卒業研究報告書

題 目

BSPモデル上での
パラレルクイックソート

指 導 教 員

石 水　隆 助手

報 告 者

02–1–47–012

吉 村　秀 明

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 [7] 上で高速にソーティングを行う並列アルゴリズムを提案する。BSP モデルとは分散メモリ型の並列計算モデルであり、PRAM(Parallel Random Access Machine)[7] と異なり並列アルゴリズムの通信および同期にかかるコストを考慮したモデルである。このため、BSP モデルはより現実の並列計算機に近いモデルとして注目されており、現在様々な問題に対して BSP モデル上で高速に実行できる並列アルゴリズムが求められている。しかし、従来の PRAM のアルゴリズムは通信が考慮されていないため、これを BSP モデル上で実行させても効率よく実行できるとは限らない。従って、通信や同期を考慮した BSP モデル用のアルゴリズムを設計する必要がある。本研究では基本的な問題であるソーティングに対して、BSP モデル上で効率良く解く並列アルゴリズムを提案する。本研究で提案するアルゴリズムはクイックソートをベースとしている。クイックソートはデータ内のある値を基準値とし、データをその基準値以下のデータからなる部分データと基準値以下のデータからなる部分データに分割し、各部分データを再帰的に分割していくことによりソーティングを行う。分散メモリ型並列計算モデルである BSP モデルでは、あるプロセッサが持つデータを他のプロセッサが使うためには、プロセッサ間で通信および同期を行わねばならない。しかし、一般的に 1 メッセージあたりの通信時間 g および同期時間 L は内部計算時間に比べて極めて大きいとされる。従って高速なアルゴリズムを設計するには通信メッセージ数および同期回数を減らすことが必要となる。BSP 上でクイックソートを行う場合、各ループでデータを分割後、あるプロセッサから他のプロセッサに対して部分データの送信が行われる。本研究では多くの場合において計算量の大きな部分を占める同期にかかる時間に着目し、同期回数を減らすことにより計算量の改善を図る。本研究で提案するアルゴリズムは、各ループでデータを分割する際に k (> 2) 個の部分データに分割することにより同期回数を減少させている。

目 次

1 序論

1.1 並列アルゴリズム

地球規模の気象シミュレーションや天体の軌道計算など、計算量の大きな問題を短時間で解く必要のある分野は多岐に渡っている。これらの問題に対して、従来の 1 台のプロセッサから成る逐次計算機を用いた逐次処理では非常に大きな時間が掛かる。このため、これらの問題を解く手法として、複数のプロセッサを持つ並列計算機 (Parallel Computer) による並列処理 (Parallel Processing) が現在注目されている。複数のプロセッサが協調してデータを処理することにより、問題を短時間で解け、またより複雑な問題を解くことができるようになる。しかし、並列処理を行うためには、プロセッサ間のデータのやり取りやメモリへのアクセス、プロセッサ間の同期等、並列特有の問題を解決せねばならない。このため、従来の逐次処理で用いられてきた逐次アルゴリズムをそのまま並列処理に用いることはできず、並列処理専用のアルゴリズム、すなわち並列アルゴリズムが必要となる。そのため、現在様々な分野で、高速に処理を行う並列アルゴリズムが求められている。

1.2 並列計算機

並列計算機は複数のプロセッサを持ち並列処理を行うことができる計算機である。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、プロセッサ間の通信はメモリを通して行う共有メモリ型並列計算機 (Shared Memory Parallel Computer) 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 (Distributed Memory Parallel Computer) に大別される。プロセッサ数の増加に従い、1 つの共有メモリに全てのプロセッサを繋ぐことは困難となる。このため、現在、プロセッサ数の多い並列計算機では分散メモリ型が主流となっている。また、複数の計算機をネットワークで繋ぎ、それ全体を仮想的な計算機として扱うクラスタ (Cluster) 処理やグリッド (Grid) 処理も幅広く行われている。

