B-40

ソフトウェアDSMにおけるホーム移動に関する一考察
Home Migration of Software Distributed Shared Memory

阿部 拓弥 † 岡本 秀輔 †
Takuya Abe Shusuke Okamoto

1. まえがき

ソフトウェア分散共有メモリ (Distributed Shared Memory) は、PC・WSノードで構成されるクラスタ環境において並列ユーザプロセス群が共有できる仮想的なメモリ空間をソフトウェアで実現したものである。各ノードでローカルに共有メモリ領域の複製を持っており、あるユーザプロセスが共有メモリ領域にアクセスすることで自ノードの複製を変更すると、システムがその変更を他のノードの複製に反映させる更新処理を行い、メモリ一貫性を保つ。高速化のために、一部のノードの更新処理を省略、遅延させるといった手法も使われる。

これらの一つとして、ホームという概念を用いた手法もある。ホームとは、共有メモリのある領域の最新データを常に保持しているノードのことである。他のノードのユーザプロセスが共有領域に対してアクセスを行う場合には、ホームの最新データによりそのノードの複製の更新を行う。この手法を用いた場合、ある共有領域に対して頻繁にアクセスを行うユーザプロセスが動作するノードをその領域のホームとすると、更新データの送受のためのノード間通信が起こらなくなり高速になる。

本稿では、ホームの役割を最適なノードへ動的に移動させる手法を提案する。これは共有メモリ領域へのアクセス履歴をとり、最近頻繁にアクセスしているプロセスの存在するノードに対してその領域のホームを移動させるものである。

2. 研究の背景

ソフトウェアDSMに関する研究では、メモリ一貫性プロトコルや、ノード間の通信性能向上を目的とした手法の提案等が数多くなされ、標準的な実現手法が確立しつつあるといえる。ホームという概念もまた、ソフトウェアDSMシステムの実現のための一般的手法として用いられるようになったが、研究すべき事柄は多数ある。

ソフトウェアDSMを用いた並列アプリケーションのメモリアクセスパターンとホームの位置は、アプリケーション全体の性能に大きく影響する。そのため、ある共有メモリ領域へのアクセスを頻繁に行うノードとその領域のホームが、一致すれば性能は向上し、逆は通信による性能低下が起こる。

最近の研究ではこの不一致を考慮して、共有メモリ領域を確保する際に、ユーザがホームの位置を指定するメカニズムが取り入れられている場合もある。

一方でホームを動的に移動する手法もある。実際に、並列プロセスのバリア同期を契機にホームを移動する手法 [1] や、共有メモリ領域に書き込みを行った時点で必ず書き込んだノードにホームを移動するという手法 [2] も提案されている。

†茨城大学理工学研究科情報工学専攻

3. 研究の目的

ホームを最適に位置付けるには、前節で述べたようにユーザがホームの位置を指定する静的な手法と、実行中にある条件を満たした時点でホームを移動する動的な手法がある。

静的な手法では、ユーザがアプリケーションのメモリアクセスパターンをホームの位置と関連付けることが可能であるため、ユーザレベルでの高速化が図れる。その反面、そのホーム位置の考慮はユーザの負担を大きくする。特にメモリアクセスパターンが複雑なアプリケーションではその指定が難しい。

本研究では、メモリアクセスパターンを考慮したホームの動的移動の手法に関する考察を行う。一般的にソフトウェアDSMでは、共有メモリ領域をページ単位で管理し、アクセス検知や更新の送受信もまたページ単位で行う。そこで、共有ページへのアクセスが頻繁なノードをそのページのホームに位置付けることで共有メモリへのアクセスの高速化をはかることを目標においた。

4. 動的移動の手法

本研究での動的移動手法の特徴は、アプリケーションのメモリアクセスの状況に応じてホームを最適な位置に移動するという点にある。そのための設計にあたり、次の事項の決定が必要であると考えた。

- ホーム移動の決定に用いる情報 (アクセス履歴)
- ホームの移動条件
- ホーム移動時の処理
- ホーム移動後の処理

4.1 アクセス履歴

ノードX上のプロセスがページPに頻繁にアクセスしているならば、ページPのホームをノードXとしたほうが効率がよい。そのためには各共有ページに頻繁にアクセスしているプロセスを調査する必要がある。そこで、各ページへの書き込み・読み込みアクセスを時系列で記録したアクセス履歴を用いて頻繁にアクセスしているプロセスを検査し、その履歴をホームの移動の決定に用いる。

ここで、アクセス状況の記録、履歴の管理は各ページのホームノードで行う。アクセス履歴の更新は、各プロセスがホーム管理者に対して自ノードの共有領域の複製にアクセスしたことを通知してきたタイミングで行う。

