LMNtal処理系SLIMのマルチスレッド化による最短経路問題のデータ並列求解

青山龍一† 上田和紀††

1. はじめに

LMNtal[1] は階層グラフの書き換えに基づく並列言語モデルである. LMNtal の実行は最初にプログラムをコンパイルして中間命令列を生成し, その中間命令列を実行時処理系によって解釈実行することで実行が行われる.[2] SLIM[3] は従来のものよりメモリの使用量の削減や実行時間の削減を考慮された実行時処理系の一つであり, 現在も盛んに開発されている.

本研究では, SLIM のさらなる実行速度の高速化の方法として通常実行におけるデータ並列に基づく並列化を試みた. そして, グラフの最短経路問題[4] を例に挙げ, このプログラム自体どのように並列化できるかを考え, さらに SLIM においてマルチコアを使い, 実際に並列化実行が行うことができるのかを実装し確かめた. また実際並列化することによってプログラムの高速化がどのくらいなされているかを検証した.

2. LMNtalとその処理系

LMNtal は階層グラフ書き換えに基づく並列モデル言語であり, 構成要素としてアトム, リンク, 膜, ルールからなる. リンクはアトム同士をつなげることができ, 膜はアトムやルールの多重集合を構成するのに使われ, これによって階層構造をなすことができる.

LMNtal の実行の流れとして LMNtal で書かれたプログラムはコンパイラによって中間命令列を生成し, 実行時処理系によって処理が実行される二段階の方式になっている. この流れを図 1 に示した. LMNtal の実行時処理系は開発当初 Java 言語によって実装されていたが, 2009 年に C 言語で実装された SLIM が開発された. SLIM の開発によって従来の実行時処理系より大幅に実行時間の短縮を成し遂げることができた.

3. 例題プログラム:最短経路問題

今回並列化にあたり, 例題として使用した最短経路問題のプログラムを説明する.

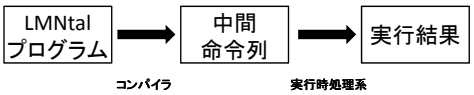

図 1 LMNtal 実行の流れ

shortestpathAll2all.lmn☆は与えられたデータからそれぞれの駅の移動によってかかる時間の最小経過時間を求める最短経路問題のプログラムである. 構成するアトムとして cost と e がある. それぞれの引数として第 1, 2 引数には駅名が第 3 引数にはその駅の間でかかる時間の数値を保持する.

このプログラムはルールの適用によってアトム cost をアトム e をもとに書き変えていく. ルールは簡単に記述すると, 図 2 に示す条件を満たしたときに書き換えられる. LMNtal プログラムにおいては最短経路問題をこの明瞭かつ簡潔な一つのルールで解くことができる.

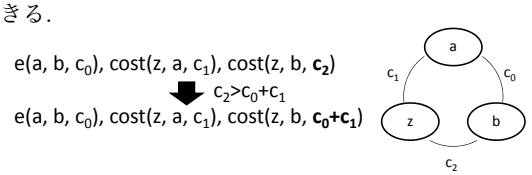

図 2 ルールの詳細

また, アトム e は初期に与えられるデータとなっている. ここで駅名をノード, かかる時間をエッジの重みとしてとらえると, アトム e のデータによってグラフを記述することができる. 記述したグラフを図 3 に示す.

このプログラムは駅間の時間を短いものへと短縮するため, 他のアトム cost が変化すると別のアトム cost に答えが影響してくる. よってルールの適用が何度も繰り返されるため, 単純にはプログラムの実行は終わらない. よって並列化することで時間の短縮につながることが最も期待されているプログラムの一つとなっ

† 早稲田大学大学院基幹理工学研究科情報理工学専攻
†† 早稲田大学理工学術院情報理工学科

☆ http://www.ueda.info.waseda.ac.jp/lmntal/demo/

ている.

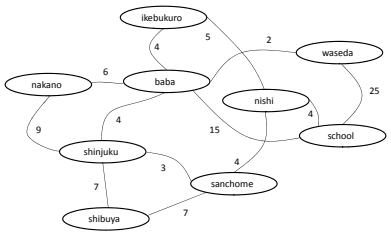

図 3　アトム e のデータから作られるグラフ

4. 並列化方針

今回並列化を行う方針として, 存在するルールをそれぞれ生成したスレッドに割り当てることで, 並列に実行させるようにした. よってプログラムによっては並列で実行できるようにプログラムを変換する必要がある. さらにルール同士同じアトムを書き換えるようにすると衝突が起こってしまい, 並列効果が出せなくなるので, 各ルールは一切干渉なく実行できるように考慮されている.

