卒業研究報告書

題 目

BSPモデル上での MSTを作る並列アルゴリズム

指 導 教 員

石 水　隆 助手

報 告 者

02–1–47–119

北 脇　真 斗

近畿大学理工学部情報学科

平成 18 年 2 月 10 日提出

概要

本研究では、BSP(Bulk-Synchronous Parallel) モデル [1][2] 上で MST(Minimum Spanning Tree) を作る効率の良い並列アルゴリズムを提案する。

また、その計算量を実験的に評価するため BSP モデル上でのアルゴリズムの実行をシミュレートするプログラムを作成し、その実行時間を測定する。BSP モデルは分散メモリ型非同期式並列計算モデルであり、従来型の PRAM (Parallel Random Access Machine) モデル [3] とは異なり、通信および同期にかかる時間を考慮したモデルであり、クラスタ (Cluster) 処理およびグリッド (Grid) 処理による並列化に対応したモデルとして注目されている。PRAM モデル上で作成した並列アルゴリズムは BSP モデル上では効率良く実行できるとは限らない。そのため、多くの問題に対して BSP 上で効率良く動く並列アルゴリズムが求められている。

本研究で提案するアルゴリズムは、CREW PRAM 上で MST を解くアルゴリズムである Sollin[4] のアルゴリズムをベースにしている。一部のプロセッサに対してメッセージ送信が集中して行われるため、通信時間の増加が起こる。そこで本研究で提案するアルゴリズムでは、この問題を避けるためにプロセッサ間に 2 分木構造を構築して通信を行うことにより、プロセッサ間の負荷分散を得ている。

目 次

1 序論

1.1 本研究の背景

1.1.1 並列計算機 [6]

1 代のプロセッサを用いて一時点に 1 個のステップを実行するアルゴリズムを逐次アルゴリズム (sequential algorithm) という。一方、複数のプロセッサを用いて同時にいくつかの操作を並列に実行できるアルゴリズムを並列アルゴリズム (parallel algorithm) という。並列アルゴリズムは 1 つの問題に対して同時に 2 つ以上のプロセッサが動作できる並列計算機 (Parallel Computer) により実行される。

並列計算機は複数のプロセッサを同時に動作させる方式で、計算機自身の歴史に迫るほどの長い研究開発の歴史を持つ。複数のプロセッサを用いるのには主として以下の 3 つの目的がある。

1. 同時に動作させることにより単一ジョブの処理を高速に行う (高速処理)
2. 同時に動作させることにより、高い信頼性を得る (高信頼性)
3. 複数のプロセッサでメモリ、ディスク、入出力を供給することにより多くのユーザに対して効率よく資源を運用し、コストを低減する (資源の共有)

この内高信頼性を目的とするシステムは多重化システムと呼び、普通は並列計算機の中には入れない。また資源を共有を主たる目的とするシステムはどちらかというと分散システムの中に入る。このため並列計算機といった場合、単一ジョブの高速化を主たる目的としたシステムを指す。多数のプロセッサを使うことでシステム全体のダウンを防いだり複数のユーザ複数のジョブを動かすことにより計算資源の効率的な運用を行うことができることも、ある種の並列計算機の利点の 1 つである。しかし独立なジョブの同時実行しかできず複数のプロセッサの並列処理により単一ジョブの高速化ができないシステムの場合は並列計算機と呼ばないのが普通である。　並列計算機は全てのプロセッサが共通したメモリに対して読み書きを行い、プロセッサ間通信は共有メモリを通じて行う共有メモリ型並列計算機と、それぞれのプロセッサが局所メモリを持ち、プロセッサ間の通信はネットワークを通じて行う分散メモリ型並列計算機に分類される。プロセッサ数の増加に伴い 1 つの共有メモリに全てのプロセッサを接続することは困難となるため、現在プロセッサ数の多い並列計算機は分散メモリ型が主流になっている。また複数の計算機をネットワークで繋ぎ、それ全体を 1 台の仮想的な並列計算機として用いるクラスタ（Cluster) 処理やグリッド (Grid) 処理も幅広く行われている。

1.1.2 並列処理の必要性 [7]