1.3 並列計算モデル

並列アルゴリズムの設計・解析は、並列計算機を抽象化した並列計算モデル上で行われる。代表的な並列計算モデルとして、PRAM(Parallel Random Access Machine)[5], Mesh[5], Hyper-cube[5], BSP (Bulk-Syncronous Parallel) モデル [7], CGM (Coarse Grain Multi-Computer)[2] などがある。

PRAM は共有メモリ型並列計算モデルであり、全ての演算が 1 単位時間で行われる、1 命令毎に同期が取られる、通信のコストが一切発生しない、等

の仮定が設けられた理想的なモデルである。このため PRAM 上でのアルゴリズムの設計・解析は比較的容易に行うことができる。しかし、PRAM 自体の実現は困難であり、PRAM 上で設計したアルゴリズムは現実の並列計算機では必ずしも効率良く実行できるとは限らない。このため、現在主流となってきた分散メモリ型並列計算機に対応するモデルとして注目されているのが BSP モデルである。BSP モデルは分散メモリ型並列計算モデルであり、通信のオーバヘッドや同期のオーバヘッドを考慮することができるモデルである。そこで本研究ではモデルとして BSP モデルを採用し、この上で高速に実行できる並列アルゴリズムの設計を行う。

1.4 ソーティング

ソーティングは基本的な問題であり、様々な分野で広く用いられる。このため、並列計算機上で高速にソーティングを解くことができる並列アルゴリズムを開発することは重要な課題である。サイズ n のデータに対し、逐次アルゴリズムでは、クィックソート [3] やマージソート [3] を用いて $O(n \log n)$ 時間でソーティングを行うことができる。また Reischuk は CREW PRAM 上で p 台のプロセッサを用いて $O(\frac{n \log n}{p} + \log n)$ 時間でソーティングを行う確率的並列アルゴリズムを提案した。Cole は CREW-PRAM 上で p 台のプロセッサを用いて $O(\frac{n \log n}{p} + \log n)$ 時間でソーティングを行うアルゴリズムを提案した [1]。本研究では、BSP モデル上で p プロセッサを用いて任意の整数 $k(2 \leq k \leq p)$ に対し $O(\frac{n \log n}{p} \log \frac{n}{p} + n(\log k + g) + L\frac{\log p}{\log k})$ 時間でソーティングを行う決定性並列アルゴルズムを提案する。ここで g は 1 メッセージ辺りの送受信時間、L は同期時間である。

1.5 本報告書の構成

次項の 2 節では PRAM や BSP、ソーティングについてより詳しく述べると供に、本研究で提案する改良を加えたアルゴリズムを詳しく記述する。3 節にはその実行結果を示し、4 節で本研究における結論や今回の研究で明らかになった今後の課題などを記載し考察も行う。

2 準備

2.1 PRAM(Parallel Random Access Machine) モデル

PRAM(Parallel Random Access Machine) モデル [5] は複数のプロセッサがメモリを共有したモデルであり共有メモリ型並列計算モデルと言われる。図 1 に PRAM の概念図を示す。このモデルで並列アルゴリズムを実行すると、各プロセッサは入力データの読み出しや書き込み、最終結果の書き出しの為に共有メモリにアクセスする。PRAM の各プロセッサは共有メモリ上の任意の位置にあるメモリセルに対して 1 単位時間で読み書きでき、また全ての演算は 1 単位時間で行うことができる。また、PRAM は細粒度同期式 (Fine Grain Synchronicatoin) であり、1 単位時間ごとに全てのプロセッサで同期が取られる。プロセッサ間の通信は共有メモリを通じて行われる。PRAM はこのように通信や同期にかかるコストが一切発生しないなどの理想的な仮説が設けられている為、並列アルゴリズムの設計を容易なものにし、問題の並列性をある程度理論的に検証することを可能にしている。また、PRAM は他の並列計算機モデルの基礎となることも多く PRAM を対象に設計されたアルゴリズムは数多く存在する。しかし、PRAM は理想的なモデルであるため現実とのギャップがあり、これらのアルゴリズムを実行できる効率の良い並列計算機は実際には存在しない。

