卒業研究報告書

題 目

キャッシュ付 PRAM 上での並列ソート

指導教員

石水隆助手

報告者

01-1-26-081

櫻打 純視

近畿大学理工学部電気工学科

平成 17 年 2 月 19 日提出

目次

第1章　序論

1.1.　並列処理

今日、多くの分野においてコンピュータによる高速処理が求められている。高速処理を行うための手法としては、性能のいい高速な計算素子を用いるか、あるいは並列処理を行うかの 2 つの手法がある。ここで並列処理とは、大きな仕事を複数のより小さな仕事に分割し、これらの分割した仕事を同時に処理することにより元の大きな仕事を高速に処理することである。近年、計算素子は高度に集積化され、極めて高速に処理を行うことができる。しかし、素子の集積化による高速化はいずれ限界に達すると考えられる。一方、並列処理にはそのような限界は本質的に存在しない。このため、並列処理に対して大きな期待が寄せられている。

1.2.　並列計算機

複数のプロセッサを持ち並列処理を行える計算機が並列計算機(Parallel Computer)である。並列計算機は、共有メモリ(Shared Memory)と複数のプロセッサ(Processor)から成る共有メモリ型並列計算機(Shared Memory Parallel Computer)と、局所メモリ(Local Memory)を持つ複数のプロセッサとそれを結ぶ通信網から成る分散メモリ型並列計算機(Distributed Memory Parallel Computer)に分類される。一般的に共有メモリ型並列計算機は高速な通信を行えるが、プロセッサ数が増えるにつれその実現は困難になる。一方、分散メモリ型並列計算機は多数のプロセッサを持つ計算機を実現し易いが、プロセッサ間の通信には時間がかかり、またデータの局所性も考慮しなければならない。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となっている。

1.3.　並列アルゴリズムと並列計算モデル

並列計算機上では、従来の逐次アルゴリズム(Sequential Algorithm)を元にしたプログラムは効率良く実行できず、並列処理を考慮した並列プログラム(Parallel Program)を作る必要がある。並列プログラムの元となるアルゴリズムが並列アルゴリズム(Parallel Algorithm)である。並列アルゴリズムは仕事をどのように分割し、各プロセッサに対して分割した仕事を割り当てるかを考慮して設計される。

一般的に、並列アルゴリズムの設計・開発は並列計算機を抽象化した並列計算モデル(Parallel Computing Model)の上で行われる。並列計算モデルを用いることにより、個々

の並列計算機ごとに異なる性質に囚われることなく並列アルゴリズムの設計・開発を行うことができる。

逐次計算では、標準的な計算モデルとしてRAM(Random Access Machine)が用いられる。一方、並列計算ではメモリの形式、プロセッサ間の通信や同期、データの扱い方の違いなどが異なるため標準的な計算モデルを決めにくい。このため PRAM(Parallel Random Access Machine),メッシュ(Mesh)接続型並列計算モデル、ハイパーキューブ(Hyper-Cube)接続型並列計算モデルなど様々な並列計算モデルが提案されている。

1.4. キャッシュと二次記憶

現在のほとんどのコンピュータは、メモリとして小容量だが高速なキャッシュ(Cache)と低速だが大容量な二次記憶を持っている。ほとんどの処理では、少数のデータが繰り返し使用される。そのため、ほとんどの場合、使用したデータをキャッシュに一時保存しておき、再度同じデータを用いるときはキャッシュ上から読み出すことにより高速に処理を行うことができる。キャッシュの大きさには制限があるため、どのデータをキャッシュに残すかは考慮しなければならない。また、通常、二次記憶からのデータの読み出しはある程度まとまった大きさのデータとして扱われるため、連続したメモリセル内に保存されているデータに対しては効率良く取り出すことができる。

1.5. 本研究の目的

本研究では PRAM の拡張モデルであるキャッシュ付 PRAM(Cached Parallel Random Access Machine)の上で整列問題に対して効率の良い並列アルゴリズムを提案すること目的とする。整列問題は様々な分野に応用できる基本的な問題であり需要が高い。このため並列計算機上で高速に実行できるアルゴリズムを開発することは非常に重要な課題ある。また、キャッシュ付 PRAM は多くの実在する並列計算機に対応し汎用性が高いモデルである。

第2章　準備

2.1.　PRAM

PRAM(Parallel Random Access Machine)は RAM を並列化した共有メモリ型並列計算モデルである。PRAM は同一の性能を持つ複数のプロセッサ(Processor)と全てのプロセッサが任意の位置にあるメモリセルにランダムアクセスできる共有メモリ(Shared Memory)から成る。PRAM の各プロセッサは、共有メモリ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、任意の演算を 1 単位時間で行えると仮定されている。また、1 単位時間ごとに全てのプロセッサで同期が取られていると仮定されている。

