TSUBAME 共同利用 平成 25 年度 学術利用 成果報告書

高性能と高生産性を両立する並列分散ランタイムシステム
Parallel and Distributed Runtime System Realizing High Performance and High Productivity

田浦 健次朗
Kenjiro Taura

東京大学 情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo

邦文抄録（300 字程度）

当研究グループでは、大規模並列計算機において高性能な並列プログラムを容易に開発できるようにすることを目的として、タスク並列プログラミングモデルに関する研究を行っている。本研究課題では、昨年度から引き続いて、当研究グループで開発している分散メモリ並列計算機向けタスク並列処理系 MassiveThreads/DM の性能改善・機能追加を実施し、TSUBAME 上での動作確認と性能評価を行った。結果として、二分木上再帰タスク生成、Unbalanced Tree Search、N クイーン問題の 3 つのベンチマークプログラムについて、171 ノード 2048 コア使用時に良好な負荷分散性能が得られることを確認した。

英文抄録（100 words 程度）

Our group has studied task-parallel programming models to improve productivity in developing high performance parallel programs on large-scale distributed memory machines. In this project, we conducted performance improvement for MassiveThreads/DM, a library providing task-parallelism on distributed memory machines, and performance evaluation on the TSUBAME supercomputer. As a result, we confirmed that MassiveThreads/DM achieved good scalability in three benchmark programs (Binary Task Creation, Unbalanced Tree Search, NQueens) with 2048 processors.

Keywords:
Parallel programming, task-parallelism, global work stealing, global address space

背景と目的

近年、並列計算機の複雑化が進んでいる。特に分散メモリ並列計算機は、単に構成する計算ノードの数が増えているだけでなく、計算ノード内に複数の CPU を持ち、さらに CPU 内にプロセッサコアを複数持つような複雑な構成も珍しくなくなっている。このような並列計算機の性能を最大限引き出すためには、複数のメモリ・ネットワーク階層について熟知し、その知識に基づいて並列プログラムを記述しなければならない。

当研究グループでは、この問題を解決するために、(1)分割統治形式による通信効率の良い並列アルゴリズム（cache-oblivious アルゴリズム）を設計し、(2)そのアルゴリズムによって再帰的に分解されるタスク・データを、各メモリ・ネットワーク階層に適切に配置するようなタスク並列処理系の実現を目指している。このようなタスク並列処理系は、共有メモリ並列計算機では軽量マルチスレッド処理系（Cilk, Java fork/join framework など）という形で実現され、広く使われるようになってきている。

当研究グループでは、その一環として、次の二つのシステムから構成される分散メモリ並列計算機向けタスク並列処理系 MassiveThreads/DM を提案する。

1. 分散メモリ並列計算機において、ノード間負荷分散を実現するタスク並列ライブラリ:
   従来のタスク並列ライブラリの多くは共有メモリ計算機向けであり、ノード内での動的な負荷分散機能を提供する一方で、ノード間で負荷分散を行う機能を提供しているものは数少ない。また、ノード間負荷分散機能を提供している処理系についても、コード変換を用いてタスクのマイグレーションを実装しており、利用するためには専用のコンパイラを使う必要があるなど、利便性の点で問題がある。

   この問題に対して、提案するタスク並列ライブラリでは、タスクのもつスタックをノード間で転送する

ことにより、コード変換を用いない C ライブラリとしての実装を実現する。

2. タスク並列処理系からの利用に特化した大域アドレス空間ライブラリ:

タスク並列モデルと大域アドレス空間モデルを組み合わせるにあたっては、タスク並列モデルの「計算機のもつ並列性と比べて非常に多くのタスクが生成される」という特徴が問題となる。既存の大域アドレス空間モデルにおいては、大域メモリにアクセスする際に、アクセスするメモリ領域の実体が存在するノードとの通信が発生する。これは、前述のタスク並列モデルの性質と組み合わせたときに、大量に生成されたタスクがそれぞれ通信を行うこととなり、(1)通信の準備にかかるオーバーヘッドがタスクの数だけ発生し、また(2)データ量の少ない通信によって通信帯域を活用できないといった点で、計算機のもつ性能を最大限活用する上での障害となる。