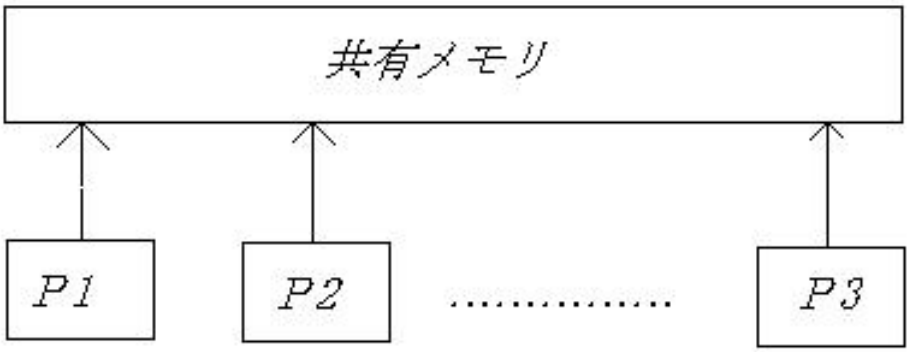

図 1: PRAM(Parallel Random Access Machine) モデル

2.2 BSP(Bulk-Synchronous Parallel) モデル

従来型の PRAM は通信や同期のコストを考えない理想的なモデルであった。初期の並列計算機ではプロセッサの処理能力は低く、プロセッサの内部演

算時間に比べてプロセッサ間の通信はさほど考慮されていなかった。しかし、ここ数十年でプロセッサの処理能力は急激に向上したため、通信や同期にかかるコストというのが処理時間において大きなウエイトを占めるようになってきた。このため近年、PRAM モデルは現実の並列計算機とは程遠いモデルであると言われるようになってきた。このような理由により、Valiant により BSP(Bulk-Synchronous Parallel) モデル [7] が提案された。BSP モデルは非同期式分散メモリ型の並列計算モデルである。BSP は局所メモリを持つ複数がプロセッサとそれらを結びつけるネットワークおよびプロセッサ間でバリア同期を取るための同期機構からなる。プロセッサ間の通信はネットワークを通して 1 対 1 でメッセージ交換をすることにより行われる。なお、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。図 2 に BSP の概念図を示す。。BSP モデルは通信遅延や同期時間等を表す為に以下のパラメタを持つ。

- p : プロセッサ数
- g : 1 つのメッセージを送信あるいは受信するのにかかる時間
- L : バリア同期を取るのにかかる時間　(通信遅延時間)

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。各プロセッサが実行するプログラム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成る内部計算フェーズと、送信あるいは受信命令の列からなる通信フェーズで構成されており、各プロセッサは割り当てられたスーパーステップを非同期に実行する。スーパーステップの命令終了後、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。あるスーパーステップで各プロセッサが各々w 個の内部計算命令と各々h 個の通信命令を実行する場合、そのスーパーステップの時間計算量は $O(w + gh + L)$ となる。以下ではプロセッサを $P_0, P_1, P_2 \cdots P_{p-1}$ と表記する。