より速い計算機の必要性は計算機が考案された時から止むことがない。新しい応用は現在の計算機をその限界まで利用しようとしている。これまで計算機が 1 秒間に実行できる処理数が過去約 50 年間におよそ 2 年半ごとに 2 倍になっている。　最も簡単なものから最も高性能なものまで今日存在する大多数の計算機は概念的にはお互いに非常によく似ている。それらのアーキテクチャ(構造) と演算は J.Neumann が考案し 1940 年代に定式化された設計原理に従っている。その操作の流れは非常に単純であり基本的には次のようである。制御装置 (control unit) は命令とオペランド (operand) を記憶装置 (memory unit) から取ってきて、それらを処理装置 (processing unit) に送る。その結果がメモリに戻される。この動作系列が各命令ごとに繰り返される。そこでは処理装置、制御装置、記憶装置がそれぞれただ 1 つであり一度にただ 1 つの命令を実行する。

並列処理を用いると状況が一変する。並列計算機は複数の処理装置またはプロセッサからなり、ある問題を解く際その問題の異なる部分問題を同時に解き全体の結果を導くのに協力し合う。使用されるプロセッサ数は数個から数万個にまでわたる。結果的にこれまでの単一プロセッサの計算機では問題を解くのにかかる時間よりも非常に少ない時間で解くことができる。このアプローチは次のような理由から魅力的である。

- 多くの問題に対して並列的な解法の方がごく自然である。
- 並列処理では問題または問題クラスを解くのに最も適したアーキテクチャを選ぶことが可能である。

また近年マイクロプロセッサがより安くなり、それらを互いにつなぎ合わせる技術が進歩したため非常に多数のプロセッサを持つ汎用コンピュータを作ることが可能になり、また実用的になってきた。並列コンピュータを利用するハードウェアやアルゴリズムや理論モデルの開発が爆発的に活動が行われてきている。将来、並列性がさらに使われていくことは明らかなようである。

1.1.3 MST(Minimum Spanning Tree) 問題

本研究では代表的なグラフ問題である MST(Minimum Spanning Tree) を対象する。

頂点数 n の MST 問題に対し、Prim は RAM 上で $O(n^2)$ 時間の逐次アルゴリズム [5] を提案した。Sollin は CREW-PRAM 上で $O(log^2 n)$ 時間 n^2 プロセッサの並列アルゴリズム [4] を提案した。本研究では BSP モデル上で $O(ng + L\log^2 n)$ の並列アルゴリズムを提案する。

2　準備

2.1

2.1.1　PRAM(Parallel Random Access Machine)[3]

並列ランダムアクセス機械 (PRAM,Parallel Random Access Machine) は無限個のメモリセルから成る共有ランダムアクセスメモリ M と、それにつながっている p 個の汎用プロセッサ $P_1, P_2...P_p$ からなる。各プロセッサが自分自身の計算のために私有 (またはローカル) メモリをもつがプロセッサ間のコミュニケーションは全て共有メモリを介して行われる。特に指定しない限りアルゴリズムの入力は最初の n 個のメモリセルにあり、その入力は 1 番目のセル (または初めの一連セル) に置かれていると仮定する。全てのプロセッサは同一のプログラムを実行するが各プロセッサは自分自身のプロセッサ番号 (index) に応じて異なった仕事を実行するように指示してもよい。共有メモリ型であるため、PRAM ではデータの局所性を無視でき通信のコストが発生しない。また PRAM は全てのプロセッサが 1 命令ごとに同期を行う細粒度同期式 (Fine-Grain Synchronization) モデルであり同期のコストも発生しない、さらに全ての命令は演算命令、メモリ追加命令、入力命令等、その種類に関係なく 1 単位時間で実行できる一様コストを仮定している。PRAM は以上の非常に強い仮定を設けている単純なモデルであるため、PRAM 上でアルゴリズムの設計・解析を行いやすい。

しかし、PRAM 自体の実現は困難である。また現在主流となっている分散メモリ型並列計算機やクラスタ処理による仮想並列計算機上では共有メモリを仮定している PRAM のアルゴリズムは必ずしも効率よく実行できるとは限らない。　また、これらの計算機では通信および同期に非常に大きなコストがかかるため、これらを正確に評価できる新たな計算モデルが必要である。

