組み込み機器のためのメモリ管理における断片化の改善について

高橋毅 (910060)

北陸先端科学技術大学院大学 情報科学研究科

2002 年 2 月 15 日

キーワード: 組み込み機器、メモリ管理、断片化、Viewer、アルゴリズムの理解 .

背景

近年の半導体技術の進歩に伴い、コンピュータのメモリ容量は飛躍的に大きくなった。組み込み機器の分野であっても、大規模化、複雑化が急激に進み、例えば、現在の携帯電話や PDA などの携帯端末は、10 年前の標準的なコンピュータに匹敵するほどの性能をもち、今や小型 Java VM までもが搭載されるようになった。しかし、そうはいってもメモリは有限の資源である。メモリの断片化の問題が未だに重大な問題として扱われていることからも分かるように、メモリを効率的に使用することは難しい。

断片化の解消を目的とした研究は多く、断片化の測定方法もいくつかある。しかし、これらの研究や測定方法は十分なものではない。なぜなら、メモリ管理方法では必ずトレードオフの関係をもつ。例えば、速度とメモリの関係がある。また、測定方法はアプリケーションや環境によって重視される問題が違う。従って、組み込み機器での断片化の問題を、これらの研究や測定方法に適用することは難しい。なぜなら、組み込み機器とデスクトップ分野では使用される用途が違うからである。例えば、デスクトップ分野を対象にした断片化の測定方法は、いかにメモリを消費せずに、アプリケーションの実行ができるかということを目的にしている場合が多い。一方、組み込み機器は、長時間の動作に耐えなければならない機器も多く、これらを同じ測定方法で測定することはできない。しかし、組み込み機器を対象とした断片化の解消を目的とした研究や、断片化の測定方法は見当たらない。

一方で、例えば、KVM のメモリ管理部分の、ガベージコレクタ (GC) は単純な mark-sweep GC で、コンパクションしないため断片化の発生する可能性があること。また、非インクメンタル GC であるため、GC の動作中はユーザプログラムが中断するという問題をもつ。このことから、組み込み機器でも断片化が問題視されていることが分かる。

目的

本研究では、組み込み機器のための、断片化の可能性を軽減する割り当て方法の提案を目的としている。その際、各環境で最も適した割り当て方法とそのパラメータを導くためのテスト環境の構築もおこなう。組み込み機器は長時間の動作に耐えなければならない機器もあるので、本研究では断片化の可能性を減らし、長時間の動作に耐えるメモリ割り当て方法を目指す。

断片化の発生を抑えるための方法は、コンパクションの導入と、割り当て方法の改良の 2 通りが考えられる。しかし、本研究では以下の理由によりコンパクションを導入せず、割り当て方法の改良するアプローチを選択した。

- 単純なコンパクションの導入は、GC 実行中にユーザプログラムの停止時間を長くする。また、コンパクションの処理を分散させた場合、全体の処理が重くなり、リソースの限られた組み込み機器には向かない場合がある。例えば、リアルタイム性を重視した組み込み機器には向かない。
- KVM は C 言語で実装されているため、ポインタの識別が完全ではない。よって、オブジェクトの移動には複雑な処理が必要である。
- 断片化の発生を抑えるアロケータを使って、コンパクションを導入すれば、コンパクションの回数を減らすことができる。
- 組み込み機器では、単一のアプリケーションのみが動作する場合がある。この場合、環境に特化したチューニングを施すことによって、良い性能をだす場合があると考えられる。例えば、Linux に導入されているスラブアロケータがある。

次に、本研究で必要だと考えるテスト環境の機能を以下に示す。

- **メモリ内部をグラフィカルに表示すること**
  満足のいく断片化の測定方法はない。メモリ内部の状態 (空き領域、使用領域、内部断片化) の状態を視覚的にみることで、断片化の状態を人間が直接的に把握して、パラメータや割り当て方法の変更ができる。
- **パラメータの変更が容易であること**
  割り当て方法の性能は、アプリケーションごとに違うため、理論的に適した割り当て方法を導くことは難しい。しかし、多くの実験をすることで、結果的に性能の高い割り当て方法を見つけることも 1 つの手段である。できるだけ多くの実験をするためには、割り当て方法やパラメータの変更を迅速におこなう必要がある。