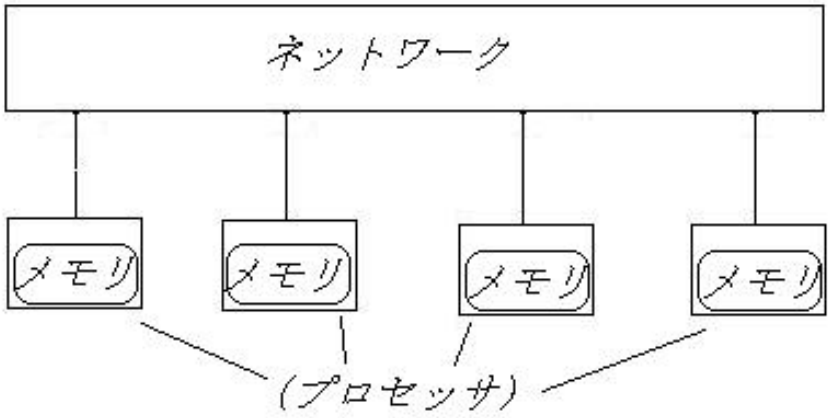

図 2: BSP(Bulk-Syncronous Parallel) モデル

2.3 クイックソート

本研究ではクイックソートをベースとして BSP モデル上で実行可能なアルゴリズムを提案する。クイックソートはランダムなデータを整列するアルゴリズムとしては最も効率的なもので最速のソーティングであり、$O(n \log n)$ 時間で実行出来る。クイックソートの一般的な流れを説明すると、配列データの中から任意のデータを１つ選び、その値を基準として全てのデータ振り分ける。具体的には、その基準値より小さい値のものは前に、また、大きいものは後ろにというようにデータを前後に振り分けていく。この作業を前の操作で作られた２つの部分配列にも行う。この作業を繰り返し行い、全ての部分配列が 1 ないしは 0 になるとソーティング完了となる。パラレルクイックソートでは分割して部分配列ができる度に、その部分配列にプロセッサを割り当てていけば効率良く並列処理を行うことができる。また部分配列数が p 個に達した場合は各配列に 1 つずつプロセッサを割り当てソーティングを行えばよい。図 3 にクイックソートにおける配列の分割の様子を示す。

3　BSP モデル上でのクイックソートアルゴリズム

3.1　2 分木を用いたクイックソートアルゴリズム

BSP モデル上でソーティングを行う場合、クィックソートによって分割した 2 個の部分配列のうちの片方を他のプロセッサに送信し、それぞれの部分配列を 2 台のプロセッサを用いて分割、という処理を部分配列の個数が p 個になるまで再帰的に繰り返し、その後各プロセッサで逐次クィックソートを用いてソーティングを行えばよい。

あるプロセッサを用いてサイズ n の配列 A をサイズ n_1, n_2 の 2 個の部分配列 A_1, A_2 に分割するのに掛かる時間は $O(n)$ である。また、片方の部分配列を他のプロセッサに送るのに掛かる時間は $O(gn_1 + L)$ または $O(gn_2 + L)$ 時間となる。分割の基準値を適正に選択すれば $n_1 \approx n_2 \approx \frac{n}{2}$ となる。よって、分割に掛かる時間は $O(n + g\frac{n}{2} + L)$ である。1 度の分割により部分配列の個数は 2 倍になるので、p 個の部分配列を得るには $logp$ 回の分割が必要となる。また、各分割で適切な基準値を用いた場合、i 回目の分割の開始時点における部分配列のサイズは $\frac{n}{2^{i-1}}$ となる。従って、$\log p$ 回の分割に掛かる時間の和は

$$\sum_{i=1}^{\log p} O(\frac{n}{2^{i-1}} + g\frac{n}{2^i} + L) = O(gn + L\log p)$$

となる。各分割で適切な基準値を用いれば p 個の各部分配列のサイズは $\frac{n}{p}$ となるので、クィックソートを用いて $O(\frac{n\log n}{p})$ 時間でソーティングを行える。従って、全体の計算量は

$$O(\frac{n\log n}{p} + gn + L\log p)$$

となる。

3.2　本研究で提案するアルゴリズム

一般に、同期に掛かる時間 L は非常に大きいと考えられるため、プロセッサ台数 p が大きいとき、2 分木を用いたクィックソートの計算量は大きくなる。そこで本研究では、同期回数を減らすため、k 分木を用いたクィックソートアルゴリズムを提案する。

以下に本研究で提案するアルゴリズム k-QS を示す。初期状態においては、プロセッサ P_0 がサイズ n の配列 A を保持していると仮定する。また、プロセッサ台数 p は n に比べて充分に小さいとする。

(algorithm k-QS)