2.1.2 BSP[1][2]

プロセッサ能力の向上に伴い、PRAM、分散メモリ型並列計算モデルでは重点を置かれていなかったプロセッサ間の通信コストがプロセッサ内部の演算コストとともに、並列計算のコストにおける重要な要素となってきた。また、同時に多くのプロセッサが大部分の処理を他のプロセッサと同期せずに処理を行う非同期処理も主流になってきた。これらの特徴を持つ最近の並列計算機に対しては、PRAM を代表とする従来の並列計算モデルでは、アルゴリズムの評価を正確に行うことが困難であり、これらの特徴に対応した新しい並列計算モデルが望まれていた。BSP(Bulk-Synchronous Parallel) モデルは通信コストを同期周期、通信命令実行時間を表す L,g という 2 つのパラメタにより表すことが可能になっている。また同期機構を仮定することにより、非常に緩いどうきの処理に対応可能になっている上記の要求に対応したモデルである。　BSP モデルは valiant によって提案された非同期式並列計算モデルであり、以下の構成要素からなる。

- 局所メモリを持つ複数のプロセッサ (プロセッサ数を p とし、各プロセッサを $P_i(1 \leq i \leq p)$ で表す。)
- プロセッサ間の 1 対 1 メッセージ通信を行う完全結合網
- プロセッサ間の同期を実現するための同期機構 BSP モデル上での並列アルゴリズムは各プロセッサが実行するプログラムにより表される。各プロセッサはスーパーステップの列からなる。各スーパーステップは内部計算命令の列からなる内部計算フェーズと、送信命令、受信命令の列からなる通信フェーズで構成されており、各プロセッサはスーパーステップの命令を終了後、プロセッサ間でバリア同期を取り、次のスーパーステップの実行に移る。バリア同期とは協調して動作する多数のプロセッサの歩調を合わせることを目的とした同期プリミティブである。バリア同期を実行して同期を取る場合、全てのプロセッサがバリアに到達するまでどのプロセッサも実行を継続できず封鎖される。メッセージの受信については各スーパーステップ中の通信フェーズで送信されたメッセージは同一のスーパーステップの通信で受信されるが、そのメッセージはその次のスーパーステップ以降でしか利用できないと仮定する。

  BSP モデルは以下の 2 つのパラメタにより、具体的なネットワーク構造やメッセージ配送の仕組みを抽象化している。

  - L :バリア同期周期
  - $g(\leq L)$:1 個の送信命令または受信命令の実行に必要な時間

3　研究内容

本研究で提案するアルゴリズムは、CREW PRAM 上で MST を解くアルゴリズムである Sollin[4] のアルゴリズムをベースにしている。Sollin のアルゴリズムは以下の 3 つのステップから成る

- (1). 各頂点の最小の重みを持つ辺を求め、
  その辺を親とする有向グラフを作る
- (2). 1 で作成した有向グラフに対して各頂点の根を求める
- (3). 各頂点から根に対して辺の情報を送り、根以外の頂点を削除する

以上のステップを頂点数が 1 個になるまで繰り返す。　BSP 上で MST 問題を解くには、各頂点に対して 1 台のプロセッサを割り当て、各プロセッサが並列に上記の 3 ステップを繰り返せばよい。
しかし、Sollin のアルゴリズムをそのまま BSP 上で実行した場合、ステップ (3) において根となる頂点を担当するプロセッサに対してメッセージ送信が集中して行われるため、通信時間の増加が起こる。そこで本研究で提案するアルゴリズムでは、この問題を避けるためにプロセッサ間に 2 分木構造を構築して通信を行うことにより、プロセッサ間の負荷分散を得ている。また、その計算量を実験的に評価するため BSP モデル上でのアルゴリズムの実行をシミュレートするプログラムを作成し、その実行時間を測定する。