提案する割り当て方法

断片化の可能性を減少させる割り当て方法として、本研究では以下の 2 つの方法を提案する。

- Block buddy system
- Separate first fit 法

Block buddy system は例えば、8KB のオブジェクトを 10 個、16KB のブロックを 10 個というようにオブジェクトの大きさごとに個数を限定し、そのオブジェクトの集合を Buddy system で扱う割り当て方法である。

この方法には利点はブロックに割り当てるオブジェクトの個数を固定することで、比較的大きなオブジェクトに対しても一定量以上確保することができることである。なぜなら、小さなオブジェクトが割り当てられない場合は、特例でそれよりも大きな領域に割り当てることもできるのに対し、大きなオブジェクトが割り当てられない場合は、他の領域に割り当てることができない場合が多いからである。

Separate fit 法は、全体の記憶領域をいくつかの領域に分割し、各領域に対して割り当てるオブジェクトの大きさの範囲を決めて、first fit 法で割り当てる方法である。一般的な first fit 法と比べ、割り当てるオブジェクトの大きさを限定することで、1 つの領域が扱うオブジェクトの種類が少なくなる。よって、記憶領域の利用効率が向上する。分割する領域の大きさを誤ると利用効率は低下する。

断片化の測定方法

一般的な断片化の測定方法を以下に示す。

$$\frac{\text{実際に割り当てたオブジェクトの総量} - \text{要求されたオブジェクトの総量}}{\text{実際に割り当てたオブジェクトの総量}}$$

この測定方法は、使用するヒープ領域が小さければ小さいほど良い性能をだす。よって、内部断片化を発生させると性能が下がる。また、突発的に大きなオブジェクトを割り当てた場合、性能が下がりやすい。しかし、本研究では、長時間の動作に耐えるという視点で断片化を測定するためこの測定方法では不十分である。本研究で提案する測定方法を以下に示す。

$$\frac{\text{空き領域全体} - \text{割り当てられたオブジェクトの総量}}{\text{空き領域全体}}$$

を測定し、横軸を時間軸、縦軸を上記の計算結果としたグラフを作成する。このグラフが、右肩上がりの場合は断片化を発生していると判断し、横軸に平行であるほど再利用性が高いと判断する。つまり長時間の動作に耐られると評価する。

実験

断片化の状態を計測するために本研究では以下の 2 つの実験方法が考えられる。

- 実環境での実験
- 乱数を用いた実験

実環境で実験をすると、実行環境やアプリケーションに適した割り当て方法とパラメータを得ることができる。しかし、割り当て方法の変更が複雑であり、割り当て方法やパラメータを変更するたびにコンパイルし直さなければならないため、時間がかかる。一方、乱数を用いた実験は、導出した割り当て方法とパラメータは平均的な性能を表したものになるが、実環境での実験と比べ、割り当て方法の変更とパラメータの変更が容易となる。また、割り当て方法やパラメータを変更するたびにコンパイルに必要な時間は、実環境での実験と比較すると、少なくすむ。

実環境での実験も大切であるが、できるだけ多くの実験データを効率的に取得することが重要であるため、本研究では乱数を用いた実験をおこなう。実験方法を手順を以下に示す。

1. KVM 上で動作するアプリケーションから、割り当てるオブジェクトの大きさの統計をとり、リストを作成する。オブジェクトを割り当てる場合、乱数を用いて、このリストからオブジェクトの大きさを選ぶ。
2. 記憶領域が一杯になると、全記憶領域に対するある一定の大きさの領域を解放する。
3. 再び、割り当てる。このとき、割り当てることのできたオブジェクトの大きさの総量を算出する。
4. 100 回、割り当て · 解放を繰り返し、割り当てたオブジェクトの総量の平均値を算出する。

この実験結果では、First fit 法が右肩下がりの結果であったが、それ以外の割り当て方法は大きな範囲でみれば横軸に平行であった。平均値でみると Separate first fit 法ともに高い性能を示した。

結論

本研究では、各割り当て方法を比較するためのテスト環境の構築をおこない、単純なアイデアで、性能の向上が期待できることを示した。また、断片化の様子を視覚的に見ることで、各割り当て方法の動作を理解するのに役立つとともに、直感的な改良を施せることが分かる。本研究で提案した割り当て方法は、First fit 法よりも高い性能を示した。