卒業研究報告書

題 目

BSPモデル上でのクイックソート

指 導 教 員

石 水　隆 助手

報 告 者

02–1–47–100

福 井　達 也

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)[2] モデル上でソーティングを行う効率の良い並列アルゴリズムを提案する。

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)[1] モデルと異なり、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。

本研究で提案するソーティングアルゴリズムは、クィックソートをベースとし、各プロセッサへのデータの割り当て方および基準値の取り方を考慮することにより、BSP 上で効率良く実行することができる。本研究で提案する並列アルゴリズムは、ランダムな値をクィックソートにおけるデータ分割の基準値とすることにより、基準値を選択する時間を短縮している。平均的には基準値の値はほぼ中央値となるため、平均的にはプロセッサに対してデータを均等に割り当てられ、プロセッサ間で負荷分散が行われることにより、平均的時間計算量は短縮される。また、本研究では提案したアルゴリズムの計算量を実験的に評価するため、シミュレートプログラムを作成しその実行時間の測定を行う。

目 次

1 序論 **1**
1.1 本研究の背景 1
1.1.1 並列処理とは 1
1.1.2 並列処理の目的 1
1.1.3 並列計算機 1
1.1.4 PRAM モデルと BSP モデル 2
1.1.5 ソーティング 2
1.2 本研究の目的 3
1.3 本報告書の構成 3

2 準備 **4**
2.1 BSP(Bulk Synchronous Parallel) 4
2.2 並列クイックソート (Parallel quick sort) 5
2.3 PRAM 上での並列クイックソート 5

3 研究内容 **7**
3.1 BSP モデル上のクイックソートアルゴリズム 7
3.2 シミュレートプログラム 8

4 結果・考察 **9**
4.1 処理時間の測定の方法 9
4.2 処理時間の測定の結果 9
4.2.1 平均時間についての考察 9
4.2.2 出力についての考察 10

5 結論 **11**

6 謝辞 **12**

A 付録について **14**

1　序論

1.1　本研究の背景

1.1.1　並列処理とは

１つの目的を持った処理を、いくつかの処理に分割して、これらを同時に行うのが、並列処理 (Parallel Processing) である。100 枚ある答案を採点して最も良い成績の答案を探すという問題を例にする。採点者が２人いれば、１人で全ての答案を採点する時の半分の時間で終わり、３人いれば３分の１の時間で終わることは簡単に想像できる。しかし、ここで 100 人いたら 100 分の１になるだろうか。答案を配ったり、集めたり、最も良い点数を探すのにも時間がかかる。計算機を使い並行処理を行う場合においても、同じような問題が起こる。このように、処理を分割したために新たに必要となる処理をオーバヘッドという。２人や３人で処理をする時にはほとんど問題にならないが、処理する人が増えるにつれオーバヘッドを考慮する必要がでてくる。これは、全体の作業量と作業者の数のバランスの問題である。計算機の世界では、作業量とは処理すべきデータ量で、作業者とはプロセッサのことである。

1.1.2　並列処理の目的

並行処理の目的は大きく分けて２つある。第一の目的は、問題を小さいサイズの部分問題に分割し、複数のプロセッサ上で各部分問題を同時に処理することにより、高速化を図ることである。第二の目的は、耐故障 (Fault Toletant) 性や無待機 (Wait Free) 性を得ることである。システムを多重化することにより、何台かのプロセッサが故障してもシステム全体としては処理を行うことができる。本研究では、第一の目的である処理の高速化を取り上げている。

1.1.3　並列計算機

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が並列計算機 (Parallel Computer) である。1980 年代後半並列計算機の製品化が活発になり、現在ではスーパーコンピュータや大規模なサーバなどはすべて並列型になっている。ワークステーションやパソコンも、高性能のものは２～４個のプロセッサを持っている。また最近は、複数の計算機をネットワーク接続して並列/分散処理をさせるクラスタ (Cluster) 処理も利用されている。さらには、地球規模でネットワーク接続されたコンピュータで分散処理を行うグリッド (Grid) 処理も研究されている。並列計算機は、共有メモリ型並列計算機 (Shared Memory Parallel Computer) と分散メモリ型並列計算機 (Distributed Memory Parallel Computer) の２つに分類される。共有メモリ型並列計算機は共有メモリ (Shared Memory) とそれに接続のプロセッサから成る。一方、