提案する大域アドレス空間ライブラリでは、この問題を解決するため、既存の大域アドレス空間モデルに対して、次の二つの機能を新たに導入する。

(a) キャッシュ機能: タスク間で共有されるキャッシュ領域を提供する。この機能を利用することによって、あるタスクがまとめてデータを読み込み、他のタスクは通信なしでキャッシュからデータを読むだけで済むようにすることが可能になる。

(b)大域メモリの配置を動的に変更する機能(ページマイグレーション): タスクがプロセス間を移動した際に、タスクのアクセスする大域メモリも同時に移動させることができる。

概要

本利用課題では、昨年度から引き続いて、TSUBAME を用いた MassiveThreads/DM の設計・開発を実施した。今年度は主に軽量タスク機構について、実装方法の見直しによって、タスク生成コストを低減するとともに、負荷分散性能の改善に取り組んだ。また、これまで用いてきたワークスティーリングによる動的負荷分散に加えて、新たにライフライン方式[Saraswat et al. 2011]による動的負荷分散を実装し、比較評価を実施した。ライフライン方式は、プロセッサ間のタスク分散の流れをあらかじめライフライングラフと呼ばれるグラフで表現しておき、ワークスティーリングが失敗したときにライフライングラフにより規定されるタスクの流れに基づいてタスクを受け渡すことにより、タスクを分散させていく負荷分散方式である。ライフライン方式を用いることによって、ワークスティーリングによる通信処理コストを低減させることができる。

また、大域アドレス空間機構については、従来は分散配列の各プロセッサへの分散方式は 1 次元分散のみであったところを、二次元分散を指定可能とするなど、プログラマによる性能チューニングのための機構を新たに実装した。

結果および考察

MassiveThreads/DM の細粒度スレッド機構に関して、TSUBAME 上で動作確認をするとともに処理系の性能を評価するため、TSUBAME の 171 ノード 2048 プロセッサを用いた構成で評価実験を実施した。

動作確認と性能評価にあたっては、次の 3 つのベンチマークプログラムを用いて、そのタスク生成性能(1 秒間あたりのタスク生成数)を計測した。

1. 二分木状再帰タスク生成: 指定した再帰レベルまで、各タスクが二つの子タスクを再帰的に生成するベンチマークである。比較的負荷分散が容易なベンチマークであり、処理系の基礎的な負荷分散性能を評価するために用いる。
2. Unbalanced Tree Search: 乱数を用いて不規則な状態空間を再帰的に探索するベンチマークである。このベンチマークは、負荷分散性能の評価を目的として作られたもので、タスク並列処理系の評価に広く用いられているものである。
3. N クイーン問題: N クイーン問題を解くベンチマークプログラムであり、BOTS ベンチマークスイートから MassiveThreads/DM 上に移植した。このベンチマークもタスク並列処理系の評価に広く用いられているものである。

各ベンチマークの評価結果を図 1、図 2、図 3 に示す。グラフ凡例中の lifeline は、ライフライン方式による負荷分散を示し、random backoff は、単純なワークスティーリングによる負荷分散に加えて、ワークスティーリング失敗時にワークスティーリングにかかった時間に基づいたバックオフを設けたものを示す。random backoff によるバックオフ時間は、前回のワークスティーリングにかかった時間を T_{steal} として、rand(0, 1) × T_{steal} で決定される。また、exponential backoff は、random backoff におけるバックオフ時間の上限を指数的に増やしたもので、そのバックオフ時間は、ワークスティーリングに連続で失敗した回数を N_{fail}（≦32）、前回のワークスティーリングにかかった時間を T_{steal} として、rand(0, 1) × 2^{Nfail} × T_{steal} という式を用いて決定される。

