プロファイルを利用したDSMプログラムのパフォーマンスチューニング

堀内律之† 吉川克哉‡ 城田祐介‡ 吉瀬謙二‡ 本多弘樹‡ 弓場敏嗣‡

†電気通信大学電気通信学部 ‡電気通信大学大学院情報システム学研究科

1 はじめに

PC クラスタの分散メモリシステムを対象に逐次プログラムを並列化する方法として，ソフトウェア分散共有メモリ（DSM）を用いた方法がある．ソフトウェア DSM では，ソフトウェアの制御によって PC クラスタ上に仮想的な共有メモリが提供されるため，プログラマは明示的な通信を記述する必要がない．したがって，並列プログラミングを容易に行なうことができる．しかし，単に逐次プログラムを並列化しただけではプログラムが高速に実行されるとは限らない．

本稿では，プロファイル情報を利用し，プログラマレベルでパフォーマンスチューニングを行なう手法を提案する．Scope コンシステンシモデル [1] を用いたホームベースソフトウェア DSM システム JIAJIA[2] を利用し，提案手法の評価を行なった結果について報告する．

2 パフォーマンスチューニング手法の評価手順

本研究におけるパフォーマンスチューニング手法の評価手順を**図 1** に示す. まず，パフォーマンスチューニング手法を 3 つ提案し，それぞれの提案手法に対してどのようなプロファイル情報が必要であるか検討する．その後，必要なプロファイル情報を得るためのコードを JIAJIA に挿入する．同システム上で DSM プログラムを実行し，プロファイル情報を取得する．得られたプロファイル情報を用いて DSM プログラムのチューニングを行ない，再度 DSM プログラムを実行する．その実行結果と，チューニング前のものとを比較することで，提案手法の評価を行なう．

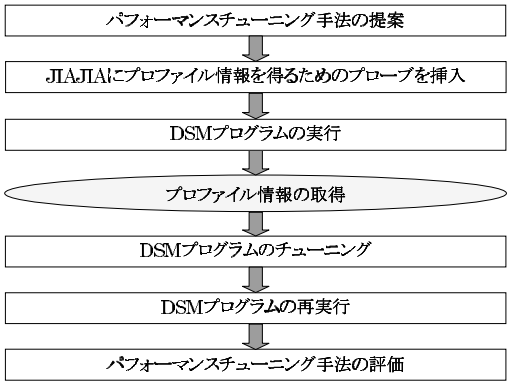

図 1: パフォーマンスチューニング手法の評価手順

3 パフォーマンスチューニング手法の提案

3.1 ページリクエストによるホームノードのコンテンションの回避

本稿では，コンテンションを,「ホームノードに対するページリクエストの集中」と定義する．

DSM プログラムを実行する際，コンテンションの発生がパフォーマンス低下の原因となることが起こり得る．したがって，コンテンションの発生を抑えることによってパフォーマンスチューニングを行なう．

- 必要となるプロファイル情報

  ページリクエストの応答時間（ホームノード以外のノードがホームノードへページリクエストを発行してから，ページを取得するまでの時間）.

- パフォーマンスチューニング手法

  コンテンションの発生をページリクエストの応答時間によって検出する．そして，read しかされない共有ページについて，相当するメモリ領域をそれぞれのローカルメモリにとり，コンテンションの発生を回避することでパフォーマンスチューニングを行なう．パフォーマンスチューニングは，使用可能なメモリに余裕がある場合，検出したページリクエストの応答時間の長いページから順に行なう．

3.2 負荷分散を考慮したホームノードの動的再配置

ホームベースソフトウェア DSM では，ホームノードへのライトバックが過大なオーバヘッドになる．よって，メモリアクセスパターンに適合したデータ配置 (ホームノードの配置) ができるかどうかがパフォーマンスに影響を与える．JIAJIA の Home Migration 機能 [3] を利用すると，あるページに関してバリア間で唯一の write を行ったノードにホームノードが動的に再配置され，上記オーバヘッドが削減される．しかし，この配置

Performance tuning of DSM programs using profile information

Noriyuki HORIUCHI†, Katsuya YOSHIKAWA‡, Yusuke SHIROTA‡, Kenji KISE‡, Hiroki HONDA‡and Toshitsugu YUBA‡

†The University of Electro-Communications

‡Graduate School of Information Systems, The University of Electro-Communications