分散メモリ型並列計算機は局所メモリ (Local Memory) を持つ複数のプロセッサと、それらを結びつけるネットワークから成る。

一般に共有メモリ型計算機は通信遅延が短く、プロセッサ間の同期も取りやすいが、プロセッサ数を増やすことは困難である。逆に分散メモリ型計算機は比較的多数のプロセッサを持つことができるが、プロセッサ間の通信には時間がかかる。このため、プロセッサ数が少ない並列計算機は共有メモリ型が、プロセッサ数が多い並列計算機は分散メモリ型が主流になっている。

1.1.4 PRAM モデルと BSP モデル

PRAM(Parallel Random Access Machine) モデルは複数のプロセッサがメモリを共有するモデルである。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内のデータ 1 単位時間で行うことができる。加えて PRAM は細粒度同期式 (Fine Grain Synchronization) であり、1 単位時間ごとに全てのプロセッサで同期をとることができる。

このように並列処理機構が理想的に設定されているため、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。さらに PRAM は他の並列計算機モデル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 は通信時間などを考慮しない理想的なモデルであるため PRAM 自体の実現は困難であり、現実とのギャップがあることが問題である。

この問題を考慮したモデルが BSP(Bulk Synchronous Parallel) モデルである。BSP モデルとは通信時間、同期間隔を考慮した非同期式分散メモリ型並列計算モデルである。また局所メモリを持つ複数の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とるための同期機構からなり、プロセッサ間の通信はネットワークを通じてメッセージ交換をすることにより行われる。

本研究では BSP モデルを対象とする。

1.1.5 ソーティング

ソーティングは様々な分野で利用されている基本的な問題であり、各計算モデル上でこれを高速に解くアルゴリズムを得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イックソート (Quick Sort) やマージソート (Merge Sort) を用いて $O(n \log n)$ 時間でソーティングを行える。Cole は CREW-PRAM 上で $O(\frac{n \log n}{p} + \log n)$ 時間 p プロセッサの並列アルゴリズムを提案した。

1.2　本研究の目的

本研究では BSP モデル上で効率良くソーティングを行う並列アルゴリズムを提案する。様々な分野で利用されているソーティングの高速化を図るためにシミュレートプログラムを作成し、本研究で提案したアルゴリズムの実験的評価を行う。

1.3　本報告書の構成

- ２章:並列計算モデルおよび並列クイックソートの説明をしている。
- ３章:本研究で提案したアルゴリズムの説明をしている。
- ４章:プログラムをシミュレートし、処理時間の測定と出力の検証を行っている。
- ５章:４章の結果から結論を述べている。

2 準備

2.1 BSP(Bulk Synchronous Parallel)

BSP(Bulk Synchronous Parallel) とは Valiant により提案された分散メモリ型並列計算モデルである。BSP は以下の要素からなる。

- 局所メモリをもつ複数のプロセッサ:各プロセッサは１〜P のプロセッサ番号を持ち、P_1,P_2,P_3 $\cdots$ P_P と表される。
- 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
- プロセッサ間のバリア同期を実現するための同期機構

