4ZB-4

ソフトウェアDSM上でのアプリケーション実行時間の定式化

鈴木 祥† 坂口 朋也† 多 忠行‡ 吉瀬 謙二‡ 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 クラスタのような分散メモリ環境上に仮想共有メモリ空間を構築するソフトウェア分散共有メモリシステム (S-DSM) が開発されている．S-DSM 上での並列プログラミングは，MPI などのメッセージパッシングライブラリ比べてプログラムの可読性や書きやすさに優れている．代表的な S-DSM の実装には，TreadMarks[1],JIAJIA[2],SMS[3] 等がある．

S-DSM には対象となるアプリケーションとの整合性が存在する．アプリケーションによっては期待された性能が発揮されないことが知られている．S-DSM の特徴を考慮したプログラミングを行うことで，アプリケーションの性能向上が期待できる．従って，S-DSM とアプリケーションとの整合性の解明は重要である．

本稿では，S-DSM とアプリケーションとの整合性を調査するため，実験を通しアプリケーション実行時間の定式化を目指す．プロセッサ数をパラメタとして定式化し，最小値，つまり最も実行時間の短くなる構成を導出する．

また，プログラミング方式の違いによる実行時間への影響も調査する．いくつかの異なる仮想共有メモリ空間上へのデータ配置法を実装し，それぞれについて定式化を試みる．それにより，適切なデータ配置方式の検討を行う．

2 アプリケーション実行時間の定式化

本稿では，S-DSM のひとつである JIAJIA をベースに構築された Mocha ver0.4 を定式化の対象とする．

Mocha の大きな特徴のひとつに、ホームベース型の S-DSM であることが挙げられる．ホームベース型とは，各プロセッサは全体の仮想共有メモリ空間の一部をそれぞれホームとして最新のデータを保有する方式である．自身がホームでないデータへアクセスする場合にプロセッサ間通信によるデータの授受が行われることで，一貫性が保証される．各プロセッサがホームとして持つ仮想共有メモリのサイズは，ユーザが任意に指定できる．

2.1 アプリケーションと方式

アプリケーションとして行列積を行う MM を対象とする．問題サイズは 4096 × 4096 とし，以下の 3 つの仮想共有メモリ空間上のデータ配置方式を対象とする．

- **横ブロック分割法 (方式 1)** 行列 A，B，C を各プロセッサが均等にブロック分割する (JIAJIA に付属の方式)．
- **縦ブロック分割法 (方式 2)** 行列 A，B は方式 1 と同様，行列 B は縦にブロック分割する．
- **非計算ノード作成法 (方式 3)** 計算を行わないノードを 1 つ置き，このノードが行列 B のデータをすべてホームとして持つ．

2.2 定式化の方法

アプリケーション実行時間を 2 つの要素に分けて考える．

- データ送受信や同期の S-DSM 内部処理の時間
- 実際の計算部分の時間

以下，前者を S-DSM 時間，後者を計算時間と呼ぶこととする．プロセッサ数の増加とともに，各プロセッサの S-DSM 時間は増加，計算時間は減少していくことが予想される．

S-DSM Mocha に， S-DSM の内部処理のログを出力させる機能を追加する．得られたログファイルから静的に S-DSM 時間と計算時間の分離を行う．

計算の際に自身がホームとして保有していないデータが必要になると，その都度そのデータを保有しているホームノードにデータを要求し，データを受信するという S-DSM の内部処理に入る．このため，計算時間のみを抽出することは困難である．逆に，S-DSM の処理の開始終了は明確なため，ログをファイルに出力させることは可能である．そこで，S-DSM の処理時間の合計を求め，総実行時間との差を求めてこれを計算時間と定義する．

2.3 定式化のための実行時間の測定

測定に用いた環境を表 1 にまとめる．この環境でアプリケーションのログを測定する．

表 1: 実験に用いた PC クラスタの構成

CPU	Intel Pentium4 Xeon 2.8GHz
メモリ	512MB
ネットワーク	ギガビットスイッチ
OS	RedHat Linux 9

Execution Time Analysis of a Matrix Multiply Program on Software DSMs
† Sho SUZUKI, Tomoya SAKAGUCHI
‡ Tadayuki OHNO, Kenji KISE, and Toshitsugu YUBA
The University of Electro-Communications (†)
Graduate School of Information Systems, The University of Electro-Communications (‡)

