並列一般化 LR パーザ PGLR の Multi-PSI 上での性能評価

沼崎 浩明† 池田 朋男†† 田中 穂積†

†東京工業大学 工学部
〒152 目黒区大岡山 2-12-1 TEL 03-3726-1111 (内線 4175)
††株式会社 東芝, 総合研究所
〒210 川崎市幸区小向東芝町 1 番地

あらまし

本研究では，並列論理型言語 KL1 で記述した並列一般化 LR パーザを，並列推論マシン Multi-PSI 上に実現するための負荷分散方式を開発し，実験より性能を評価している．Multi-PSI を効率良く動作させるためには，プロセッサ間の通信量を抑え，各プロセッサの負荷を均等化する必要がある．本研究では静的負荷分散と動的負荷分散の長所を生かした融合方式に基づく負荷分散を実現している．規則数 180 の純粋な英語の CFG と，これに構文的な制約を与えた文法を用い，13 の文の解析時間を測定した結果，純粋な CFG では，64 台のプロセッサを用いて最大 12.3 倍，制約を与えた場合は，最大 4.5 倍という台数効果を得た．

Evaluation of the Performance of a Parallel Generalized LR Parser on Multi-PSI

Hiroaki NUMAZAKI† Tomoo IKEDA†† Hozumi TANAKA†

†Department of Computer Science, Faculty of Engneering,
Tokyo Institute of Technology
2-12-1 Meguro-ku Ohokayama, Tokyo 152
††Research and Development Center, TOSHIBA Corporation
1 Toshiba-machi Komukai, Saiwai-ku, Kawasaki-shi, Kanagawa 211

Abstract

In this research, we develop a load distribution method for our parallel generalized LR parser implemented in KL1 on Multi-PSI machine with 64 processors. In order to reduce the communications between processors and to balance the load on all processors, our method takes the advantages of both the static and the dynamic load distributions. The experiment using 180 CFG rules for English grammar and the same rules with the augmentations of syntactic constraints revealed that at the maximum we can achieve 12.3 times of speed-up with the pure CFG rules and 4.5 times of speed-up with the augmented CFG rules.

1 はじめに

本研究では，並列論理型言語KL1を用いた並列パーザPGLR(Parallel Generalized LR parser) を構築し，疎結合型の並列推論マシンMulti-PSIを用いてその性能を評価している．PGLRパーザは自然言語の高速な解析を実現するためのパーザとして開発され，文の構文的な曖昧性を並列に解析する．そのアルゴリズムは，効率の良いことで知られているLR法に基づいている．LR法は与えられた文法から定まる全ての解析状態をLR状態遷移図として表し，これに基づく状態推移により構文解析を進める．すなわち，LRパーザは初期状態から，一語一語の入力語に応じて状態間を推移し，入力の終了時に最終状態に到達した時，文を受理する．また，その状態推移の過程で，適用する規則の制約条件を評価することにより，無駄な解析を削減することができる．本来，LR法は文脈自由文法のサブセットであるLR文法に対して，決定的な構文解析を行うアルゴリズムとして開発されたものであるが，これを一般の文脈自由文法に適用したアルゴリズムを一般化LR法と呼ぶ．一般化LR法の解析は，文脈自由文法の持つ曖昧性により，非決定性が生ずる．すなわち，一つの入力語に対して複数の推移先を持つ状態が生成されることがある．その状態は文の解析が曖昧になる場所を示している．PGLRパーザは，曖昧性のある状態に到達した場合，全ての可能性を並列に解析する．

PGLRパーザの効率性は，既にシミュレーションにより評価している[1]．本研究では，これを疎結合型のの並列計算機Multi-PSI上に効率良く実現するための負荷分散方式を開発し，その性能を評価する．

近年，KL1で記述されたプログラムをMulti-PSI上に実現するための様々な負荷分散方式が開発され，その性能が評価されている．自然言語処理においても，既に寿崎らのPAXに対する負荷分散の実現[2]，山崎らのシステムLaPutaについての報告[3]がある．寿崎らの研究と本研究とは，ほぼ同じ枠組に基づいており，比較可能である．この方法はPAXのアルゴリズムに密着した負荷分散方式を開発しているのに対し，本研究が一般のOR並列探索問題に対する負荷分散方式を実現している点が異なる．両者の負荷分散方式の善し悪しを比較するには，本方式をPAXに適用してその性能を評価すれば良い．山崎らの研究は，形態素から文脈に至る解析を統一的な枠組で行おうとする試みであり，対象とする範囲が広い．負荷分散には静的な方式を採用しており，今後さらに改善される可能性がある．

