4H-04

Multiple PVASを用いたMapReduceフレームワークの木構造処理による性能評価

本田 舜 †, 佐藤 未来子 †, 並木 美太郎 †
東京農工大学 †

1 はじめに

並列処理はシステムの高速化における重要な要素となっている．特に，データ処理においてはGoogleが考案した分散処理モデルMapReduceが広く用いられている[1]．現状では木構造処理に焦点を当てた研究は少ない．理由として，階層構造がMapReduce処理に適用し辛いといったことが挙げられる．さらに，MapReduce処理を木の深さごとのように複数回行う必要がある場合が多く，MapReduce処理とは別に転送などの冗長時間が生じてしまうという問題点が存在した．

本研究では，MapReduceによる分枝限定法アルゴリズムを用いて[2]，メニーコアコプロセッサであるIntel Xeon Phi向けのMapReduceフレームワークMrPhi[3]上へ実装を行う．実装の際に様々な高速化手法を試み，上記の問題点を解決しつつ，メニーコア上で木構造処理を行う際の高速化の指針を得る．さらに，本研究室内で研究がされている大域仮想空間モデルMultiple PVAS(M-PVAS)[4]を適用したバージョンへの実装も検討し，M-PVASを木構造処理へ適応した場合の性能を評価する．

2 MapReduce処理のための木構造

1章で示した課題の解決のために，以下の手法を提案する．

2.1 木構造

木構造の生成はシーケンシャルな部分が多く，Xeon Phi上よりもホストマシンで生成したほうが高速に完了する．本研究では，ホストマシン上で木の生成を行い，Xeon Phiへ転送することにより探索を行う．そのため，木構造はいくつかのブロックに分割して，ホストマシンとXeon Phi間で転送を行う．

ホストマシン上で木構造の転送処理とXeon Phiでの探索を平行して行えるように，メモリ上のノード配置を最適化する．本研究では，ノードの探索は同じ深さのノードが並列に実行されるBreads-firstを選択し，メモリへもBreads-firstに合わせてノードを配置することにより，ブロックの転送待ち時間を削減する．

2.2 木構造探索アルゴリズム

本研究にてMapReduceで処理するのは先行研究[2]の分枝限定法アルゴリズムとする．まずMap処理で各ノードの探索を行ない，Reduce処理では，探索結果に対して，そのノード以降を探索してもいい結果が得られないと判断した場合に探索を打ち切る．

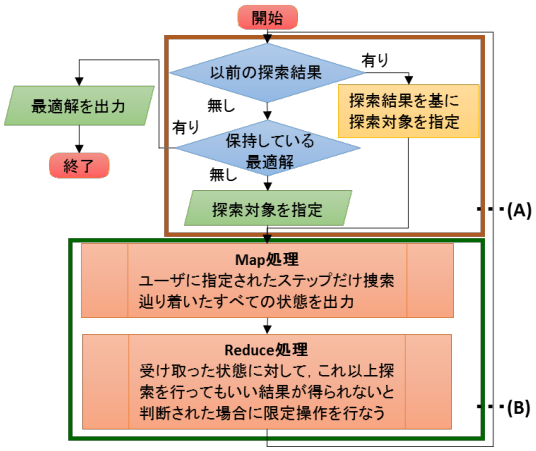

図1: 分枝限定法アルゴリズム

図1において，(A)がホストマシン，(B)がXeon Phiで行われる処理である．(B)の処理は並列に行われるが，(A)の処理はシーケンシャルとなるため，Xeon Phi上で行なうよりもホストマシンで実行したほうが高速となるためである．M-PVASでは，ホストマシンとXeon Phi間で仮想アドレス空間を共有することが可能であり，それを利用して転送時間を削減することにより高速化が期待できる．特に，(A)と(B)間で行われるデータ転送は1つあたり100byte以下であり，そのような小さいデータのやり取りの場合，MPIと比較して大幅な時間削減が可能であるため，入力データを転送せずに直接参照することにより高速化が見込める．

3 システム設計

3.1 全体構成

2章で示したMapReduceプログラムの全体構成を図2に示す．(A)および(B)は図1の記号に対応する．ホストマシン上での木構造の生成，転送とXeon Phiでの探索のオーバラップが行われるように，ホストマシン上で木構造生成を行いながら(①)，埋まったメモリブロックから逐一送信する(②)．この時，Xeon Phi上でホストマシンから木を受け取り次第探索を行なう(③)ことでオーバラップを実現する．