入力: サイズ n の配列 A。プロセッサ P_0 が A を保持する。

出力: A のソート済み配列 B。プロセッサ P_i $(0 \leq i < p)$ が B の部分配列 B_i を保持する。

1. 以下の操作を $\frac{\log p}{\log k}$ 回繰り返す。i 回目の繰り返しの開始時点においてプロセッサ P_j $(0 \leq j < 2^{i-1})$ は配列 $A_j^{(i)}$ を保持しているとする。ただし $A_0^{(1)} = A$ である。
   (a) プロセッサ P_j $(0 \leq i < 2^{i-1})$ は配列 $A_j^{(i)}$ から $3(k-1)$ 個のデータ $d_j^{(1)}, d_j^{(2)}, ..., d_j^{3(k-1)}$ をランダムに選び出す
   (b) プロセッサ P_j $(0 \leq i < 2^{i-1})$ はデータ $d_j^{(1)}, d_j^{(2)}, ..., d_j^{3(k-1)}$ を逐次クィックソートを用いてソーティングする。ソーティング後のデータを $e_j^{(1)}, e_j^{(2)}, ..., e_j^{3(k-1)}$ とする。
   (c) プロセッサ P_j $(0 \leq i < 2^{i-1})$ は配列 $A_j^{(i)}$ を以下の式を満たす k 個の部分配列 $A_{jk}^{(i+1)}, A_{jk+1}^{(i+1)}, A_{jk+2}^{(i+1)}, ..., A_{jk+k-1}^{(i+1)}$ に分割する。
      - $\forall d \in A_{jk}^{(i+1)}, d < e_j^{(2)}$
      - $\forall d \in A_{jk+l}^{(i+1)}, e_j^{(3l-1)} \leq d < e_j^{(3l+2)} (1 \leq l < k-1)$
      - $\forall d \in A_{jk+k-1}^{(i+1)}, e_j^{3(k-1)-1} \leq d$
   (d) プロセッサ P_j $(0 \leq i < 2^{i-1})$ は部分配列 $A_l^{(i+1)}$ $(jk \leq l < jk+k)$ をプロセッサ P_l に送信する。
2. プロセッサ P_j $(0 \leq j < p)$ は逐次クィックソートを用いて部分配列 $A_j^{\frac{\log p}{\log k}}$ をソーティングし、ソーティング後の配列を B_j とする。

図 4 に従来型の 2 分木を用いたクイックソートと、本研究で提案した k 分木を用いたクイックソートの実行の様子を示す。

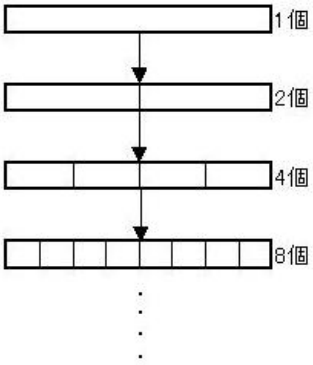

図 3: クイックソート実行の様子

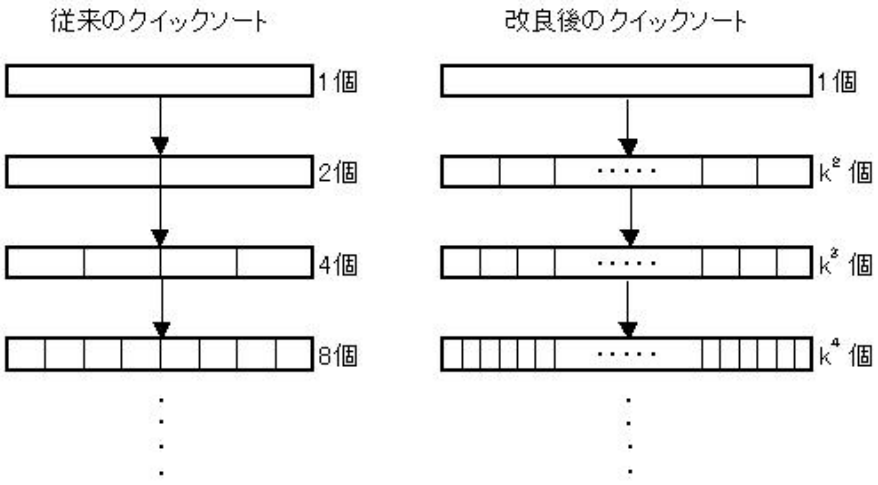

図 4: 従来型のクイックソートと改良後のクイックソートの実行の様子

4　結果と考察

4.1　アルゴリズムの計算量

アルゴリズム q-QS が正しくソーティングを解くことは明らかである。以下では、アルゴリズム q-QS の計算量について考察する

アルゴリズム q-QS の計算量については以下の定理が成り立つ。

定理 1 任意の整数 k (≥ 2) に対し、アルゴリズム q-QS は BSP モデル上で $(O\frac{n\log n}{p} + n(\log k + g) + L\frac{\log p}{\log k})$ 時間でソーティングを解く。

