Multi-PSIを用いた並列横型探索アルゴリズムの負荷分散に関する一考察

2S−5

池田 朋男　沼崎 浩明　田中 穂積

(東京工業大学 工学部)

1 はじめに

並列計算機を効率良く動作させるためには，並列性の高いアルゴリズムを用いるとともに，負荷の均等化を図るための負荷分散の方法を検討することが重要である．本研究では，OR並列性のある問題に対し，横型の全解探索を行なうアルゴリズムに対する負荷分散の方法を検討した．

実際には，横型探索問題の一例として，並列一般化LR構文解析アルゴリズム [1]を取り上げた．我々はすでに，負荷分散を行なうために各探索レベルのプロセスの同期を必要とする方法を試みた[2]．その方法には，各探索レベルで処理を区切るため，実行が逐次的になるという問題点があった．これを改善するために，本研究では，同期を必要としない負荷分散方法，すなわち，あるプロセスがデータを一つ出力するたびに，次のレベルのプロセスを直ちに別のプロセッサに割り当てる方法を考案し，これを並列論理型言語KL1で記述した．この二つの方法の処理時間を実際にMulti-PSI上で測定した結果，台数効果の大幅な改善が見られた．

2 横型探索問題のKL1による記述

横型探索問題は，KL1によって一般的に次のように記述できる．

```
search([], In, Out) :- true|
    Out = In.
search([Data|Rest], In, Result) :- true|
    proc(Data, In, Out),
    search(Rest, Out, Result).
```

プロセスsearchは，入力中の各データに対して起動するプロセスである．その第一引数は，入力データのリスト，第二引数はそのプロセスが呼ばれるまでに得られた解析情報を得るストリーム，第三引数は解析結果を返すストリームである．またプロセスprocは第一引数に与えられたデータData一つ分の解析を進めるプロセスである．

各searchプロセスは入力ストリームに解析情報を受けとるとプロセスprocを呼び出す．その解析結果は直ちに出力ストリームに送られ，次のsearchプロセスの入力となる．

3 一般化LRパーザについて

一般化LR構文解析法は，LR法を一般の文脈自由文法に適用するために拡張したアルゴリズムである．解析する文に曖昧性がある場合，これを異なるプロセッサ上で並列に処理することができる．

一般化LRパーザにおいては，プロセスsearchの第一引数は入力語のリスト，第二引数はそれが呼ばれる前までに得られた解析情報を得るストリーム，第三引数は解析結果を返すストリームであり,，これらのストリームにはスタックが流れる．またprocは第一引数のDataに与えられた一語分の解析を進めるプロセスであり，入力文が曖昧性を持つ場合，入力の一つのスタックに対して複数のスタックを出力する．

我々の一般化LR構文解析法の並列アルゴリズムとしての特徴は，

- 横型探索によって，可能な全ての解析木を並列に求める．
- ある一つの解析が失敗すると，そのプロセスは消滅するため，探索空間は単調には増加しない．
- プロセスの分岐位置及び分岐数は，入力文によって異なるため，[3]の方法のように，あらかじめ負荷分散を行なうレベルを設定することは困難である．

4 負荷分散方法

ここでは，パージングプロセスprocの仕様を変えずに負荷分散を行なうことを考える．

[2]の方法は，searchプロセスが自分が使用可能なプロセッサ数を持ち歩き，プロセスprocが全てのデータを出力した後，そのデータ数を数え，そのデータに対する次の処理(次のsearchプロセス)を各プロセッサに割り当てていた．

この方法では，プロセスprocが全てのデータを出力するまで次の解析に進めない．LR法の場合，各入力語ごとに処理(procの実行)を区切ると，解析時間のオーダが上がるという問題点がある[4]．

この問題点を解決するため，第二の方法として，次のような方法を考案した．

- プロセスprocが一つデータを出力した直後に，そのデータの処理(次のsearch)を分散するプロセス(distribute)を先に実行する．
- プロセスprocがデータを全て出力した時点で未使用のプロセッサが存在する場合，先に負荷分散されたプロセスに未使用のプロセッサを割り当て，次のレベルの負荷分散で使用する．

```
search(_, [], In, Out):- true|
    Out=In.
search(PS, [Word|Words], In, Result):- true|
    proc(Word, In, Out)@priority($, -100),
    distribute(PS, Words, Out, Result).

distribute({PP,PN,PI}, Words, In, Result):- true|
    dt(PP, 1, _, _, {PP,PN,PI}, Words, In, Result1),
    merge(Result1, Result).

dt(_, DN, PM, II, {PP,PN,PI}, _, [], R):- true|
```

A Consideration of a Load Distribution for a Parallel Breadth-First Searching Algorithm using Multi-PSI
Tomoo IKEDA, Hiroaki NUMAZAKI, Hozumi TANAKA
Department of Computer Science, Tokyo Institue of Technology