3.2 木構造

木構造の生成のために，まず予め指定したブロックサイズ分のメモリ領域を確保し，領域内にノードを先

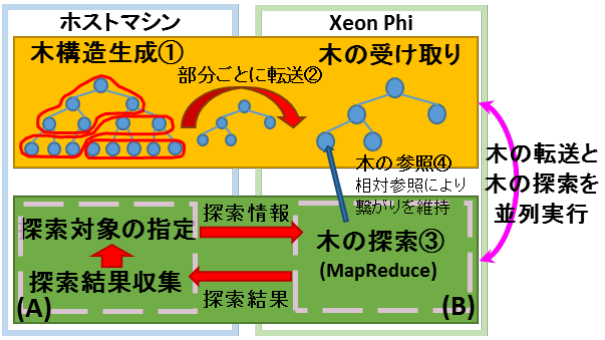

図 2: システムの全体構成

頭から配置していく．そのブロックが埋まった時点でMPI を用いてブロック全体を転送し，次のブロックのメモリ領域を確保する．

ノード間の接続はポインタによる絶対参照ではなく相対参照にすることにより，異なるマシン上でも探索を行える木構造を実現する (図 2 内④)．木のノード間を相対参照するために，全ノードに対して生成順に番号を付与し，子ノードの番号を所持することにより参照を行う．前述のブロックのメモリ領域について，転送の際に先頭アドレスを配列に保持しておき，以下のコードにより参照先ノードのポインタを取得する．

N: 参照先のノード番号
B: ブロック内のノード数
arr: 全ブロックの先頭アドレスを格納した配列
%: 除算の余りを求める演算子

$$\mathrm{arr}[N\ /\ B][N\ \%\ B]$$

3.3 MapReduce 処理

MapReduce 処理においては，木の探索を Map 処理，探索先に限定操作を行うかの判別を Reduce 処理が行い，それらをメニーコアで並列処理する．MapReduce 処理本体はすべて Xeon Phi 上で行われる．入力として探索対象のノード番号を受け取った Map ワーカーは，そのノードから探索を開始し子ノードを出力する．Reduce ワーカーは，それらの子ノードに対して，以降を探索していい結果を得られるかを判断する．もし得られないと判断された場合はフラグを付けてホストマシンへ送信する．

オリジナルの MrPhi ではホストマシンと Xeon Phi 間の入出力を MPI を用いて行っており，この部分を M-PVAS を用いてやり取りすることで転送オーバヘッドを削減する．

4 実装と評価

2 章および 3 章で示した設計を基に実装を進めている．実装が完了した部分のうち，ホストマシンと Xeon Phi の連携に関する部分について評価を行った．木の深さが 25(約 1.25GB) の最適化問題に対して，すべての処理を Xeon Phi 上で行った場合と，データ処理をホストマシンで行い MapReduce 処理を Xeon Phi で行った結果を図 3 に示す．

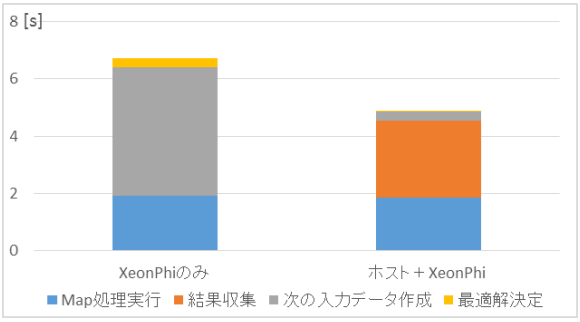

図 3: 提案手法を用いた性能評価結果

図 3 より，両マシン間連携を行うことにより入力データ処理の時間を削減することができ，この部分の提案手法については高速化に有効であるという指針を得た．2 章で触れた M-PVAS を用いたバージョンの実装については本原稿執筆時点で実装を進めている段階であるが，現状の結果では転送によるオーバヘッドが大きいため，M-PVAS により転送時間を削減できればオリジナルに対する優位性を実証できる．実装が完了した段階で，M-PVAS を適用した場合の高速化への貢献度を検証する．また，他の提案手法についても実装を急ぎ，上記手法と同様に評価を行なう．

5 おわりに

本研究では，MrPhi 上へ分枝限定法の MapReduce アプリケーションを設計に際して，木構造の生成，転送と Xeon Phi での探索のオーバラップや双方のマシンによる連携など，様々な並列化手法による高速化を画策した．また，マシンの連携に関する評価を行い，本手法が高速化に対して有効であるという結果を得た．

今後の課題として，提案手法を適用した場合についても評価を進め，メニーコア上での木構造処理に対する高速化の指針を得る．また，M-PVAS を適用した MrPhi 上での性能の検証を完了し，M-PVAS による木構造処理の有効性を確かめる．

参考文献

[1] J. Dean and S. Ghemawat, "MapReduce: Simplied Data Processing on Large Clusters", *Proceedings of 6th Symposium on Operating Systems Design and Implementation*, Volume 6, Oct. 2004, pp. 137-149.

[2] 中川遼, 『MapReduce による列挙木探索手法の提案と評価』, 第 *10* 回情報科学ワークショップ, Session E-7, Sep. 2014, pp. 1-9.

[3] M. Lu, et al., "MrPhi: An Optimized MapReduce Framework on Intel Xeon Phi Coprocessors", *EEE Transactions on Parallel and Distributed Systems*, Volume PP, No. 99, 2014, pp.1-14.

[4] M. Sato, et al., "Design of Multiple PVAS on Infini-Band Cluster System Consisting of Many-core and Multi-core", *In Proceedings of the 21st European MPI Users' Group Meeting*, Sep. 2014, pp.133-138.