4 結論

本研究では、BSP モデル上で MST 問題を解く並列アルゴリズムを提案した。

本研究で提案したアルゴリズムは、メッセージ 1 つ辺りの通信時間が大きい環境でも高速に MST 問題を解くことができる。

本研究のアルゴリズムにおいて高速化が得られたのは、プロセッサ間で 2 分木構造を構築して通信を行うことで負荷分散を行い、少数のプロセッサへの負荷の集中を避けたからである。従って BSP モデル上で効率の良いアルゴリズムを設計するためには、負荷を考慮して行う必要があると言える。

本研究で提案するアルゴリズムの計算量、および Sollin のアルゴリズムをそのまま BSP 上で実行した場合の計算量を表 1 に内部計算時間、通信時間、同期時間をシュミレートプログラムで計測した結果を表したグラフを図 1、図 2、図 3 に示す。Sollin のアルゴリズムに対して、本研究で提案したアルゴリズムは同期回数は増加しているが、送受信メッセージ数は減少している。

よって、本研究で提案したアルゴリズムはメッセージ 1 つ辺りの通信時間が大きい環境では Sollin のアルゴリズムよりも高速に実行できる。

表 1: 各アルゴリズムの計算量

Sollin	$O(n^2g + L\log n)$ 時間	n プロセッサ
本研究	$O(ng + L\log^2 n)$ 時間	n プロセッサ

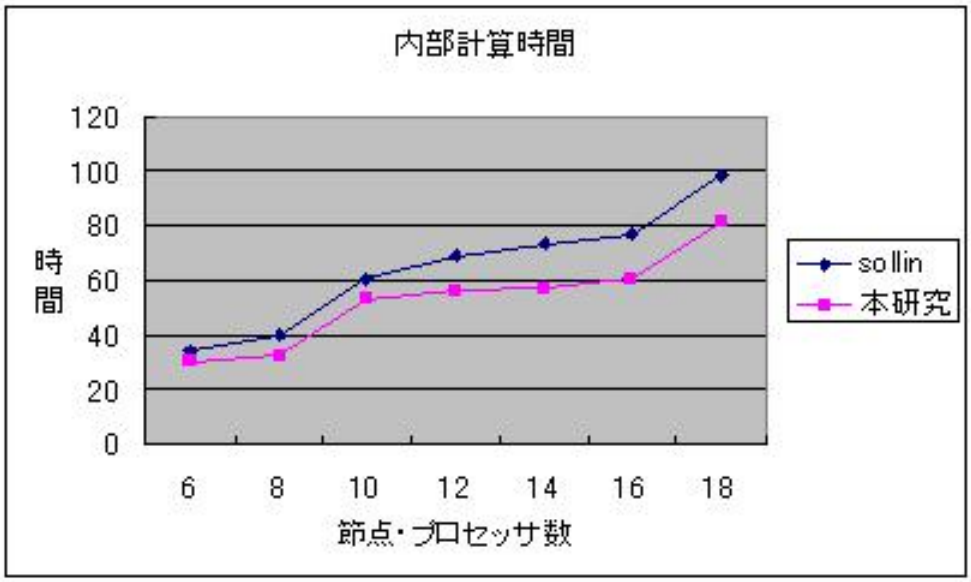

図 1: 内部計算時間

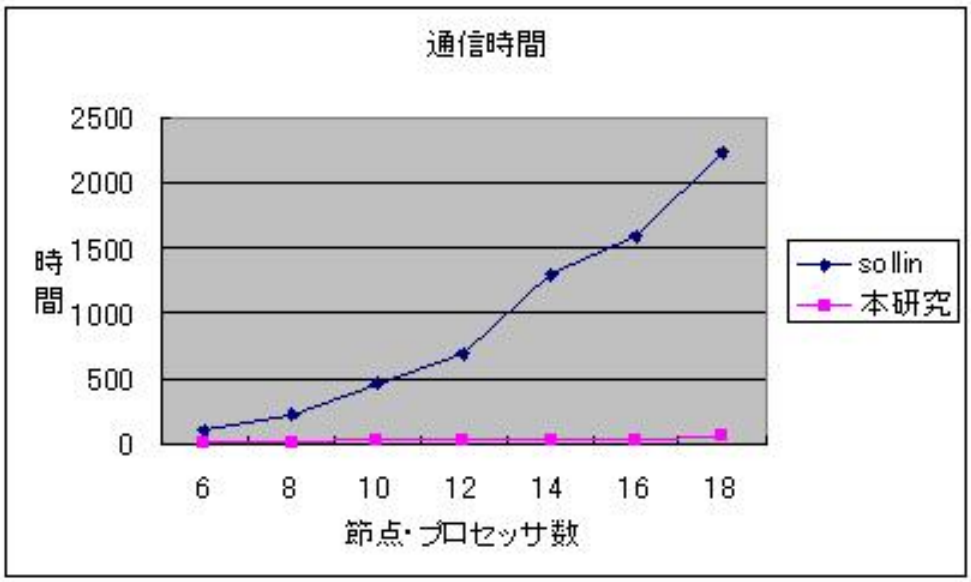

図 2: 通信時間

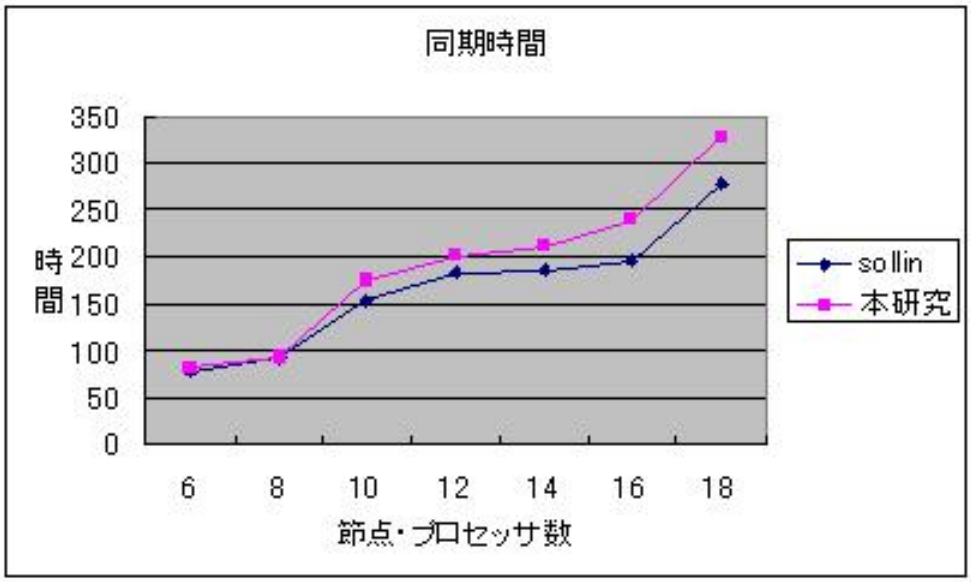

図 3: 同期時間