4.2 ウィンドウ

メモリのアクセスパターンは時々刻々と変化するものであり、あるときはプロセス1が共有ページXに頻繁にアクセスし、あるときはプロセス2がページXに頻繁に

アクセスするかもしれない。この変化を捉えるために、前節で述べたアクセス履歴情報に対してウィンドウという概念を導入する。

ウィンドウとは履歴の全体のうちで、最近のアクセスを見るためのもので、ウィンドウサイズは最近のページアクセスの何回までを履歴として考えるかを決定する数値のことである。図1はあるページXに対するアクセス履歴の例を表している。図中の数字はアクセスしたプロセスの識別番号、"r" は読み込みアクセス、"w" は書き込みアクセスを表している。これにウィンドウサイズ4を適用した場合、履歴は図2ように太枠で囲われた部分のみとなる。

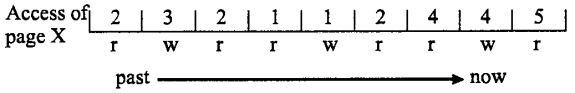

図 1: ページXに対するアクセス履歴の例

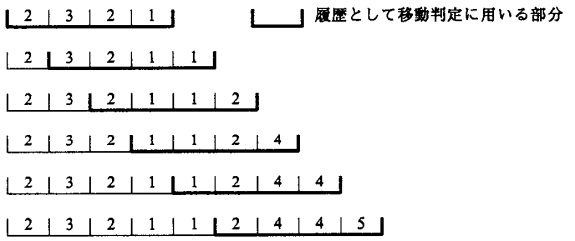

図 2: ウィンドウサイズ4を設定した場合の例

ウィンドウを用いることで、ウィンドウサイズ分のアクセスのみを保持しておけばよく、過去の全てのアクセス履歴を保持しておく必要がなくなるという利点もある。また、ホーム移動の際にはアクセス履歴を転送する必要があるため、その際の通信量も減らすことができる。

4.3 ホームの移動条件

1. ウィンドウサイズ内でアクセス数が一番多いプロセスが動作するノードへそのページのホームを移動する
2. 一度ホームを移動したらその直後の数回のアクセスに関してはホーム移動を行わない (履歴はとり続ける)
3. アクセス数が最大のプロセスが複数あるとき、その中にページのホームで動作するものが含まれていればホームは移動しない
4. アクセス数が最大のプロセスが複数あり、かつその中にホームで動作するものがなければ、書き込みが多いプロセスのノードにホームを移動する

1では、ウィンドウサイズ内でのアクセスが一番多いプロセスが、最近最もアクセスを頻繁に行っているので、そのノードにページのホームを移動するものである。

2,3は頻繁なホームの移動を避けるための条件である。ホームの移動による効果を評価する際の指標の一つに、ホーム移動の頻度が考えられる。2においては、ホーム移動の直後の何回移動を行わないかというパラメータを検討していきたい。3においてはホームプロセスのアクセス回数を重視しているが、ホームでないプロセスのアクセス回数を重視してホームの移動を起こすという選択肢もある。

4において書き込みが多いプロセスに対して優先的にホームが移動するのは、プロセスが共有メモリに書き込みを行う際に同じ場所に頻繁に書き込むケースが多いと判断したからである。

4.4 ホーム移動時の処理

ホーム移動の判定は履歴を管理しているホームノード上で行い、ページに対するアクセスを検知したタイミングで、ホーム移動条件を満たしていればホームの移動を行う。

ホーム移動時には、現在のホームから新しくホームになるべきノードのホーム管理者対してホーム移動要求を送り、そのページの最新データだけでなく、以後の管理のために履歴情報も送信する。

4.5 ホームの移動後の処理

実際にホームの移動を行ったら、その他のプロセスに対してホーム移動の通知を行う。全てのノードに通知を行うとその部分で通信オーバヘッドが生じるので、移動通知はページアクセスの履歴を調べ、最近そのページにアクセスしているプロセスにのみ行う。それ以外のプロセスは、移動が起こったページにアクセスを行い、元ホーム管理者に対して更新データを転送・要求を行う際に、新ホームの位置が伝わるようにする。

5. まとめ

本研究では、ソフトウェアDSMに関するホームの動的移動に関する考察を行った。ホーム移動条件のために各共有ページに対するアクセス履歴、ウィンドウサイズを用いた。ホーム移動決定はホームで行うようにし、アクセス履歴もホームで管理するようにした。

問題点として、アクセス履歴に読み込みアクセスを含めることにより、並列プロセスが行う強制的な読み込み検知でのオーバヘッドが大きくなる可能性が挙げられる。また、メモリアクセスパターンによっては無駄なホーム移動を行ってしまうかもしれないため、どのタイプの並列アプリケーションに適しているかを考える必要がある。

参考文献

[1] Jae Woong Chung , Byeong Hag Seong , Kyu Ho Park, "Moving Home-based Lazy Release Consistency for Shared Virtual Memory Systems"

[2] Benny Wang-Leung Cheung , Cho-Li Wang , Kai Hwang ,"JUMP-DP: A Software DSM System with Low-Latency Communication Support"

[3] 阿部 拓弥 , 岡本 秀輔 , "ソフトウェア DSM ライブラリにおける更新差分の分割管理手法", 並列処理シンポジウム (JSPP)2001