本論文の構成は次のとおりである．第2章では，PGLRパーザの特徴を示す．第3章では，本研究で用いた負荷分散の方式を示す．第4章では，64台版のMulti-PSIを用いてPGLRパーザの性能を評価する．また，Prolog上の逐次型パーザSGLR[4]を一般のワークステーション上で実行した時の解析速度と比較し，Multi-PSI上のPLGRの性能について考察する．第5章では，結論と今後の課題について述べる．

2 PGLRパーザの特徴

PGLRパーザは一般化LR法に基づくパーザであり，以下の特徴を有する．

- 入力文を左から右へと解析する．
- 文の構文的な曖昧性を並列に計算する．
- 重複計算の回避の際に不要な同期を避けるための，Incomplete Stackと呼ぶデータ構造を採用している[1]．
- 文法に与えた制約により，漸進的な曖昧性解消を行なうことができる．
- 全てのプロセスはデータ駆動により動作する．

この並列パーザの計算モデルを図1に示す．このモデルは，一般化LR法の解析過程を11種類のプロセスに区分し，それらをLRパーズ表のエントリに応じて組み合わせ，それをKL1の節として記述したものである．文の解析は，単語の並びに応じてプロセスを動的に結合することにより行う．図2は簡単な解析例を示している．各プロセスは隣接するプロセスとの間にストリームを張り，そこでスタックの受渡しをする．解析の途中で複数のスタックが生成された場合は，それらを並列に処理することができる．

2.1 重複計算の回避

一般に，構文解析では異なる複数のプロセスが同一の部分木を構築する際に重複計算が生ずる．逐次的な構文解析では，効率性にとって重複計算を回避することは必要不可欠であるが，並列処理においては，プロセッサの資源に余裕がある限り，重複計算を許しても文の解析時間は変わらない．プロセッサ間の通信コストが大きい疎結合型の並列計算機を用いた場合は，重複計算を回避するための通信がオーバーヘッドとなり，逆に効率が低下する可能性が大きい．

このような考察から，本研究ではパーズプロセスの数がプロセッサの台数よりも少ない間は重複計算を許し，プロセス数がプロセッサ数を上回り，一台のプロセッサで複数の解析木を構築する必要が生じた時，初めて重複計算を回避するという方式を採用している．すなわち，異なるプロセッサ間では，重複計算を許す代わりに通信を避け，そのオーバーヘッドを回避している．

3 負荷分散方式

負荷分散を考える際にまず定めることは，静的な負荷分散を行うか，動的な負荷分散を行うかということである．静的負荷分散とは，プロセスのプロセッサへの割り付け方を予め定めておく方式のことであり，動的負荷分散とは，実行時に暇なプロセッサを検出して，そこへプロセスを随

