B-005

Dalvik VM GC の性能の評価

A Performance Evaluation of Dalvik VM GC

永田 恭輔† 中村 優太† 野村 駿† 山口 実靖†
Kyosuke Nagata Yuta Nakamura Shun Nomura Saneyasu Yamaguchi

1. はじめに

スマートフォンやタブレット PC が普及し、これらの端末の重要性が高まっている。Android はこれらの端末のプラットフォームとして高いシェアを持ち、特に重要なプラットフォームとなっている。Android には Dalvik VM という仮想機械が搭載されており、Android のアプリケーションは Dalvik VM の上で動作する。Dalvik VM には GC 機能が搭載されており、この性能がアプリケーション性能に影響を与える。本稿では、Dalvik VM の GC の性能について考察する。

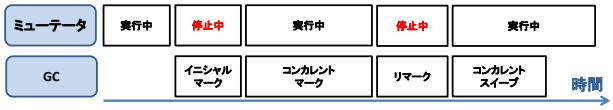
2. Dalvik VM の GC

Dalvik VM は GC アルゴリズムとしてマーク&スイープを採用している。マーク&スイープのメリットとして、実装が容易であること、参照カウントと異なり GC が動いてないときは高速であることや循環参照も解放できるということが挙げられる。一方デメリットとしては、全てのオブジェクトを辿らなければならないため時間がかかるという点が挙げられる。また Dalvik VM の GC では、CoW にてプロセス間で共有されているメモリにマーキングによる書き込みが生じないように、ビットマップマーキング方式が採用されている[1]。

Android 2.3 以降のバージョンでは、コンカレント GC が採用されている。コンカレント GC は GC 処理とミューテータ（アプリケーション）処理を図 1 の様に並列処理する GC である。コンカレント GC の処理は以下の通りである。まず、イニシャルマークとしてルートオブジェクトに印をつける。この間は、ミューテータは停止（ストップザワールド）する。次に、コンカレントマークとしてルートから辿れるオブジェクトに印をつけていく。この間は、ミューテータと GC が並列に動作し、ミューテータは停止しない。次に、リマークとしてコンカレントマーク中に生じた書き込みの整合性をとるためにミューテータを停止（ストップザワールド）させてマーキングを行う。最後に、コンカレントスイープとして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、フリーリストに繋ぎ、再利用可能な領域とする。コンカレントスイープはミューテータと並列に処理が行われる。非コンカレント GC では全てのマーク処理中ミューテータが停止（ストップザワールド）するのに対し、コンカレント GC では 2 度の停止（ストップザワールド）の間ミューテータが動作するため、停止（ストップザワールド）時間が大幅に削減されている。

† 工学院大学大学院 工学研究科 電気・電子工学専攻
Electrical Engineering and Electronics, Kogakuin University Graduate School

図 1 コンカレント GC の処理



3. 性能評価

本章では、様々な環境でのコンカレント GC の性能評価を行う。

3.1. 測定環境

Nexus7 上で自作ベンチマークを実行し測定を行った。端末の詳細は表 1 の通りである。

表 1 測定環境

OS	Android 4.2.2
CPU	NVIDIA Tegra 3 T30L 1.3GHz クアッドコア
メモリ	1GB
ストレージ	16GB

3.2. ベンチマーク

本節では、ベンチマークについて説明する。ベンチマークは初めに、指定数のオブジェクト（これは非ゴミオブジェクトとなる）を生成する。次に、ゴミオブジェクトを生成し続けることで、GC を起動させる。非ゴミオブジェクト（ルートオブジェクトから辿れるオブジェクト）数やオブジェクト同士のリンクの書き換え数を変化させ、GC の処理にどのような影響を与えるかの評価を行う。

測定では GC 時間とミューテータの性能を取得しており、GC 時間は GC のそれぞれのフェーズに要する時間であり、性能は単位時間当たりの for 文のループ回数である。for 文内ではオブジェクトのリンク書き換えとゴミオブジェクト生成を行っている。

3.3. 非ゴミオブジェクト数と GC 性能の関係

非ゴミオブジェクト数の変化による、GC のそれぞれのフェーズの時間の変化を図 2, 3, 4 に、ミューテータの性能の変化を図 5 に示す。図内の（有）、（無）はリンク書き換えの有無を表している。図 2, 3, 4 から、非ゴミオブジェクト数が 10^3 以下ではあまり変化はないが、それより多くなると非ゴミオブジェクト数が多い程 GC 時間が増加することが分かる。図 5 の結果から、マーク中は非 GC 中よりもミューテータの性能が低下することが分かる。リンク書き換えの有無を比較すると、GC 中にリンクの書き換えを行うと、オブジェクトが多くなるにつれてリマーク時間のみ大きくなるということが分かる。

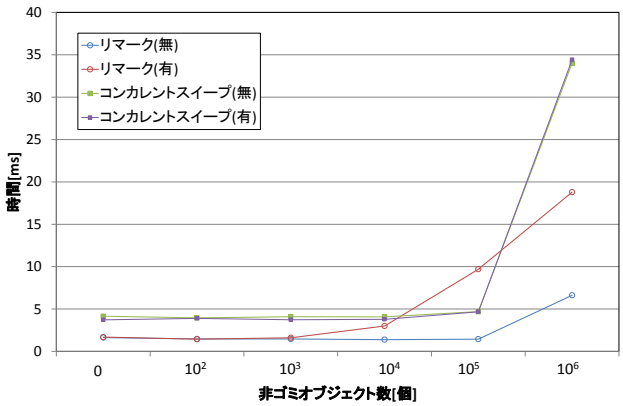


図 2　非ゴミオブジェクト数と GC 時間(1)

