3G-02

Android OS におけるオブジェクト寿命を考慮した GC 範囲の選択に関する一考察

濱中 真太郎[†1]　栗原 駿[†1] 福田 翔貴[†1] 森 竜佑[†2] 小口 正人[†3] 山口 実靖[†2]

工学院大学 電気電子工学専攻[†1] 工学院大学 工学部情報通信工学科[†2]

お茶の水女子大学 理学部情報科学科[†3]

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[1]．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．そのため，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 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効率よく GC を行う．ART においては前回の GC 以降に割り当てられたオブジェクトのみを対象とする Sticky GC と全オブジェクトを対象とする Partial GC という 2 つの GC が存在する．

本稿では，Android のアプリケーションが生成するオブジェクトを動的に解析する ART Monitor を用いることにより，生成されたオブジェクトの情報に基づき Sticky GC と Partial GC の選択制御による GC 効率の向上させる手法について考察を行う．

2. ART の GC

GC はヒープ領域のゴミオブジェクトを発見し，そのメモリを開放するプログラムである．基本的なアルゴリズムとして Mark and Sweep, Reference Count, Copying といったアルゴリズムがあるが，ART の標準の GC は Mark and Sweep がベースの Concurrent Mark and Sweep (CMS)が採用されている．Mark and Sweep は使用中のオブジェクトに印をつけ，印の付いていないオブジェクトをゴミオブジェクトとみなしその領域を解放するアルゴリズムである．ART の CMS は STW 時間の短縮が優先され，アプリケーションスレッド実行中に並列して GC スレッドの処理が行われる．よって，マーク処理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，マーク後にリマーク処理が行われる．リマーク処理はアプリケーションスレッドを停止(STW)し，マークフェイズ中にオブジェクトの参照関係が変更された箇所を再度マークし直す．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，マークフェイズ全体を STW とする Mark and Sweep よりも CMS の方が STW 時間は短くなる．

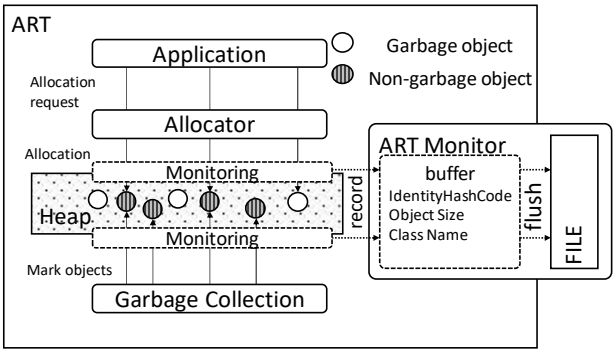

図 1. ART Monitor 概要

また，CMS にも世代別 GC の考えが適用されている．CMS では，前回の GC 以降に割り当てられたオブジェクトを対象とする Sticky GC と全オブジェクトを対象とする Partial GC という 2 つの GC が存在する．

Sticky GC か Partial GC の選択は次のように行っている．1. 初回の GC は必ず Sticky GC を実行，次に Partial GC を実行， 2. Partial GC 実行直後の GC は Sticky GC， 3. Sticky GC 実行直後は前回の Partial GC とスループットの比較を行い，終了した Sticky GC のスループットのほうが高い場合は Sticky GC を実行，そうでない場合は Partial GC を実行しこのスループットを記録する．

3. ART Monitor

我々は Android OS のソースコードの改変することにより ART のオブジェクト情報のモニタリングシステム(ART Monitor)を開発した[2]．概要を図 1 に示す．このモニタリングシステムはオブジェクトの情報と，そのオブジェクトの発生と，マークされた回数を記録することができる．

記録するオブジェクト情報はオブジェクトの IdentityHashCode とオブジェクトのサイズ，クラス名である．IdentityHashCode とは原則，別々のオブジェクトであれば異なる値を示すものである．オブジェクトの発生はアプリケーションからオブジェクトの生成要求が来たときに ART が要求を受け取りアロケータに従いオブジェクトを配置する処理を記録する．マークされた回数は CMS のマーク・リマークでオブジェクトにマークする処理を時系列に従い記録する． これらの情報からオブジェクトの寿命を算出できる．

A Study on Selection of GC Based on Object Lifetime in Android
†1 Shintaro Hamanaka, Shun Kurihara, Shoki Fukuda, Electrical Engineering and Electronics, Kogakuin University Graduate School
†2 Ryusuke Mori, Information and Communication Technology, Kogakuin University
†3 Masato Oguchi, Department of Information Sciences, Ochanomizu University

4. オブジェクト寿命と GC 性能調査

Sticky GC は前回の GC 以降に生成したオブジェクトの多くがゴミである場合に有効である．そうでない場合に Sticky GC を実行してしまうとゴミでないオブジェクトに対して判定を行うことになり性能が低下すると考えられる．よって，GC の選択は生成されたオブジェクトの寿命に基づき行われることが好ましいと考えられる．しかし，2 章で述べたように Sticky GC と Partial GC の選択はスループットにより行われる．