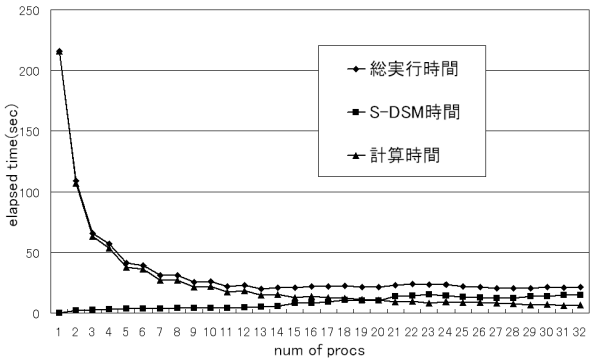

図 1: 方式 1-横ブロック分割法

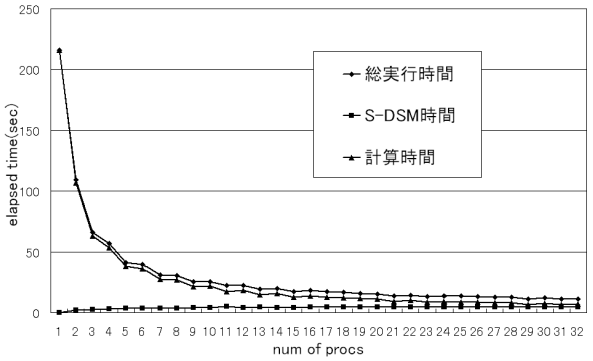

図 2: 方式 2-縦ブロック分割法

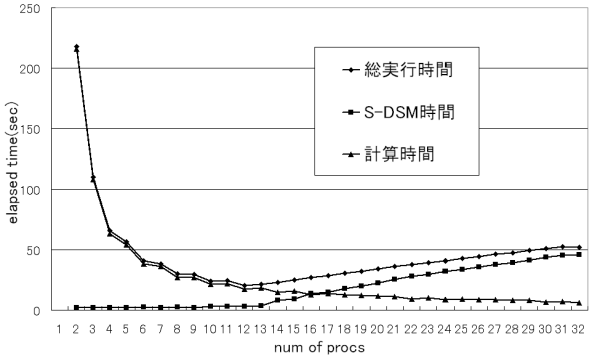

図 3: 方式 3-非計算ノード作成法

2.4 実行時間の分離と式の導出

得られたログファイルから，それぞれの方式において S-DSM 時間と計算時間の分離を行う．図 1，図 2，図 3 に結果を示す．

S-DSM 時間については，最小二乗法を用い 1 次関数に近似した式で表す．また計算時間については，線形スピードアップが得られているため，プロセッサ数に反比例した式で表すことができる．

分離した S-DSM 時間と計算時間をそれぞれプロセッサ数を N として定式化し，3 つの手法について導かれた式は表 2 の通りである．

表 2: 定式化結果 (sec.)

横ブロック分割法	$216/N + 0.487N + 0.445$
縦ブロック分割法	$216/N + 0.0731N + 2.84$
非計算ノード作成法	$216/(N-1) + 1.70N - 10.4$

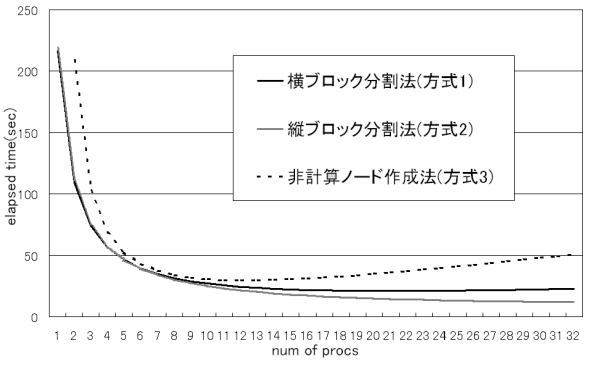

図 4: 定式化による 3 方式の比較

3 定式化結果を用いた議論

3 つの手法を定式化したグラフを図 4 に示す．JIAJIA に付属のアプリケーション MM は横ブロック分割法 (方式 1) であったが，行列積の計算は行列 B の要素を列方向に必要とするため，方式 2 では縦に分割する方法をとった．その結果，通信時間の減少につながった．

方式 2 の実行時間はグラフの上ではプロセッサ数 32 台において 11.4 秒と最小である．しかし，いまだ減少傾向にあり最小値となるプロセッサ数はより大きいと考えられる．そこで，定式化により求めた方式 2 の式における最小値を求める．$N = 54.3$ つまりプロセッサ数 54 台にて実行時間が最小となり，その実行時間は 10.8 秒と予想される．

4 おわりに

実行時間の定式化を行うことで，行列積の計算における適切なデータ配置方式やプロセッサ数を得ることができた．また，測定していない範囲のデータを予測し導き出すこともできた．行列積を行うアプリケーション MM は，実装方式を変えることで非常に小さな S-DSM 時間で実行することができ，かつ台数効果もあるので S-DSM との整合性はよいといえる．

今後の課題は，より多くのアプリケーションについて同様の定式化を行い，S-DSM との整合性の評価や適切な手法の検討を行っていくことである。

参考文献

[1] P.Keleher, S.Dwarkadas, A.L.Cox and W.Zwaenepoel: TreadMarks: Distributed Shared Memory on Standard Workstations and Operating Systems, *Proc. of the Winter 1994 USENIX Conference*, pp. 115–131 (1994).

[2] M.Rasit Eskicioglu, T.Anthony Marsland, Weiwu Hu and Weisong Shi: Evaluation of the JIAJIA Software DSM System on High Performance Computer Architectures, *HICSS-32* (1999).

[3] 緑川博子, 飯塚肇: ユーザレベル・ソフトウエア分散共有メモリ SMS の設計と実装, 情報処理学会論文誌 Vol.42, No.SIG9(HPS 3), pp. 170–190 (2001).