(証明) 各繰り返しにおいて、分割の基準値として適切な値を用いれば分割前のデータサイズを n とするとき、分割後の k 個のデータサイズは高い確率でそれぞれ $\frac{n}{k}$ となる。従って、i 回目の繰り返しの開始時点で各プロセッサが保持するデータのサイズは高い確率で $O(\frac{n}{k^{i-1}})$ である。

アルゴリズム q-QS の (a) において、$3(k-1)$ 個のデータ選択は $O(k)$ 時間で行うことができる。また、(b) においてソーティングは $O(k\log k)$ 時間で行える。(c) において、サイズ n 個のデータを k 個に振り分けは $O(n\log k)$ 時間で行うことができる。従って、i 回目の繰り返しにおけるデータの振り分けは $O(\frac{n\log k}{k^{i-1}})$ 時間で行うことができる。(d) では各プロセッサは $\frac{n}{k^{i-1}}$ 個のデータを送信し、$\frac{n}{k^i}$ 個のデータを受信する。従って i 回目の繰り返しにおける時間計算量は

$$O(\frac{n\log k}{k^{i-1}} + g\frac{n}{k^{i-1}} + L)$$

時間となる。また、2. において各プロセッサは $O(\frac{n}{p})$ 個のデータをソーティングするので $O(\frac{n\log n}{p})$ 時間で実行できる。

よって、アルゴリズム全体の計算量は

$$O(\frac{n\log n}{p} + n\log k + gn + L\frac{\log p}{\log k})$$

となる。 □

$\log^2 k = \frac{L\log p}{n}$ のとき最小となる。従って以下の系が成り立つ。

系 1 アルゴリズム q-QS は BSP モデル上で $O(\frac{n\log n}{p} + gn + \sqrt{Ln\log p})$ 時間でソーティングを解く □

5　まとめ

本研究では BSP モデル上で $O(\frac{n \log n}{p} + n \log k + gn + L\frac{\log p}{\log k})$ 時間でソーティングを行う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、同期時間 L が長いとき効率良く実行することが出来るアルゴリズムである。

謝辞

本研究の報告書を作成するにあたり、石水隆助手には色々なことを基礎から根気よく教えていただいたり、無理を言って夜遅くまで残っていただいたりと本当にお世話になりました。これほど生徒の為に尽力して下さる先生は中々いらっしゃらないと思います。僕は石水助手の研究室に入れて本当によかったです。また、同研究室の西谷君や小林君のアドバイスや協力にも感謝の意を表したいと思います。誠に有難うございました。

参考文献

1) R.Cole, Parallel merge sort, SIAM Journal of Conputing, Vol.14, No.4, pp.770–785, 1988.

2) F.Dehne, A.Fabri and A.Rau-Chaplin, Scalable Parallel Computational Geometry for Corse Grained Multicomputers, Proceeding of ACM Symposium on Computational Geometry, pp.298–307, 1993.

3) 石畑 清, アルゴリズムとデータ構造, 岩波書店 pp.161–184, 1989.

4) 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 モデル及び BSP* モデル上の並列アルゴリズム, 電子通信学会論文誌 D-I, Vol.J82-D-I, No.4, pp.533–542, 1999.

5) J.JáJá, An Introduction to Parallel Algorithms, Addison-Wesley Publishing Company, 1992.

6) R.Reischuk, Probabilistic algorithms for sorting and selection, SIAM Journal of Computing, Vol.14, No.2, pp.396–409, 1985.

7) L.G.Valiant, A Bridging Model for Parallel Computation, Comm. of the ACM, Vol.33, No.8, pp.103–111, 1990.