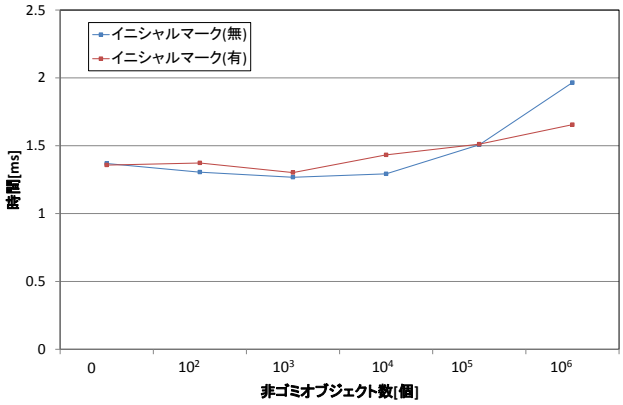


図 3　非ゴミオブジェクト数と GC 時間(2)

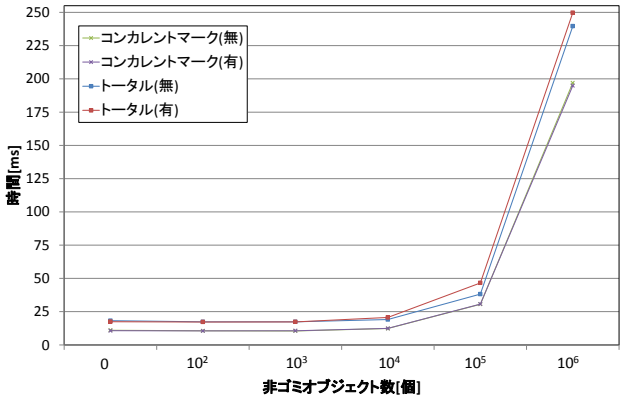


図 4　非ゴミオブジェクト数と GC 時間(3)

3.4. リンク書き換え頻度とリマーク時間の関係

前節の結果から、GC 中にリンクの書き換えが行われると、リマーク時間が大きくなることが分かった。前節のリンク書き換えはゴミオブジェクト 1 個生成するたびに既存の非ゴミオブジェクト同士でランダムに張ったリンクを別のランダムに選択したオブジェクトに対して張り替えている。これらの非ゴミオブジェクトの全てはルートオブジェクトからリンクされており、非ゴミオブジェクト同士のリンクを失ってもゴミオブジェクトとなることはない。本節では、このリンク書き換え頻度をゴミオブジェクト 1 個生成のたびに、1 回から 5 回に変えてリマーク時間を測定した。結果を図 6 に示す。書き換え頻度を増加させるにしたがって、リマーク時間も増加することが確認された。これは、コンカレントマーク中に変更されたオブジェクトが多く、リマークで調査しなおすオブジェクト数が増えたためだと考えられる。

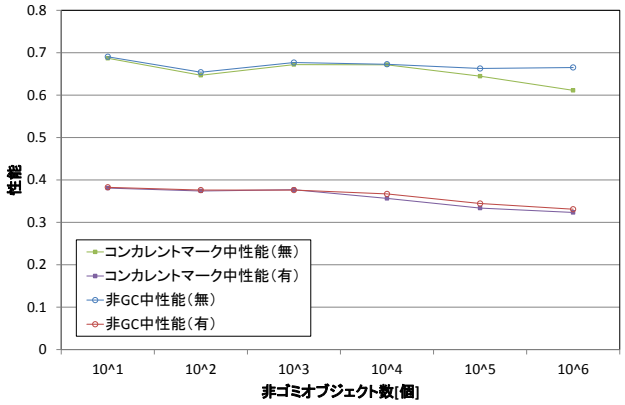


図 5　非ゴミオブジェクト数とミューテータ性能

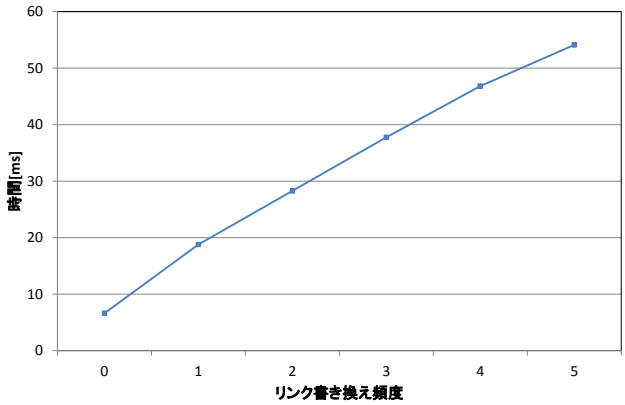


図 6　書き換え頻度とリマーク時間

4. まとめ

本稿では、オブジェクト数の変化や、オブジェクトの書き換え頻度の変化による GC 時間やミューテータの性能への影響を評価した。結果として、これらにより GC に要する時間が大きく変わることが確認された。今後はコンカレントマーク中のミューテータの性能について考察していく予定である。

謝辞

本研究は JSPS 科研費 24300034, 25280022 の助成を受けたものである。

参考文献

[1] TOMOHARU UGAWA, HIDEYA IWASAKI, TAIICHI YUASA, “Improvements of Recovery from Marking Stack Overflow in Mark Sweep Garbage Collection”, IPSJ, Vol.5, No.1 (2012).