7G－7

プログラム特性を考慮したメモリ負荷制御方式

松本隆明　濱井和夫　池田裕之

(NTT情報通信処理研究所)　(NTTソフトウェア㈱)

1．はじめに

スワッピング制御によるメモリ負荷制御においては、業務プログラムの大型化、端末の高機能化による端末操作時間の短縮化により、従来の小規模プログラムを前提とした方式では、充分な性能が得られないという問題が発生している。本稿では、これに対する一方式について考察する。

2．負荷制御方式上の問題点

システムのメモリ使用特性の変化に伴い、最適かつ効率的なメモリ負荷制御を行うためには以下の問題点を解決する必要がある。

(1) 制御単位が大きい

プログラムが大型化すると、メモリをタスク単位で二次記憶媒体に退避・回復するスワッピング制御やプログラムのローディング処理にかかるCPU時間やI／O時間が増加する。(オーバヘッドの増加)

(2) タスクのWAIT時間が短い

端末の高機能化により、従来人間が端末を操作していた部分を端末側で自動的に応答することが可能となった。このため、いままで端末操作のためにタスクがウェイトしていた時間が短縮され、スワッピング制御の動作間隔が短くなりスラッシングが発生しやすくなる。

(3) 負荷変動が大きい

CPUの性能向上、オペレーティングシステムの提供する論理空間の拡大により、同時に走行可能なジョブ数が増加しているため、メモリ負荷変動の幅も大きくなっている。急激にメモリ負荷が上昇した場合など、集中的にスワッピング制御が動作してスラッシングが発生する。

3．負荷制御単位の分割

上記問題点を解決するため、プログラムの参照特性を活かし、効率的なメモリ負荷制御を行う方式を提案する。一般的にプログラムにはローカリティ(参照の局所性)があるため、新たなプログラムの起動時あるいは端末操作の完了によるタスクのスワップイン時には、局所参照部分だけをメモリにローディングするだけで充分動作可能となる。そこで、スワッピングやローディングにおける制御の単位を以下のように分割する。

(1) スワッピング単位の分割

タスクのスワップイン要求時、当該タスク全体ではなく、スワップアウト時に参照していたアドレス、スワップアウト時のワーキングセットサイズ、システム全体のメモリ負荷から決定される単位(LWS：ローカルワーキングセット)の部分のみを二次記憶媒体から回復する。未回復部分に対するアクセス時は、ページフォルトを契機として、LWS単位で必要部分が回復される。

(2) ローディング単位の分割

新たなプログラムのローディング時も同様に、当該プログラムの開始アドレス、プログラム作成時のセグメントサイズ、システム全体のメモリ負荷から決定される単位(LU：ローディングユニット)の部分のみをロードする。ロード後は直ちに実行可能となる。未ロード部分にアクセスした場合も同様に、ページフォルトを契機としてLU単位で未ロード部分をローディングする。

このように、ローディングやスワッピングの単位をプログラムのローカリティを考慮して小さく設定することにより、対象となるプログラムのサイズに関わりなく充分な性能を得ることが可能となる。

4．二次記憶媒体への先行書き込み

メモリ使用率が高くなり、スワッピング、ページングが集中して発生するとスラッシングにより他のプログラムの動作ができなくなる場合がある。そこで、CPUやI／Oの使用率が低いときにあらかじめ使用頻度の低いページを二次記憶媒体に書き出しておくことにより、実際にメモリが不足したときに速やかにページを置き換える。前述したようにプログラムにはローカリティがあるため、使用頻度の低いページを先行書き込みによって二次記憶媒体に書き込んでから他プログラムのために取り上げられるまでの間に当該ページが更新されて無効となる確率は低い。したがって、先行して二次記憶媒体に書き込むこの方法は非常に有効となる。なお、先行して二次記憶媒体に書き込まれたページはすぐに空きページとはせず仮空き状態としてメモリ内容を保存する。ローカリティの変化により、仮空きページにアクセスされた場合でも、二次記憶媒体から読み込むことなくアクセス可能とする。

5．効果

(1) スワッピング単位の分割による効果

・スワッピング時のオーバヘッドを削減できる。
・スワップイン時に必要なメモリが少なくて済むので、メモリ空き待ちによる長期沈み込みが起こりにくい。

(2) ローディング単位の分割による効果

・大規模なプログラムを分割してロード可能なのでメモリ不足による長期実行待ちが起こりにくい。
・システムの多重度が増大しても大幅なメモリ負荷上昇が起こりにくくなるためスワッピング、ページングの動作の発生を低く抑えることができる。

(3) 二次媒体への先行書き込みによる効果

・従来、メモリ高負荷時に集中して発生していたスワッピング、ページングのI／Oを分散させることができるため、メモリ高負荷時のI／Oオーバヘッドを削減できる。

6．おわりに

メモリデータベースの出現等、システムのメモリはますます大量に使用される傾向にあり、大規模なメモリ使用に適した本方式はさらに有効になっていくものと思われる。

Memory Worklord Control Considering Programs Behavior

Takaaki MATSUMOTO　Kazuo HAMAI　Hiroyuki IKEDA

NTT　NTT　NTT Software Corp.