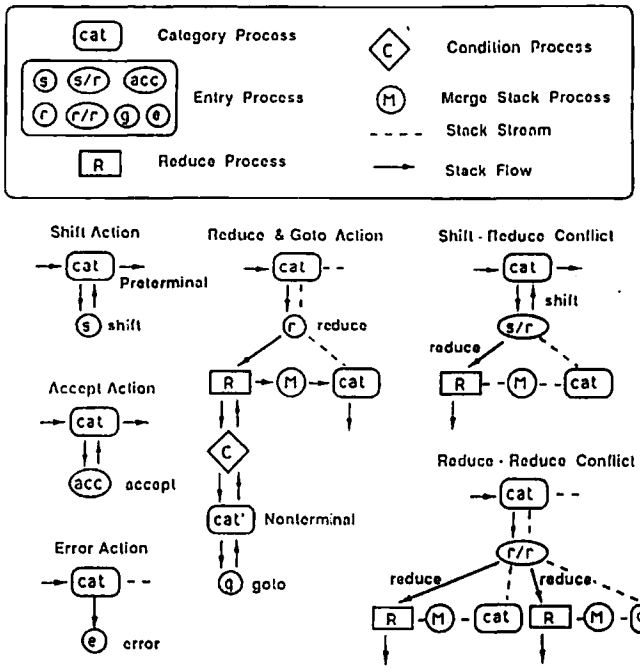

図 1: PGLR パーザの計算モデル

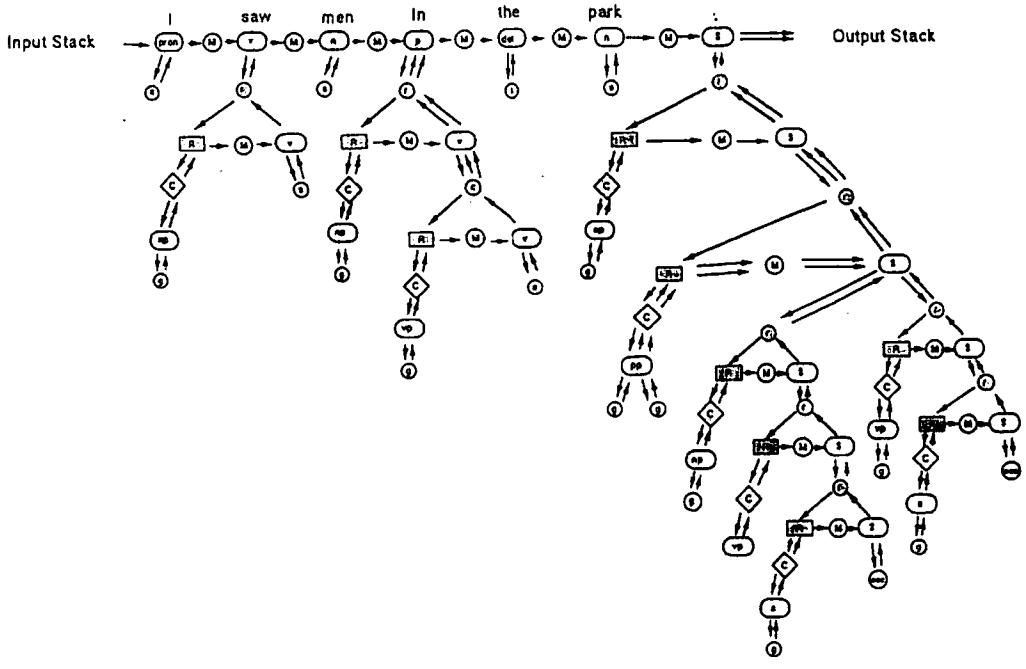

図 2: PGLR パーザによる文の解析例

時割り付ける方式のことである．静的方式は，実現が比較的容易であり，動的方式と比べてプロセッサ間の通信量が少なくて済む．一方，動的方式は実現のコストが静的方式よりも高い反面，負荷の均等化を図り易いという長所がある．

本研究では，一般化LR構文解析の探索問題としての特徴を考慮して，静的な負荷分散と動的な負荷分散とを融合した方式を採用している．すなわち，解析する文の曖昧性が少ない場合は静的負荷分散を行い，負荷分散を実現するプロセスのオーバーヘッドを抑え，曖昧性の数がプロセッサの台数を上回る場合は，動的負荷分散により，暇になったプロセッサに負荷の再割りつけを行う．

また，全ての探索レベルで負荷分散を行なうことも本方式の特徴である．ここでいうレベルとは，入力語を一語読み進むごとに処理を分割するための区切りである．このようなマルチレベルの負荷分散を行う理由は，処理を一語進めるごとに探索空間の幅が増減することによる．すなわち，詰め込みパズルの例[6]のように，負荷分散のレベルを予め定めてしまうと，負荷の偏りが生ずるためである．

以下では，本研究で採用している静的負荷分散方式，動的負荷分散方式，さらにそれらを融合した方式について説明する．

3.1 静的負荷分散方式

本研究で採用している静的負荷分散方式は，池田の考案した方式[5]に従っている．その特徴は，各プロセスが使用可能なプロセッサの資源(プロセッサ番号の集合)を予め確保していることである．あるプロセスが複数のサブプロセスに分かれる際，サブプロセスを自分が確保しているプロセッサ上に割り付ける．全てのサブプロセスを分散した後，使用可能なプロセッサが残っている場合，それらを均等に分割してサブプロセスに使用可能な資源として与える．

3.2 動的負荷分散方式

本研究では，動的負荷分散を導入するために，暇なプロセッサの情報を管理するプロセッサマネージャを設ける．以下これをPMと記する．各プロセスはPMと通信することによりプロセッサ資源の情報を受渡しする．この動的負荷分散方式は，「1プロセッサ・1プロセス」の原則の上に成り立っている．すなわち，あるプロセッサに，プロセスが一つ割り付けられた後，それが稼働している間は，そのプロセッサに他から他のプロセスを割り付けることはできないという原則である．これにより，プロセスの終了をPMに報告することと，PMが暇なプロセッサを検出することが同義となる．

本方式と古市らの動的負荷分散方式[6]とは，暇なプロセッサが存在しない時の処理が異なる．本方式では，プロセッサの獲得要求が出された際，暇なプロセッサが存在しない場合，PMはその要求に対して，暇なプロセッサが存在しないということを報告する．それを受けとったプロセスは，負荷分散をあきらめる．一方，古市らの方式では暇なプロセッサが見つかるまで待つという戦略を採用している．