共有メモリ型であるため PRAM ではデータの局所性は考慮しない。またプロセッサ間通信は共有メモリを通して行われるため、通信遅延は発生せず、同期のコストも存在しない。このため PRAM 上では並列アルゴリズムの設計・開発を行うことが容易となる。また問題そのものが持つ本質的な難しさを考慮し易い。加えて、PRAM 上で設計された並列アルゴリズムは他の計算モデル上でも有用であることが多く、汎用性が高いと考えられている。

プロセッサの共有メモリに対するアクセスについては以下のように仮定されている。複数のプロセッサによる共有メモリ内の異なるセルへのアクセス対しては、全てのプロセッサが自由に読み書きできる。複数のプロセッサによる共有メモリ内の同一のセルへのアクセスに対しては、それをどのように処理するかにより PRAM は以下の 3 種に分類される。

- EREW PRAM(Exclusive Read Exclusive Write) 同時読み出しも同時書き出しも行えない。
- CREW PRAM(Concurrent Read Exclusive Write) 同時読み出しは行えるが同時書き込みは行えない。
- CRCW PRAM(Concurrent Read Concurrent Write) 同時読み込みも同時書き込みも行える。

2.2.　キャッシュ付 PRAM

キャッシュ付 PRAM(Cached Parallel Random Access Machine)は各プロセッサが一定のサイズのキャッシュを持っている。各プロセッサは、キャッシュ上のデータに対してのみ演算を行う事ができる。一方、共有メモリに対しては直接演算を行うことができず、一旦キャッシュにデータを読み込まなくてはいけない。共有メモリからキャッシュへの読み書きには、通信遅延が発生する。これらの特性を表すために、キャッシュ付きＰＲＡＭは以下に示す三つのパラメータを持っている。

- プロセッサ数 p
- キャッシュサイズ c：各キャッシュは、c 個のデータを保存することができる。
- 通信遅延時間 d：キャッシュ - 共有メモリ間でデータの読み書きを行うときの遅延時間。各プロセッサは、共有メモリ上の連続した c 個のメモリセル上にあるデータを d 時間でキャッシュに読み出すことができる。また各プロセッサはキャッシュ上の c 個のデータを d 時間で共有メモリ上の連続したメモリセル上に書き込むことができる。

2.3. 整列

整列(Sorting)とは、一定の規則(数値の昇順、文字列の辞書式順等)に従いデータを並べ替える操作である。整列は重要な基本問題の一つであり、様々な分野で使用されるため、高速な整列アルゴリズムの必要性は非常に高い。RAM では、クィックソート(Quick sort)、マージソート(Marge Sort)、ヒープソート(Heap Sort)等、高速で実用性の高い整列アルゴリズムが知られている。PRAM でも並列サンプルソート(Parallel Sample Sort)、奇偶マージソート(Odd-Even Marge Sort)を初め多くの整列アルゴリズムが提案されている。

本研究では、PRAM 上のアルゴリズムである並列クィックソート(Parallel Quick Sort)および並列マージソート(Parallel Marge Sort)をキャッシュ付 PRAM 上で実行させたときの時間計算量を解析し、キャッシュ付 PRAM の有用性および各アルゴリズムに対するキャッシュサイズと通信遅延の影響を考察する。

2.3.1. 並列クィックソート

クィックソートの基本的な処理の流れは以下の通りである。

n 個のデータからなるデータの配列 A が与えられたとき、まず A 中から適当なデータ m を選び出す。次に m を基準値として、A を m より小さいデータからなる部分配列 A_L と m より大きいデータから成る部分配列 A_R に分割する。部分配列 A_L,A_R に対し同様の処理を部分配列の大きさが 1 になるまで再帰的行うことにより、A 全体を整列することができる。

PRAM(またはキャッシュ付 PRAM)上でクィックソートの並列処理を行う場合、部分配列 A_L,A_R にそれぞれ別のプロセッサを割り当て並列処理を行うことができる。部分配列の個数が p 個になるまで配列の分割を行い、その後 p 台のプロセッサを用いて各部分配列を逐次クィックソートを用いて整列すれば良い。

