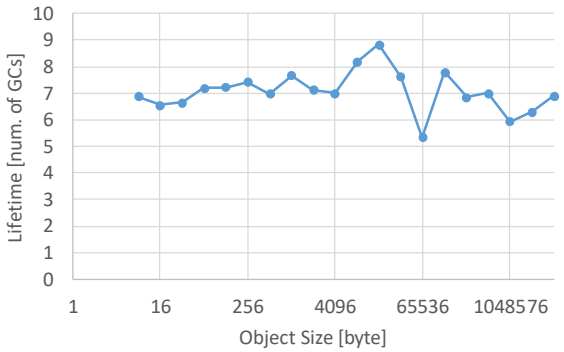

図 2　オブジェクトサイズと寿命の関係(平均)

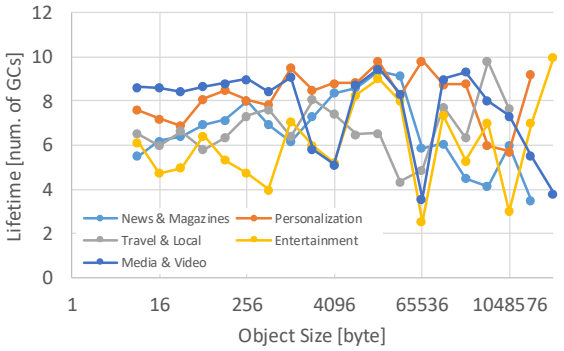

図 3　オブジェクトサイズと寿命の関係

5. まとめ

本稿では，ART のモニタリングシステムを示し，Android アプリケーションのオブジェクトの特性を調査結果と考察を示した．調査により，Android フレームワークの一部のオブジェクトはアプリケーション依らず長い寿命を持つことが判明した．この特性を考慮してオブジェクトの寿命推定を行うことにより，CMS の Sticky GC の探索範囲を小さくし GC パフォーマンスを向上させるこることができると期待できる．

今後は，より多くのアプリケーションを用いての統計値の取得，統計に基づく ART の CMS の実装改変と評価を行う予定である．

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] Stephen M. Blackburn, Perry Cheng, Kathryn S. McKinley, “Myth and realities: the performance impact of garbage collection,” SIGMETRICS '04/Performance '04 Proceedings of the joint international conference on Measurement and modeling of computer systems Pages 25-36

[2] A. W. Appel, “Simple generational garbage collection and fast allocation,” Software—Practice & Experience archive Volume 19 Issue 2, February 1989 Pages 171 - 183