BSP の概念図を第１図に示す。BSP モデルは、通信遅延、同期時間を表すために以下の３つのパラメタを持つ
P:プロセッサ数
g:１個のメッセージを送信あるいは受信するためにかかる時間
L:バリア同期時間
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ューズと、送信命令および受信命令の列から成る通信フュ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非同期に実行する。また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。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ュ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ものと仮定する。あるスーパーステップで各プロセッサ $P_i(1 \leq i \leq P)$ がそれぞれ w_i 個の内部計算命令と h_i 個の送信命令あるいは受信命令を実行するとき、そのスーパーステップ全体の実行時間 t は

$$t = O(w + gh + L)$$

であると仮定されている。ただし、w=maxw_1,w_2,$\cdots$,w_p、h=maxh_1,h_2,$\cdots$,h_p である。

スーパーステップの特性を持つことにより以下に挙げる特性を保たれる。
・データ通信の依存関係の解析ができる
・バリア同期に必要な時間を通信遅延ととらえることにより、通信コストを考慮できる
・バリア同期以外の同期は無いという非常に緩い同期を仮定するだけでいい

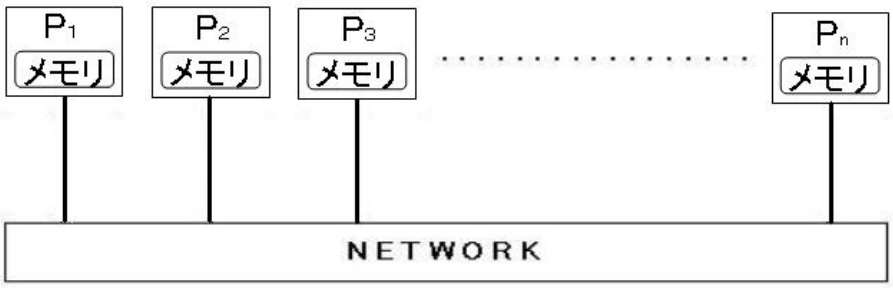

図 1: BSP モデル

2.2 並列クイックソート (Parallel quick sort)

クイックソート (Quick Sort) はランダムなデータに対しては最も効率的なソート法である。クイックソートの基本的な流れを説明する。データの中からある値を選び、それを基準値として、データ全体を基準値以下のデータから成る部分データと基準値以上のデータから成る部分に分割する。各部分データに対し、同様の作業で分割によってできた部分データに含まれるデータ数が 1 になるまで、再帰的に繰り返す。

2.3 PRAM 上での並列クイックソート

並列クイックソートは分割された区間ごとにプロセッサを割り当てることにより、高速化を目指した並列アルゴリズムである。n 個のデータが配列に入っているとする。そして、データを分ける基準となる値のことを基準値という。与えられた配列 A に対し、PRAM 上で A の区間 [0,n-1] の並列クイックソートを行う場合、手順は次のようになる。
(1)A の区間 [0,n-1] に対し、手続き (procedure)divide で呼び出す
(2) 手続き divide を呼び出し指定された区間 [l,r] に対し以下の処理を行う

- データ数が 1 以下のとき:何もしない
- データ数が 2 のとき:区間内のデータの大小を比較し、逆順なら入れ替える
- データ数が 3 以上のとき:次のようにデータを振り分ける
  - (a) 区間内から基準値を 1 つ選ぶ

– (b) 区間内のデータに対し、基準値より小さいものと大きいものとを２つの区間 [l,m-1] と [m,r] にそれぞれ振り分ける
– (c) 両方の区間 [l,m-1] および [m,r] に対し、手続き divide を用いてデータの振り分けを再帰的に行う

複数のプロセッサを用いて平列処理を行う場合、(2)　(c) の処理は２個のプロセッサで平列に行うことができる。従って、並列処理を用いてクイックソートを行う場合、区間の個数が P 個になるまでは、区間を分割するごとに１つの区間に１台のプロセッサを割り当てていく。区間の個数が P 個になった後は、各プロセッサで割り当てられた区間を逐次クイックソートを用いてソートすればよい。PRAM 上での並列クイックソートの概念図を第２図に示す。