2.3.2. 並列マージソート

マージソートの基本的な処理の流れは以下の通りである。

n 個のデータからなるデータの配列 A が与えられたとき、まず A をそれぞれ 1 個のデータからなる n 個の部分配列 $A_1,A_2,\ldots,A_n$ に分割し、2 個ずつの組(A_{2i},A_{2i+1}) $(1 \leqq i \leqq n//2)$ にし、空の配列 $B_1,B_2,\ldots,B_{n/2}$ を作る。各組(A_{2i},A_{2i+1})に対して、部分配列 A_{2i},A_{2i+1} の先頭のデータのうち小さい方を取り出し B_i に加えるという操作を繰り返すことにより 2 倍の長さの整列した部分配列 $B_1,B_2,\ldots,B_{n/2}$ を作ることができる。この操作を A 全体が 1 個の整列済み配列になるまで再帰的に行う。

PRAM(またはキャッシュ付 PRAM)上でマージソートの並列処理を行う場合、まず各プロセッサに n/p 個のサイズ 1 の部分配列を割り当て、逐次マージソートを用いてサイズ n/p のソート済み部分配列を作る。その後、部分配列を 2 個ずつ組にして各プロセッサに割り当て並列処理を行えばよい。

第3章　方法

3.1.　シミュレートプログラム

本研究では、キャッシュ付 PRAM 上での並列クィックソートおよび並列マージソートの時間計算量の解析を行うため、両アルゴリズムのキャッシュ付 PRAM 上での動作のシミュレートを行うシミュレートプログラムを作成した。付録に本研究で作成したシミュレートプログラムを示す。

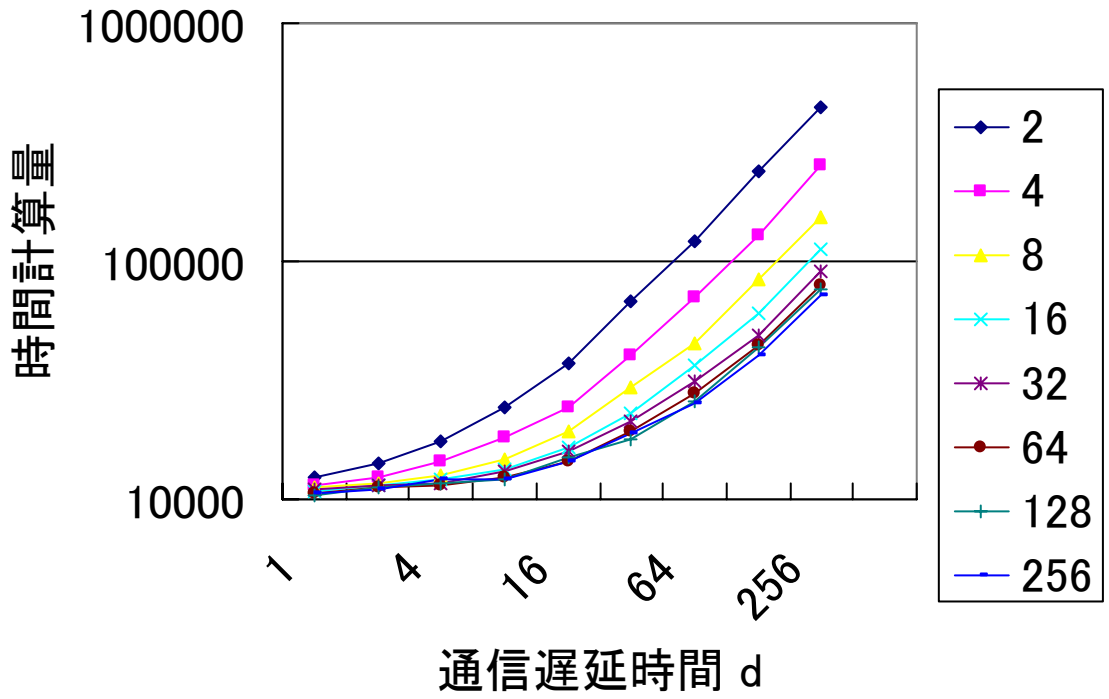
本研究で作成したプログラムは、データ数 n、プロセッサ数 p、キャッシュサイズ c、通信遅延時間 d を入力として与えたとき、キャッシュ付 PRAM 上での並列クィックソートおよび並列マージソートの実行時間を計測するプログラムである。

