PGAS 言語 X10 による半正定値計画法の実装と評価

渡部 優[†1,†4] 藤澤 克樹[†2,†4] 鈴村 豊太郎[†1,†3,†4]

1. 研究背景・目的

近年では 1 つの CPU に複数のコアを載せたマルチコア・メニーコア CPU が主流となっている．また，GPU を汎用演算処理に用いたヘテロ型アーキテクチャや，それらを組み合わせた大規模クラスタなど，プログラミングにおける計算機の環境が大きく変化している．そのような環境の中で，計算機資源を活かしたアプリケーション開発を行うには，高性能・高生産なプログラミング言語が求められる．そこで本研究では，高性能・高生産の両立を目標に開発されている PGAS 言語 X10 に焦点を当て，実際に並列アプリケーションとして半正定値計画法を実装し，大規模クラスタ上で評価を行う．そして，その実装・評価を通して，X10 の並列分散プログラミング言語としての有用性・問題点を明らかにすることを目的とする．

本研究の中長期的な実装目標としては，X10 を用いることで GPU までの統一的な実装を行い，400 ノードまでスケールするような並列アプリケーションの構築・評価を行うことであるが，本稿では現実装における複数ノードでの分散実行から得られた実験結果と知見を報告する．

2. PGAS 言語 X10

X10[1] は，高性能・高生産の両立を目標に IBM Research により開発されているプログラミング言語で，PGAS(Partitioned Global Address Space) と呼ばれるモデルを採用している．PGAS とは，単一アドレス空間をプレースと呼ばれる単位で分割し，分散メモリ環境を抽象化してプログラマに見せることで，並列処理の記述を容易に行えるようにしたモデルである．

X10 は高生産なアプリケーション開発のために，軽量な非同期タスクを生成する “async”，同期処理を行う “finish” や “clocked”，分散処理を実現する “at”，排他制御のための “atomic” などの構文を提供している．また，このほか，GPU などの異なるアーキテクチャ上でも統一的にプログラムが記述できるように設計されている．本研究では実際に，上記に述べた並列分散機構を用いて半正定値計画法の実装を行う．

†1 東京工業大学 - Tokyo Institute of Technology
†2 中央大学 - Chuo University
†3 IBM 東京基礎研究所 - IBM Research Tokyo
†4 JST CREST

3. 半正定値計画問題

3.1 半正定値計画問題とは

半正定値計画問題（SDP:SemiDefinite Programming）とは，特定の制約条件のもとで目的関数を最小化，あるいは最大化させる最適化問題の一つで，線形計画問題や二次錐計画問題などを含んだ大きな凸計画問題の枠組みである．SDP は組合せ最適化問題や整数計画問題，ノルムなどを用いた配置問題，システムと制御，ロバスト最適化，量子化学など，幅広い応用範囲を持った問題領域である．この SDP のソルバーとして藤澤らは SDPA[2](SDP Algorithm) を提案・実装している．

3.2 SDPA とボトルネック

藤澤らが提案・実装する SDPA では主双対内点法による反復解法を用いている．この解法におけるボトルネックとなる主な箇所として，Schur Complement Matrix（以下，SCM）の計算とコレスキー分解があるが，これらの計算は並列に計算することができるので，SDPA の実装では OpenMP を用いたスレッド並列で高速化を図っている．

4. X10 を用いた SDPA の実装

SDPA におけるボトルネック箇所は，前章で述べた通り SCM の計算とコレスキー分解であるが，本研究における SDPA の X10 実装（以下，SDPX）でも，X10 の並列分散処理機構を用いることで SDPA 同様の並列実装を行った．また，さらにこの実装を拡張し，より大規模な SDP に対応できる SDPARA[3]（SDPA の MPI 実装）をもとに X10 による分散実装を行った．現在の SDPX で並列化できた箇所は SCM の計算に限られるが，複数ノードで分散して処理できるため高速化が期待できる．なお，コレスキー分解には数値計算

ライブラリとして SDPA 同様に BLAS を用いている．

5. X10 実装 SDPX の性能評価

5.1 実験環境

実験には東京工業大学のスーパコンピュータ TSUBAME 2.0，Intel Xeon 2.93 GHz, 6 コア x 2 （24 スレッド），Memory 54GB，QDR InfiniBand x 2 80Gbps, SUSE Linux Enterprise Server 11 SP1 の 4 ノードを用いた．また，コンパイラは GNU GCC 4.3.4 と X10 2.2.2.1，X10 の MPI 通信には OpenMPI 1.4.2，BLAS ライブラリに GotoBLAS2 1.13 を利用した．また SDP のデータセットとして，control11 (制約式数 m=1596, 行列サイズ n=165) を用いた．