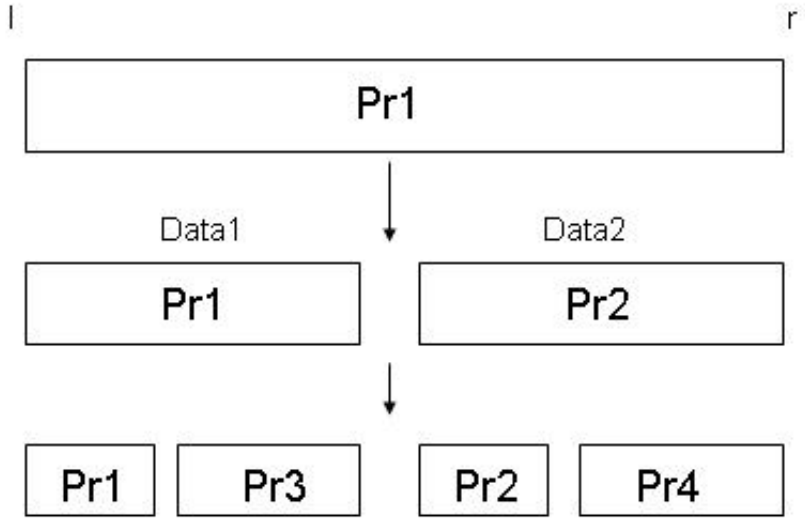

図 2: PRAM 上での並列クイックソート

3 研究内容

3.1 BSP モデル上のクイックソートアルゴリズム

以下に本研究で提案したBSP モデル上のクイックソートアルゴリズム **BSP-QS** およびアルゴリズム BSP-QS 中で使用する手続き **devide** を示す

(アルゴリズム **BSP-QS**)

入力: サイズ n の配列 A。プロセッサ P_i $(1 \leq i \leq p)$ が A のサイズ $\frac{n}{p}$ の部分配列 $A_i = \{A[\frac{(i-1)n}{p}], A[\frac{(i-1)n}{p}+1], A[\frac{(i-1)n}{p}+2], ..., A[\frac{in}{p}-]\}$ を保持する。

出力: A のソート済み配列 B。プロセッサ P_i $(1 \leq i \leq \quad p)$ が B の部分配列 B_i を保持する。

1. プロセッサグループ $\mathcal{P} = \{P_1, P_2, ..., P_p\}$ に対し手続き divide を呼び出す。
2. 全てのプロセッサ P_i $(1 \leq i \leq p)$ は逐次クイックソートを用いて保持する部分配列 A_i をソートしソート済みの部分配列を B_i とする。

(手続き **divide**)

プロセッサグループ $\mathcal{P} = \{P_l, P_{l+1}, ..., P_r\}$ を用いて以下の処理を行う

1. プロセッサ P_i $(l \leq i \leq r)$ は保持する部分データ A_i から適当な基準値候補 d_i を一つ選び出す。
2. プロセッサ P_i $(l < leqi \leq r)$ は基準値候補 d_i を P_l に送信する
3. プロセッサ P_l は基準値候補集合 $\{d_l, d_{l+1}, ..., d_r\}$ から適当な基準値 d を選び出す
4. プロセッサ P_l は基準値 d をプロセッサ $P_{l+1}, P_{l+2}, ..., P_r$ に送信する。
5. プロセッサ P_i $(l \leq i \leq r)$ は、保持する部分データ A_i を基準値 d 以下のデータから成る部分データ A_i^{low} と基準値 d 以上のデータからなる部分データ A_i^{high} に分割する。
6. プロセッサグループ $\mathcal{P} = \{P_l, P_{l+1}, ..., P_r\}$ をプロセッサグループ $\mathcal{P}^{low} = \{P_l, P_{l+1}, ..., P_{\frac{l+r}{2}}\}$ とプロセッサグループ $\mathcal{P}^{high} = \{P_{\frac{l+r}{2}+1}, P_{\frac{l+r}{2}+2}, ..., P_r\}$ に分割する
7. プロセッサ P_i $(l \leq i \leq r)$ は部分データ A_i^{low} をプロセッサグループ $\mathcal{P}^{low}$ に属するプロセッサの中の 1 台に送信し、部分データ A_i^{high} をプロセッサグループ $\mathcal{P}^{high}$ に属するプロセッサの中の 1 台に送信する