そこで，生成するオブジェクトの寿命と GC 性能の関係についてこの選択の有効性について調査を行った．測定に利用した端末は Nexus7 (2013)，CPU Qualcomm Snapdragon S4 Pro 1.5GHz，メモリ 2GB，OS Android 6.0.1 で，ベンチマークアプリケーションは自作の物を利用した．ベンチマークアプリケーションは for 文 i(10)*j(100,000)のループ内で寿命が 0 か 1 のオブジェクトを 1 つずつ，合計 1,000,000 個のオブジェクトを生成するアプリケーションである．本測定における寿命の定義は for 文の i ループを寿命の区切りとする．よって，寿命 1 のオブジェクトは同じ i ループ内ではゴミにならない．測定は寿命 0 のオブジェクトの出現比率を変化させて行った．寿命 0 と 1 のオブジェクトはそれぞれ別のクラスのオブジェクトである．また，比較のため全ての GC が Partial GC で実行されるよう改変した場合でも測定を行った．

測定結果を図 2 に示す．図の横軸は寿命 0 オブジェクトの出現比率で，値が 0 の場合は寿命が 1 のオブジェクトのみが出現し，値が 1 の場合は寿命が 0 のオブジェクトのみが出現することを示している．図の縦軸は平均 STW 時間を示している．図より寿命 0 オブジェクト比が 0.5 以下の場合は Partial GC のみを実行する手法，それ以外は通常手法の STW 時間が短いことが分かる．

5. 提案手法

前章の調査結果から，寿命 0 のオブジェクトが小数の場合は Partial GC を実行したほうが良いことが分かった．そこで寿命 0 オブジェクト比が 0.5 以下のときに強制的に Partial GC を，そうでない場合は通常手法の判定に従い Sticky GC か Partial GC の決定を行う手法を提案する．ART Monitor を導入し，前回の GC 以降に生成されたオブジェクトの数とそのうちの寿命 0 オブジェクトの数から寿命 0 オブジェクトを算出し，寿命 0 オブジェクト比が 0.5 以下の場合は GC 実行時に Partial GC が実行されるように改変を行った．

6. 性能評価

提案手法の性能評価を行う．測定環境は 4 章の測定と同様である．本測定ではクラス名よりオブジェクト寿命を推定可能であることを前提としている．

測定結果を図 3，図 4 に示す．図より，寿命 0 オブジェクト比 0.5 以下では Partial GC を実行し，それ以外の場合には通常手法の判定を行うことにより寿命 0 オブジェクトが少数の状況にて通常手法より優れた性能（短い STW 時間）を，多数の状況にて通常手法と同等の性能を実現できたと言える．

オブジェクトの寿命の推定は，文献[2] で示されている観察に基づく寿命推定により実現できると考えられる．

7. おわりに

本稿では，Android のアプリケーション実行環境である ART の GC において，アプリケーションが生成するオブジェクトの寿命と GC 性能について調査を行った．そして，寿命 0 のオブジェクトが少数の場合に通常手法の STW 時間は増加し性能が大きく下がることを確認した．また，生成されたオブジェクトをモニタリングする ART Monitor を導入し，オブジェクトの生成状況から GC を制御することにより性能低下を低減できることを確認した．

今後はオブジェクト寿命のプロファイリングや実アプリケーションへの適用を行っていく予定である．

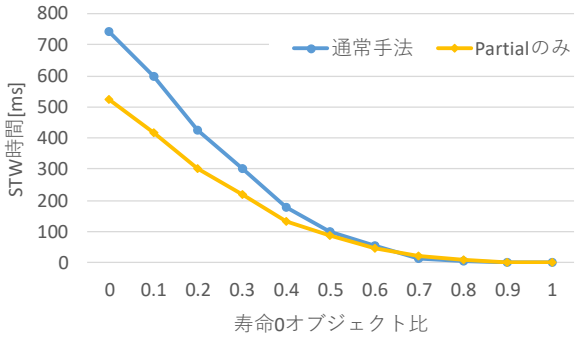

図 2. STW 時間(通常手法と Partial のみの比較)

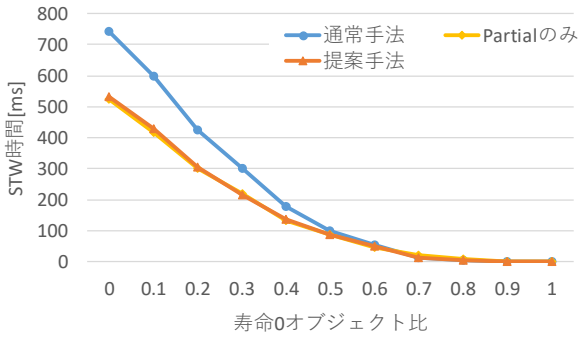

図 3. STW 時間(通常手法と提案手法の比較)

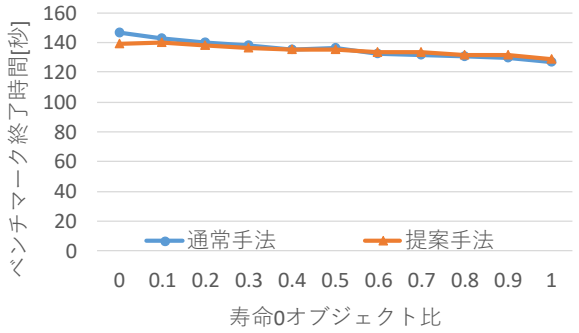

図 4. ベンチマーク終了時間

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] 永田恭輔，中村優太，野村 駿，山口実靖, "Dalvik VM コンカレント GC の STW 時間の短縮に関する考察", 情報処理学会 第 76 回全国大会，6J-2

[2] Shintaro Hamanaka, Shun Kurihara, Shoki Fukuda, Ryusuke Mori, “Object Lifetime Trend of Modern Android Applications for GC Performance Improvement”, ACM IMCOM 2017