第4章　結果

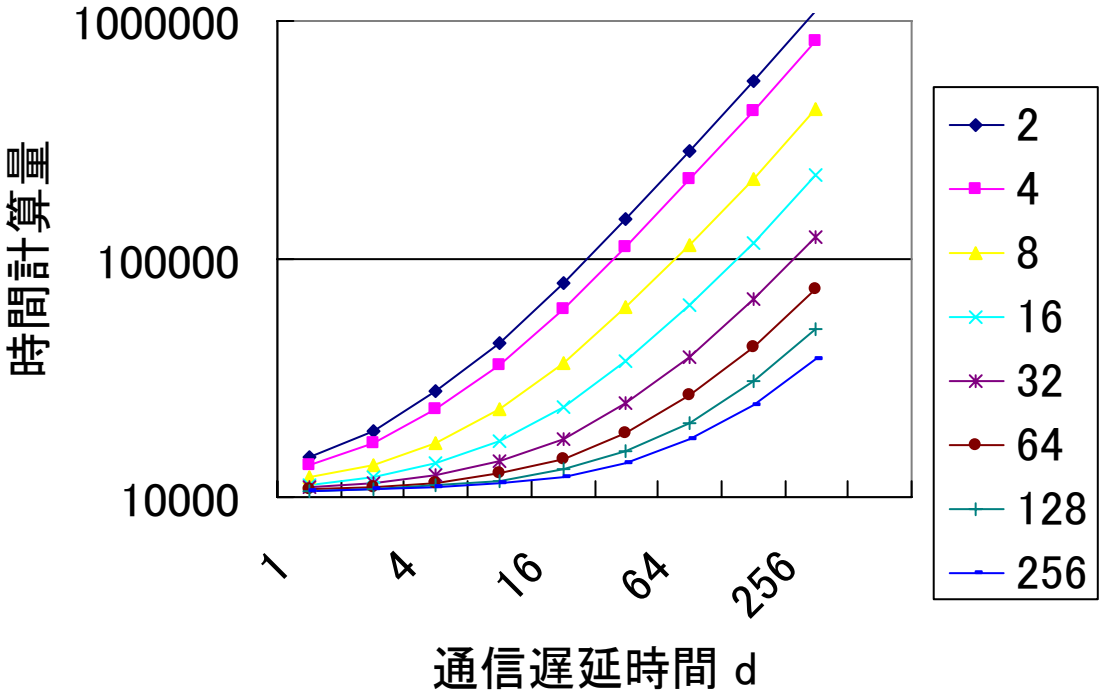
4.1.　実行結果

付録に示したシミュレートプログラムを用いて、データ数 n =1024、プロセッサ数 p =128 とし、キャッシュサイズ c および信遅延時間 d を変化させた場合のキャッシュ付 PRAM 上での並列クィックソートおよび並列マージソートの実行時間を測定した。第 1 図および第 2 図に並列クィックソートおよび並列マージソートのシミュレートプログラムの実行時間を示す。

並列クィックソート、並列マージソート共に通信遅延時間 d が大きくなるにつれ、また、キャッシュサイズ c が小さくなるにつれ実行時間は長くなる。また、並列クィックソートに比べ、並列マージソートではキャッシュサイズによる実行時間への影響が大きい。



第 1 図　並列クィックソートの実行時間
Fig.1 Running time of Parallel Quick Sort



第 2 図　並列マージソートの実行時間

Fig.2 Running time of Parallel Marge Sort

第5章　考察

第 1 図、第 2 図より、並列クィックソート、並列マージソート共にキャッシュサイズの増加に伴い実行時間が短縮したのは、キャッシュサイズの増加によって共有メモリからのデータの読み込み回数が減少したためと考えられる。

またキャッシュが小さい時はクィックソートが、また大きい時はマージソートが実行時間が短くなった。これは、マージソートは一度に連続したデータを扱うため、キャッシュを効率良く使用できるからであると考えられる。　図　からクイックソートはキャッシュメモリのグラフが重なっている所がある。また綺麗な関数にはならず、キャッシュメモリがおよそ 32 を境にマージソートの方がクイックソートの数値が高いことがわかる。キャッツシュ付き並列クイックソートの環境では、適していない。またマージソートは、逆に効率よくなっているから、この環境には適しているといえる。

第6章　結論

今回は、以上の結果になったが、効率よくするにはキャッシュ付きＰＲＡＭではキャッシュサイズと通信遅延時間によって、クイックソートを使うか、マージソートを使うか使い分ける必要がある。

参考文献

1) 白鳥則朗、木下哲男、杉浦　茂樹“システムソフトウェアの基礎”、昭晃堂、2001
2) 渋沢進　“並列分散処理入門”、培風館、東京、1985
3) 上林弥彦、岡部　寿男、浜口　清治、武永康彦“プレパラータ先生の超並列計算講義”、東京、1996
4) 谷口秀夫“並列分散処理”、電子情報通信学会、コロナ社、東京、2003
5) 湯浅太一、安村　通晃、中田　登志之“はじめての並列プログラミング”、共立出版、東京、1999
6) 今泉貴史“プログラミングに活かすデータ構造とアルゴリズムの基礎知識”、アスキー、東京、2004
7) 平田富夫“アルゴリズムとデータ構造”、森北出版、東京、2002