は必ずしも最適になるとは限らない．したがって，どのような場合にこの機能を使用したらよいのかをユーザに提示することで，パフォーマンスチューニングを行なう．

- 必要となるプロファイル情報
  バリア同期間の実行時間．
- パフォーマンスチューニング手法
  バリア同期間の実行時間によって，Home Migration 機能を適切に切り替えることで，パフォーマンスチューニングを行なう．

3.3 ページ間の関連性を考慮したページサイズの調整

DSM プログラムにおいて，ページのアクセスパターンが連続的な場合，ページサイズを大きくすることで，データのプリフェッチによりリモートページフォールトやページのリクエスト回数を削減することができる [4]．一方で，ページのアクセスパターンが不規則な場合，ページサイズが大きすぎると，取得するページに不必要なデータを含む確率が高くなる．ページサイズに比例して取得時間も大きくなるので，これがオーバヘッドとなる．このように，ページのアクセスパターンによってページサイズを変更することで，パフォーマンスチューニングを行なう．

- 必要となるプロファイル情報
  バリア同期間に起こるページへのアクセス履歴情報．
- パフォーマンスチューニング手法
  上記の履歴情報を用い，バリアごとにアクセスされたページ間の関連性について調べることで，適切なページサイズを決定し，パフォーマンスチューニングを行なう．

4 評価

4.1 評価環境

評価環境として，**表 1** で構成される PC クラスタを用いた．

4.2 評価結果

評価ベンチマークプログラムとして，行列積を行なう Matrix Multiply(MM) を用い，問題サイズを 1024 × 1024 とした．このとき，提案した各手法のチューニング前とチューニング後の実行時間と PE 数の関係を表したものが，**表 2**，**表 3**，**表 4** となる．表中の括弧内は，速度向上率 (チューニング前の PE 数 1 の実行時間/チューニング後のそれぞれの実行時間) を表わす．PE 数 8 の箇所を比較すると，手法 3.1 (**表 2**) では，チューニングにより，5.14 から 7.48 への大幅な速度向上が確認できた．

5 おわりに

本稿では，DSM プログラムのパフォーマンスチューニングを行なうための手法を提案し，その評価を行なった．各々の提案手法で，その有効性が確認された．

今後の課題としては，より多くのベンチマークプログラムによる評価を行なうことが挙げられる．

表 1: 評価環境の PC クラスタ

PE 数	8
CPU	Intel PentiumIII 866[MHz]
Memory	1GB
Network	100BASE-TX
OS	Red Hat Linux7.1

表 2: ページリクエストによるホームノードのコンテンションの回避

PE 数	実行時間 [sec]	
	チューニング前	チューニング後
1	19.53(1.00)	19.52(1.00)
2	10.79(1.81)	10.29(1.90)
4	6.40(3.01)	5.18(3.77)
8	3.80(5.14)	2.61(7.48)

表 3: 負荷分散を考慮したホームノードの動的再配置

PE 数	実行時間 [sec]	
	チューニング前	チューニング後
1	19.53(1.00)	19.53(1.00)
2	12.29(1.59)	10.79(1.81)
4	6.55(2.98)	6.40(3.01)
8	5.04(3.88)	3.80(5.14)

表 4: ページ間の関連性を考慮したページサイズの調整

PE 数	実行時間 [sec]	
	チューニング前	チューニング後
1	19.53(1.00)	19.51(1.00)
2	10.79(1.81)	10.52(1.86)
4	6.40(3.01)	5.76(3.39)
8	3.80(5.14)	3.38(5.78)

参考文献

[1] Iftode, L., Singh, J. P. and Li, K.: Scope Consistency : A Bridge between Release Consistency and Entry Consistency, *Proc. of the 8th ACM Annual Symp. on Parallel Algorithms and Architectures(SPAA'96)*, pp. 277-287(1996).

[2] Hu W., Shi, W. and Tang, Z.: JIAJIA: A Software DSM System Based on a New Cache Coherence Protocol, *HPCN Europe*, pp. 463-472(1999).

[3] Hu W., Shi, W. and Tang, Z.: Home Migration in Home-Based Software DSMs, *Proc. of the 1st Workshop on Software Distributed Shared Memory (WSDSM'99)*, (1999).

[4] 田邊浩志, 吉瀬謙二, 本多弘樹, 弓場敏嗣. 通信粒度を動的に変更するソフトウェア分散共有メモリ, 電子情報通信学会総合大会, D-6-5, p. 56(2002).