結果として、各ベンチマークについて、2048 プロセッサ構成までの動作を確認し、良好な台数効果が得られていることがわかった。また、どのベンチマークプログラムについても、ライフライン方式を用いるよりも通常のワークスティーリングにバックオフを加えたものの方が良好な性能が得られていることがわかる。この原因は現在調査中であるが、ライフライン方式は、アルゴリズムの性質上、従来のワークスティーリングと比較して 1 度に盗まれる(潜在的な)タスクの数が小さくなりがちで、結果としてプロセッサ間のタスクの受け渡し回数が増加してしまうことによると推測している。

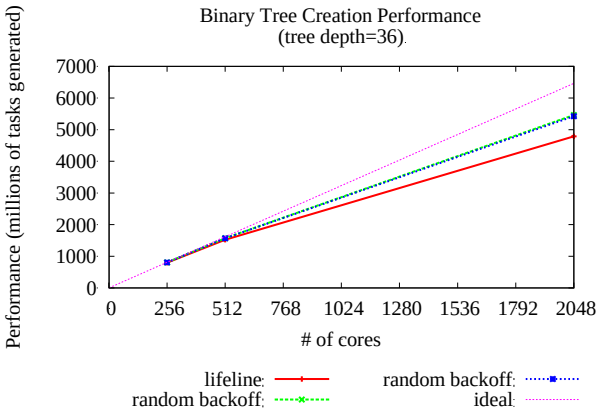

図 1. 二分木状タスク生成のタスク生成性能

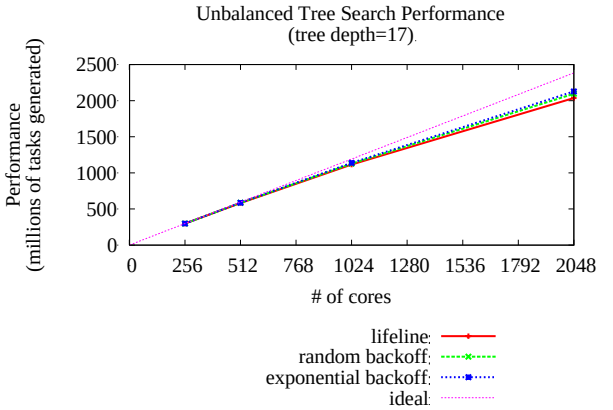

図 2. Unbalanced Tree Search のタスク生成性能

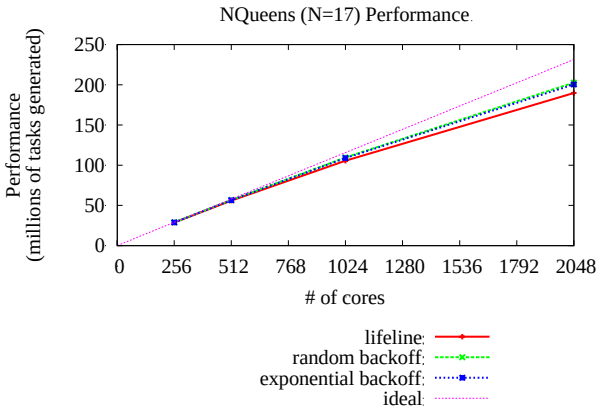

図 3. N クイーン問題のタスク生成性能

まとめ、今後の課題

本利用課題では、昨年度から引き続き、分散メモリ並列計算機における並列プログラミングの生産性を改善することを目的として、分散メモリ並列計算機向けタスク並列ライブラリ MassiveThreads/DM の設計・開発を実施した。TSUBAME 上での性能評価の結果、二分木状タスク生成、Unbalanced Tree Search、N クイーン問題の 3 つのベンチマークプログラムについて、171 ノード 2048 プロセッサ構成で良好な台数効果が得られていることを確認した。また、単純なワークスティーリングに加えて、ライフライン方式による負荷分散の性能評価を実施し、単純なワークスティーリングにランダムバックオフを加えたものについて、ライフライン方式よりも良好な性能を得られることを確認した。

今後は、今回実験を行わなかった、大域メモリへのアクセスがボトルネックとなるようなアプリケーションについて、性能評価を進めるとともに、通信回数を抑えるようなタスクスケジューリング手法の導入や、効率的な大域メモリの配置手法を検討していく予定である。