5 謝辞

本研究報告書を作成を作成するに当たり、毎日研究に遅くまで付き合って親切に教えてくださるなど、私達のために多大なご指導、ご支援を承り、先生の貴重な時間を割いて下さいました。石水隆先生には深く感謝の意を表す次第であります。

参考文献

[1] L.G.Valiant, "A Bridging Model for Parallel Computation, ", Communications of the ACM, Vol.33, No.8, pp.103–111 (1990).

[2] 石水隆, 藤原暁宏, 井上美智子, 増澤利光, 藤原秀雄:選択問題を解く BSP モデル * 上の並列アルゴリズム　電子情報通信学会論文誌 D-I Vol.J82-D-I No.4 pp.533–542 (1999).

[3] J.JáJá, "An Introduction to Parallel Algorithms," Addison-Wesley Publishing Company (1992).

[4] C.Berge and A.Ghouila-Houri,"Programming,Games and Transportation Networks,"John Wiley (1965).

[5] R.C.Prim, "Shortest connection networks and some generalizations, " Journal of Bell System Tech, Vol.36, No.6, pp.1389–1401, 1957.

[6] 天野英晴 :" 並列コンピュータ "、昭晃堂 (1996).

[7] 渋沢進 :" 並列分散処理入門 "、培風館 (1998).