図 4 を使って説明すると, 変換によってお互いのスレッドに関して一切干渉しないように適用させるアトムを完全に分けるようになっている. 最終的には先ほど変換した逆の規則で逆変換することで本来の解を出すことができる. 今回の最短経路問題のプログラムも扱いやすいように変換を行った.

今回この並列化方針に従って SLIM の並列化を行った.

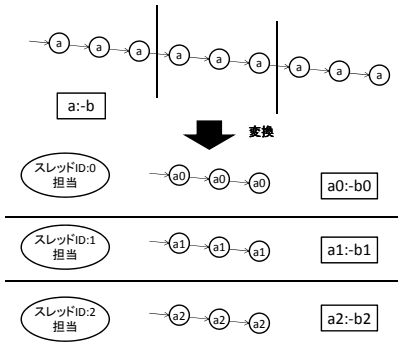

図 4　プログラムの変換

5. 評 価 実 験

今回評価した内容は変換後の逐次実行と並列実行の比較である. 実験環境は表 1 に示す. ちなみにエッジの数は 1 つのノードから平均 3 本のエッジが出るように数を合わせて測定を行った.

実行結果より作成されたグラフを図 5 に示す. 図 5 より変換後のプログラムの逐次実行においての計算量は理想的ではないが, コア数以上の並列効果がでていることが分かる. これは SLIM の設計上, 逐次実行の場合においてルール同士が干渉しあわなくても他のルールの適用による影響を考慮することにある. つまり, 並列に動かすために書き換えたプログラムがむしろ逐次実行のにおいての実行時間の増加につながったということである.

さらに, プログラムの変換によっても実行時間が短縮さている. なぜなら一本あたりのルールが扱うアトムの範囲が減ったため, 探索の範囲が狭まったことが理由として挙げられる.

表 1　実験環境

CPU	AMD Opteron 6176SE
コア数	48 (12 × 4 Processors)
メモリ	256GB
LMNtal version	1.21.20111226
SLIM version	2.2.2

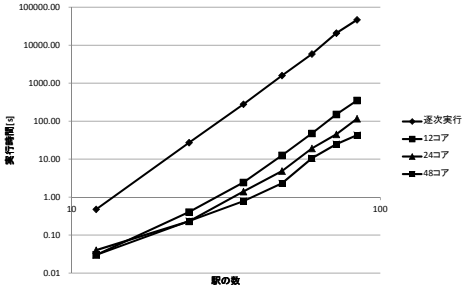

図 5　逐次実行と並列実行の比較

6. まとめと今後の課題

本研究は SLIM の並列化を実際に実装し, 最短経路問題のプログラムを動かして実行時間の削減を行うことができた. しかし, 今回の並列化は汎用性に関しては劣っていると考える. よってプログラムに依存せず動かせる SLIM の並列化をしなければならない.

また, SLIM の並列化に伴い, 最短経路問題のプログラムを動かすことによってルールの適用順序に関しての設計上の問題が生じてきたことが明白に分かり, 今後 SLIM においてのルール適用のアルゴリズムを再度見直さなければならないことも課題の一つである.

参 考 文 献

1) 乾敦行, 工藤晋太郎, 原耕司, 水野謙, 加藤紀夫, 上田和紀 : 階層グラフ書換えモデルに基づく統合プログラミング言語 LMNtal, コンピュータソフトウェア, Vol. 25, No.1, 2008
2) 村山敬, 工藤晋太郎, 櫻井健, 水野謙, 加藤紀夫, 上田和紀 : 階層グラフ書換え言語 LMNtal の処理系, コンピュータソフトウェア, Vol. 25, No. 2, 2008
3) 石川力, 堀泰祐, 村山敬, 岡部亮, 上田和紀 : 軽量な LMNtal 実行時処理系 SLIM の設計と実装, 情報処理学会第 70 回全国大会, 2008
4) A.Grama et al.: Introduction to Parallel Computing (2nd ed.), Cambridge Univ. Press, 2003.