8. プロセッサ P_i $(l \leq i \leq r)$ は 7. で受信したデータを新たな A_i とする。

9. プロセッサグループ $\mathcal{P}^{low}$ に属するプロセッサが 2 台以上であれば、プロセッサグループ $\mathcal{P}^{low}$ に対し手続き divide を再帰的に呼び出す。また、プロセッサグループ $\mathcal{P}^{high}$ に属するプロセッサが 2 台以上であれば、プロセッサグループ $\mathcal{P}^{high}$ に対し手続き divide を再帰的に呼び出す。

3.2 シミュレートプログラム

本研究で提案したアルゴリズム BSP-QS の正当性および時間計算量を実験的に評価するため、JAVA 言語を用いてアルゴリズム BSP-QS のシミュレートプログラムを作成し、その出力および実行時間を検証する。

4　結果・考察

4.1　処理時間の測定の方法

データ数 N=１０００ (個)、プロセッサ数 p=１００ (台) とし、データの送受信時間と同期時間を固定し、その場合の基準値の選択方法を変化させ、出力の検討および実行時間の検証を行っている。

基準値の選択方法は表１に示す。

- ランダム … データの中から無作為に値を１つとる
- サンプル … データの中から無作為に値を３個とり、その中から中央値をとる
- セレクト …

表 1: 基準値の選び方

	候補	決定
	ランダム	ランダム
	ランダム	サンプル
	ランダム	セレクト
	サンプル	ランダム
	サンプル	サンプル
	サンプル	セレクト
	セレクト	ランダム
	セレクト	サンプル
	セレクト	セレクト

4.2　処理時間の測定の結果

プログラムを１０００回シミュレートし、平均時間の計算を行った。また出力から処理時間の標準偏差を計算することで、データの散らばり方の検証している。

～　のシミュレートを行った結果は表２に示している。

4.2.1　平均時間についての考察

表２から基準値の選ぶ時にランダムを使うと内部計算時間は減少し、通信時間は増加していることがわかる。またセレクトを使うと内部計算時間は増

表 2: 測定結果

	平均時間	標準偏差
	2423+1191*g+21*L	2947+239*g
	2146+1000*g+21*L	2133+155*g
	4362+729*g+21*L	1096+32*g
	2073+1008*g+21*L	1705+161*g
	1928+875*g+21*L	1374+97*g
	4258+692*g+21*L	766+19*g
	4907+775*g+21*L	1178+48*g
	4504+732*g+21*L	848+33*g
	6605+677*g+21*L	561+12*g

加し、通信時間は減少していることがわかる。これらの結果から基準値の選び方を工夫すると、処理に時間がかかってしまうために内部時間が増加するが、基準値が中央値に近づくことから通信時間が短縮できると考られる。

また g の値が大きくなるにつれて、基準値の選び方を工夫する有用性がでてくることもわかる。

4.2.2 出力についての考察

表 2 から基準値を選ぶ時にランダムよりセレクトを使う方が標準偏差の値が小さいことがわかる。つまりデータにあまりばらつきがないということである。ゆえに基準値のとり方を工夫することによって、安定した結果を求めることができることが検証された。

5　結論

本研究で提案したBSP上でのクイックソートアルゴリズムをプログラムにし、シミュレートすることで基準値選択の重要性が明らかになった。1つのメッセージの送受信時間gが非常に大きい場合、もしくわ安定した結果を求めたい場合は基準値の選び方を工夫することで求めることができる。また内部計算時間を短縮したい場合には基準値をランダムに選択することで、無駄な計算時間を省くことができる。しかしこれらには考察で述べた欠点も存在する。

内部計算時間を抑え、安定した結果を出力するアルゴリズムを考えることが今後の課題となる。

6　謝辞

研究するにあたり、様々な助言をいただいた石水先生および情報論理工学研究室の皆様には深く感謝申し上げます

参考文献

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

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

[3] 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).

A　付録について

本研究で作成したプログラムのソースファイルなどは，付録として巻末にまとめておく．