TSUBAME 共同利用　平成 24 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

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、その一環として、分散メモリ並列計算機においてノード間負荷分散を自動化するタスク並列ライブラリと、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を設計・実装し、TSUBAME 上での動作確認と性能評価を実施する。結果として、タスク並列・大域アドレス空間の両方を使用する基礎的なベンチマークプログラムにおいて、良好な台数効果が得られることを確認した。

Our group studies task-parallel programming models to improve productivity in developing high performance parallel programs. In this project, as part of this effort, we design and implement a task-parallel library supporting automatic load balancing in distributed memory machines and a global address space library specialized for task parallelism. As a result, our system could achieve good speedup in a benchmark program using both task parallelism and global address space.

Keywords: parallel programming, task parallelism, load balancing, global address space

1　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、構成する計算ノードが増加していることに加えて、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階層的な構成などを採用するもの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ず、このことが高性能な並列プログラムの開発を難しいものとしている。

当研究グループでは、この問題に対して、(1)分割統治形式で通信効率の良い並列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生成されるタスクを、各メモリ階層に適切に配置するようなタスク並列処理系を実現することによって、解決することを試みている。

本利用課題では、その一環として次の二つのシステムを設計・実装し、TSUBAME 上で動作確認と性能評価を行うことを目的とする。

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[1]。

(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。

２　概要

提案システムを実装するにあたっては、まず大域アドレス空間ライブラリを実装し、その後、大域アドレス空間ライブラリの機能を用いてタスク並列ライブラリを実装する。なお、大域アドレス空間ライブラリを実装するにあたっては、下位通信ライブラリとして GASNet を用いる。

提案する大域アドレス空間ライブラリは、Partitioned Global Address Space (PGAS)をベースとして、キャッシュ機能とページマイグレーションを提供する。大域メモリに対するキャッシュ機能を導入するにあたっては、キャッシュの一貫性の保証をプログラマの責任とするようなプログラミングモデルを採用することによって、高いスケーラビリティを実現する。

ページマイグレーションを導入するにあたっては、大域メモリをページと呼ぶ単位で分割し、各ページについてホームプロセスと呼ぶプロセスを割り当てる。ホームプロセスは、ページがどのプロセス上に存在するかを管理するプロセスで、大域メモリにアクセスする際には、まずアクセスするページのホームプロセスにデータの所在を問い合わせてから、データの転送を行う。

タスク並列ライブラリに関しては、共有メモリ計算機向け軽量マルチスレッドライブラリである MassiveThreads をベースとして、ノード間負荷分散機能を実装する。MassiveThreads をノード間負荷分散に対応させるにあたっては、MassiveThreads の提供する軽量スレッドをノード間で移動させる必要がある。提案ライブラリでは、コード変換を用いない実装手法として、スレッドのスタックを全プロセスで一意な仮想アドレス上に割り当て、移動先プロセスにおいて移動元と同じアドレスにスタックを配置することによって、スレッドのノード間移動を実現する。

３　結果および考察

提案する大域アドレス空間ライブラリとタスク並列ライブラリの有効性を調査するために、評価実験を行った。まず、開発した大域アドレス空間ライブラリの基本性能を調査するため、2 つのノードを用いてデータサイズを変化させながら、通信にかかった時間を計測した。その結果を図 1 に示す。 local は大域メモリの実体が自ノードに存在する場合、remote は大域メモリの実体が他ノードに存在する場合を示している。図 1 を見ると、GASNet の GET 操作には 2-3 マイクロ秒程度かかるのに対し、提案システムの通信遅延は 10 マイクロ秒程度であった。一方自ノードに大域メモリの実体が存在する場合については、GASNet では 10 ナノ秒程度であるのに対し、1 マイクロ秒程度かかっている。これは、大域メモリを管理するページテーブルの操作が複雑であることと、実装の最適化が進んでいないことが一つの原因と考えている。また、すでにキャッシュされたデータへのアクセスにかかる時間を計測した結果、70 ナノ秒程度でアクセス可能であることがわかっている。

