リアルタイム環境に適した Garbage Collection API の研究

八十島 利典

北陸先端科学技術大学院大学 情報科学研究科

平成 12 年 2月 15 日

キーワード: ガベージコレクション, リアルタイム, Java, API, Kaffe OpenVM.

1 背景

近年、プログラミング言語 Java は広く利用されている。なぜなら、分散環境におけるセキュリティの特徴、コードの再利用性、他のプラットフォームへの移植性のような多くのソフトウェア工学の利点を持つからである。よって、リアルタイムシステムや組み込みシステムでさえ、Java 言語でコードを書くことを望む。

しかし、Java 実行環境と言語仕様は、リアルタイム環境をサポートするのに十分ではない。ひとつの理由は、Java のガベージコレクタである。Java のガベージコレクタは、手動のメモリ管理からプログラマを助ける。ガベージコレクタが自動的にメモリを管理するので、プログラマはメモリ管理について心配する必要がない。ゆえに、ガベージコレクタは Java のコードの保守性を大幅に高める。

その一方で、ガベージコレクタは最大応答時間とヒープサイズの残量の保証、そしてガベージコレクタ等の全体の処理時間の見積りや予測が難しいことから、リアルタイムシステムで Java を使用するとき、ガベージコレクタは厄介となる。

リアルタイムシステムへ Java を適用するとき、我々は以下の要求を考慮しなければならない。

- リアルタイム要求を満たすこと
- メモリを十分に供給すること
- 移植性、再利用性そして保守性のようなソフトウェア工学の利点を保護すること

しかしながら、上記の要求はガベージコレクションの一般的な手法では満たされない。また、ガベージコレクションの研究は多く行われており、ゆえにアルゴリズムの改良だけでガベージコレクションの性能向上は非常に難しい。

しかしながら、ガベージコレクションのパラメータ変更による性能向上の研究は、あまり行われていない。ゆえに、パラメータ化されたガベージコレクションのパラメータを変更することによって環境やパラメータに適応することが可能であるから、パラメータ化したガベージコレクションは性能の向上が期待できる。しかしながら、２つの問題がある。

- ソフトウェアの保守性はガベージコレクションを用いることで保たれるが、リアルタイムシステムの要求を満たすことは難しい
- ガベージコレクションはリアルタイム拡張においてリアルタイム制約を満たしにくくすることからチューニングが必要である

リアルタイムシステムに対する多くのガベージコレクションアルゴリズムは既に提案されているが、十分ではない。これは、GC の性能がメモリ使用のパターンやオブジェクトの寿命に強く依存するからであるが、ほとんど全てのアルゴリズムはこの問題について関心を持たない。我々は、GC 自身をカスタマイズする方法をユーザに全く供給していないことに注目した。我々のアイデアは、カスタマイズ可能な GC を供給することである。例えば、GC の幾つかのパラメータをユーザに許すことである。しかしながら、GC のパラメータ変更による性能向上は、十分に研究されていない。ゆえに、我々は、GC のパラメータ変更によって基礎となる環境やアプリケーションに GC を適応することで大きな性能向上を予想する。

2 目的

本論文で、リアルタイム環境に拡張した Java のガベージコレクション API(略して GC API) を提案し、実装する。GC API は、ユーザに GC のパラメータや振る舞いの変更を許すことから、リアルタイム環境に適応することが可能である。GC API は以下の３つの関数からなる。

- GC の実行間隔や優先順位の指定

  一定間隔で起動される GC(Timed-GC と呼ばれる) は既に提案されている。我々は Timed-GC をベースとした GC を用いて、実行間隔を変更するための API を実装する。実行間隔が短ければより多くのメモリを回収するが、overhead も大きくなる。その逆も同様である。実際、適切な間隔を指定することによって性能を向上することができる。

- GC の起動抑制・許可

ライトバリアの overhead でさえ許可されないクリティカルセクションが存在する。そのような場合、GC が起動されないことを保証することが重要である。この API は、この役割を供給する。

- GC が管理しないメモリの予約

  GC 経由のヒープメモリの allocate/deallocate は特別なコストがかかることより、リアルタイム制約が速い allocate/deallocate を要求する場合

  GC の起動が抑制されていることから、GC の管理下にないメモリが要求された場合

  メモリのいくらかの総量が保証されるべき場合

この API は上記の要求を満たすが、プログラマは再び手動のメモリ管理に悩まされ、幾つかの条件を違反しないように注意しなければならない。例えば、手動で割り当てたメモリ上のオブジェクトは、GC メモリ上のオブジェクトの参照を含むことはできない。

GC API を使用するならば、プログラマは GC のパラメータを各々のアプリケーションに依存して変更することができる。適切なパラメータに設定されるならば、GC の割り込みによるアプリケーションの停止時間の減少が期待できる。

3 実装の概要

本研究で、我々はリアルタイム環境のために改良された Java VM である Kaffe-1.0.b1-{rt} を使用した。我々は GC の割り込みによる停止時間を見積もることができないことから、kaffe で使用する GC は non-incremental な mark-sweep アルゴリズムであるので、このアルゴリズムはリアルタイム環境には向いていない。ゆえに、我々は時間によるトリガとプログラマによっていつでも停止させることができる Timed-GC を元にした GC を実装する。

GC を実装するにあたり以下の変更を行った。

- GC の実行間隔や優先度の指定

  GC の実行時間を変更するためにタイマスレッドを使用する。

  獲得したスレッド ID を使用して GC の優先度を変更する。スレッド ID は `kaffe_thread_interface` を用いて獲得した。

- GC の起動抑制・許可

  GC スレッドを停止させることによって GC の起動を抑制するように実装した。逆に、GC スレッドを再開させることで GC の起動を許可するように実装した。

- GC の管理下にないメモリの予約

この機能はまだ実装されておらず、デザインだけである。

4 結論

我々はリアルタイム要求を満たすため GC のパラメータや振る舞いの変更を許す GC API を提案した。実験的な実装は、不幸なことながらまだ完了していないにも関わらず実行可能性を示した。GC API の有効性を示すため、実装を完了し、GC の性能が GC API を用いることで向上される多くの例を示す必要がある。