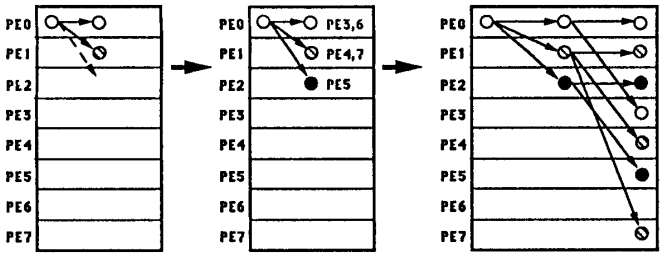

図 1: 負荷分散の例

```
        R=[],
        PM:=PP+(PN-1)*PI,
        II:=PI*(DN-1).
    dt(PE, DN, PM, II, {PP,PN,PI}, W, [H|T], R):- true|
        search({PE,PN1,II}, W, [H], R1)@processor(PE),
        dt(~(PE+PI), ~(DN+1), PM ,II, {PP,PN,PI}, W, T, R2),
        PN1:=(PM-PE)/II+1,
        R={R1,R2}.
```

プロセス search は，第一引数に自分が使用可能なプロセッサの情報を {初項，項数，公差} で表されるプロセッサ番号の等差数列の形で持っている．パージングプロセス proc がストリーム Out に一つデータを出力するごとに，優先度の高い負荷分散プロセス distribute が呼ばれる．負荷分散プロセスは，次のレベルのプロセスを数列の初項で表されるプロセッサから順に割り当てる．パージングプロセスが全てのデータを出力した後に，先に負荷分散されたプロセスに残りのプロセッサを割り当てる．各プロセスには，{はじめに割り当てられたプロセッサの番号，項数/データ数，公差 × データ数} で表されるプロセッサが割り当てられる．プロセッサ数 8 台，第一レベルのパージングプロセス proc が 3 個のデータを出力する場合の例を図 1に示す．

5 実験

今回の実験では，規則数 123 の文脈自由文法を用い，以下の 5 つの入力文に対する解析時間を測定した．使用した計算機は，ICOT で開発された，プロセッサ数 16 台版の Multi-PSI である．

1. Diagram analyzes all of the basic kinds of phrases and sentences.
2. This paper presents an explanatory overview of a large and complex grammar that is used in a sentence.
3. The annotations provide important information for other parts of the system that interpret the expression in the context of a dialogue.
4. For every expression it analyzes, diagram provides an annotated description of the structural relations holding among its constituents.
5. Procedures can also assign scores to an analysis, rating some applications of a rule as probable or as unlikely.

プロセッサ数に対する台数効果の値を図 2に示す．グラフの横軸はプロセッサ数，縦軸は台数効果である．

今回の実験では，最大で約 2.5 倍の台数効果が得られた．図 2の二つのグラフの比較より，本方式の優位性は明白である．

実際の解析中，パフォーマンスメータ (各プロセッサの稼働状態を視覚的に表示するツール) を用いて，プロセッサの状態を観察したところ，同時に稼働しているプロセッサは最大で 7 台程度であった．これには，次のような理由が考えられる．

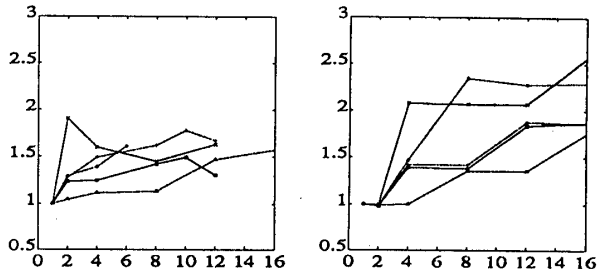

図 2: 台数効果

- 文法のサイズが小さいため，文法の持つ曖昧性がそれほど多くない．
- あるプロセッサ上のプロセスが処理を終えると，そのプロセッサは二度と稼働しない．これにより，負荷の偏りが生じる．
- あるプロセスが自分に割り当てられたプロセッサをすべて使い尽くさない場合，全く稼働しないプロセッサが生じる．

このようなプロセッサの稼働率の問題は，今後，動的負荷分散の実現によって改善することができると思われる．

6 おわりに

本論文では，並列計算機における負荷分散の重要性について述べ，実際の横型探索問題の例として，一般化 LR 構文解析法について負荷分散の実験を行った．実験の結果，負荷分散の方式のわずかな変更によって，台数効果の大幅な改善が見られた．

プロセッサの稼働率をさらに向上させるためには，より有効な負荷分散の方式を検討する必要がある．

参考文献

[1] 沼崎浩明，田村直良，田中穂積．並列論理型言語による一般化 LR 構文解析アルゴリズムの実現．情報処理学会 自然言語処理研究会，NL74-5，1989．

[2] 沼崎浩明，田中穂積．並列一般化 LR パーザの負荷分散の検討．，*KL1 Programming Workshop '90*，123-130 ページ，ICOT，1990．

[3] 古市昌一，瀧和男，市吉伸行．疎結合型計算機上での OR 並列問題に適した動的負荷分散方式とその評価．，*KL1 Programming Workshop '90*，1-9 ページ，ICOT，1990．

[4] 峯恒憲，谷口倫一郎，雨宮真人．文脈自由文法の並列構文解析．情報処理学会 自然言語処理研究会，NL73-1:1-8，1989．