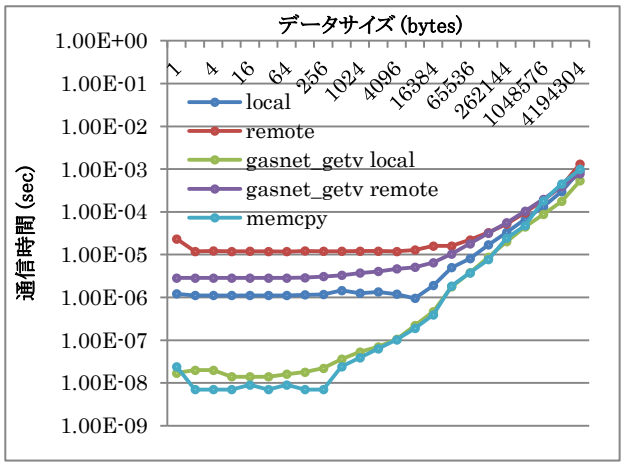

図 1. 大域メモリに対する GET 操作の実行時間

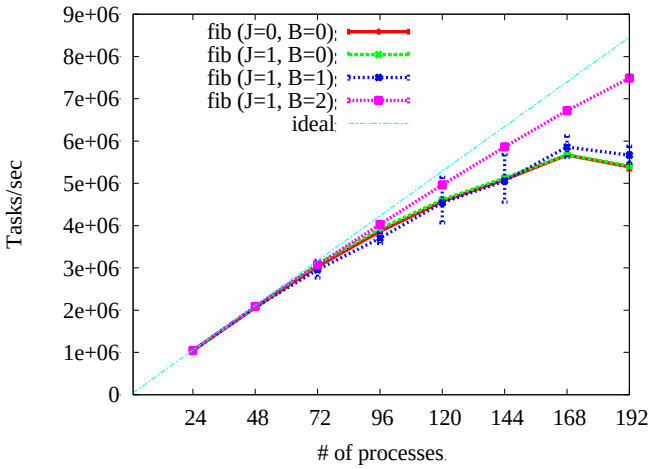

図 2. フィボナッチ数計算におけるタスク生成性能
（1 秒間あたりタスク生成数）

さらに、開発したタスク並列ライブラリの基本性能を調査するため、フィボナッチ数列を再帰的に計算するベンチマークプログラムを用いて、タスク生成のコストとスケーラビリティを計測した。その結果を図 2 に示す。図中の J はタスク間同期処理の際に、タスクを生成したプロセスにタスクを移動させた後に同期処理を行ったかどうかを示し、B はワークスティール失敗時に次のワークスティールを試行するまでのバックオフ時間を設定するか(B=1)、ランダムなバックオフ時間を設定するか(B=2)、バックオフ時間を設定しないか(B=0)を示している。結果として、J=0 から J=1 とした場合には、発生する通信の数は減少したものの、性能はほとんど改善しなかった。また、バックオフ時間に関しては、ランダムでないバックオフ時間を設定した場合では若干の性能向上が見られたが、実行ごとに性能の揺れが激しくなっている。さらに、バックオフ時間にランダム性をもたせた場合には、良好なスケーラビリティが得られている。これに関しては、ランダムなバックオフを設定することによって、ワークスティールが多発するフェーズにおける通信の衝突が緩和されたことが確認できている。

4　まとめ、今後の課題

本利用課題では、分散メモリ並列計算機における並列プログラミングについて、生産性と性能を両立するタスク並列プログラミングシステムを設計・実装し、その性能評価を行った。結果として、フィボナッチ数など基本的なベンチマークプログラムについて良好な台数効果が得られた。

今後は、より多くのプロセッサを用いた場合の性能評価や、行列積など大域的な配列を用いる必要のあるベンチマークを用いて性能評価を実施していく予定である。また、当研究グループでは、本利用課題と並行して、共有メモリ並列計算機での Adaptive Mesh Refinement や Fast Multipole Method といった数値計算手法のタスク並列実装[2,3]を行っており、これらの手法について、提案ライブラリを用いて並列化し、その性能を評価したいと考えている。

参考文献

[1]　秋山茂樹，田浦健次朗．軽量マルチスレッディング向け大域アドレス空間ライブラリ．並列／分散／協調処理に関するサマー・ワークショップ(SWoPP2012)．鳥取．2012 年 8 月．

[2]　Kenjiro Taura, Jun Nakashima, Rio Yokota, and Naoya Maruyama. A Task Parallelism Meets Fast Multipole Methods. Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA). Nov 2012.

[3]　河野 瑛，田浦健次朗．タスク並列モデルを用いた Tree-based AMR の評価．並列／分散／協調処理に関するサマー・ワークショップ(SWoPP2012)．鳥取．2012 年 8 月．