5.2 SDPX マルチスレッド実行の評価

図 **1** は既存の SDPA と本研究の SDPX 実装のマルチスレッド実行による性能比較を行ったもので，SCM とコレスキー分解の計算時間を示している．まず，図 1 左側の SDPA では 8 スレッド実行時に，コレスキー分解を含めて，最大 2.7 倍の性能向上が確認できる．

一方，図 1 右側の SDPX 実装でも，12 スレッド実行時に最大 2.7 倍性能が向上した，ただし，SDPX においてコレスキー分解の性能向上が見られないが，これは X10 側と BLAS ライブラリ側でスレッドマネージャが異なり，現実装では統一的なスレッド管理ができていないことに起因する．また，SDPX の SCM 計算の一部にも BLAS ライブラリに依存する計算が含まれており，その計算が律速となっている．

5.3 SDPX 分散実装の評価

図 **2** は中央より左側が 1 ノードあたり 1 プロセス 12 スレッド，右側が 1 ノードあたり 2 プロセス 6 スレッドで，1 ノードから 4 ノードまで変化させたときの実行結果を示している．トータルの実行時間で見ると，SCM 計算後に発生する行列のマージ（全対全通信）がオーバヘッドとなりプロセスの増加に伴い性能が低下する要因となっているが，SCM 計算のみに関しては，1 ノード 12 スレッド実行に対し，2 ノード 12 スレッド実行で約 2.2 倍，4 ノード 12 スレッド実行で約 3.1 倍性能が向上した．

6. まとめと今後の展望

本研究の実験により，現在の X10 による半正定値計画法の実装で，マルチスレッドでは SDPA と同程度の最大 2.7 倍の性能向上，また，複数ノードでの実行においては全対全通信のオーバヘッドを除いてスケールすることを確認した．一方，外部ライブラリとのスレッド管理をどのように行うべきか，また大規模実行

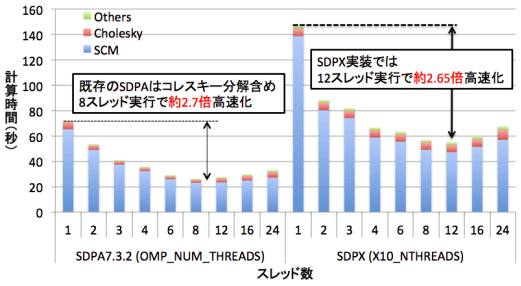

図 **1** 入力データ control11 における SDPA と SDPX の 1 ノード内マルチスレッド実行時の性能比較

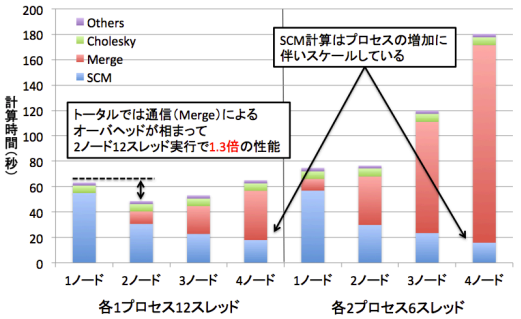

図 **2** 入力データ control11 における SDPX のマルチノード性能

時の通信のオーバヘッドをどのように解決すべきかなど，多くの課題が見えてきた．今後は実装の見直し・最適化を行い，さらには X10 の特徴を活かしたコレスキー分解の GPU 実装などを行いスケーラビリティのある実装を目指す．そしてその実装・評価を通して，並列分散プログラミング言語としての X10 処理系の有用性や問題点を明らかにしていく．

参 考 文 献

1) Philippe Charles, et al. , “X10: an object-oriented approach to non-uniform cluster computing”, In OOPSLA, pages 519-538. ACM, 2005.
2) Makoto Yamashita, et al., “A high-performance software package for semidefinite programs: SDPA 7,” Research Report B-460 Dept. of Mathematical and Computing Science, Tokyo Institute of Technology, Tokyo, Japan, September, 2010.
3) Makoto Yamashita, et al., “SDPARA : SemiDefinite Programming Algorithm PARAllel version” Operations Research. Research Reports on Mathematical and Computing Sciences October 2002, B-384