このような方式を採用した理由は次の通りである．負荷分散を試みたプロセスが，暇なプロセッサが存在せず，それをあきらめた場合，同一プロセッサ上で複数の解析を処理することになる．その際，重複計算を回避することにより，全体の計算量を低減できる場合があり，負荷分散をあきらめたことが効率性にとって有利に働くことがあるためである．

3.3 融合方式

静的方式と動的方式を融合する方式では，曖昧性の少ない文の解析においては，オーバーヘッドの小さい静的方式で負荷分散が行い，曖昧性の多い文の解析においては，動的負荷分散を行うことによって負荷の均等化を図ることができる．

この融合方式のアルゴリズムは以下のように要約できる．

1. 実行の開始時に，PMの持つ暇なプロセッサのリストを空にする．
2. パーズプロセスは全てのプロセッサの資源を確保し，PMとの通信を行うストリームを持って実行を開始する．
3. 処理を終えたパーズプロセスは，自分が確保していたプロセッサの集合をPMに送る．PMは，それをリストに蓄える．
4. プロセッサを使い尽くしたパーズプロセスが，さらに負荷分散を行おうとする際，PMに空いているプロセスが存在するかどうかを問い合わせる．
5. この時，PMは空きプロセッサのリストが空ならば，パーズプロセスにnoneという情報を返す．これを受け取ったパーズプロセスは負荷分散をあきらめる．
6. PMはリストが空でなければ，そこから一つのプロセッサ番号を取り出し，パーズプロセスに返す．パーズプロセスは，そのプロセッサに次のレベルのプロセスを投げる．

図3は，融合方式による負荷分散の様子を示している．ここでは，PE0～PE4までの5台のプロセッサを用いている．負荷分散は次のようにして行われる．

- PMはPE0に置かれ，PE1で解析を開始している．実線はプロセスの実行を示し，破線はプロセスの中断を示している．また，丸で示されたプロセスの左肩の数字は，そのプロセスが使用できるプロセッサの数を示している．

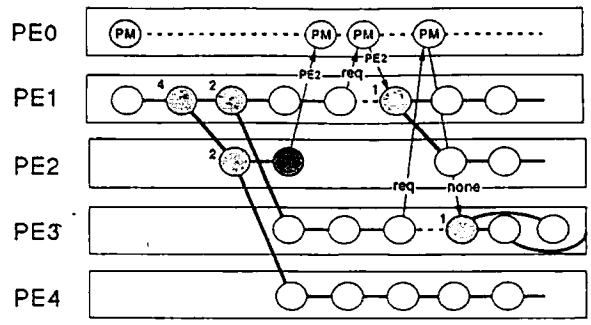

図 3: 融合方式を用いた負荷分散

- 4ステップ目までは静的に負荷分散が進み，PE2のプロセスが実行を終了すると，PE2が暇になったことをPMに報告する．
- PMはこれをリストに保持し，次の要求に備える．
- 5ステップ目でPE1上のプロセスは負荷分散しようとするが，確保しているプロセッサがPE1以外にないため，暇なプロセッサの獲得要求をPMに送る．
- これに対してPMは先ほど受け取ったPE2を返す．
- PE1上のプロセスはこれを受け取りPE2に負荷を分散する．
- 6ステップ目で，PE3上のプロセスが負荷を分散しようとして同様にPMに要求を出すが，今度は暇なプロセッサが存在しないため，PMからnoneという情報が返される．
- PE3はこれを受け取ると負荷分散をあきらめ，複数の解析を行う．その際，重複計算を避けることにより計算量を低減する．

4 性能評価

4.1 実験

本研究で提案する負荷分散方式の有用性を示すため，規則数180の純粋な文脈自由文法と，これにHornbyの文型パターンに基づく制約を与えた文法[7]を用いて解析時間とその台数効果を測定した．さらに，Prolog上のSGLRパーザ[4]をSUN-3/260上で実行した際の解析時間を示し，Multi-PSI上のPGLRの性能を比較検討する．

解析時間の測定には，以下の14の英文を用いている．

1. There are three on the table now.
2. The structural relations are holding among constituents.
3. He explained the example and he gave the rule.
4. This is a film that is developed in the research.
5. Its procedures allow phrases to inherit attributes from their constituents and sentences to get attributes.
6. This paper presents an explanatory overview of a large and complex grammar in a computer.
7. For every expression it analyzed, diagram provides an annotated description of the structural relations holding among its constituents.
8. This paper presents an explanatory overview of a large and complex grammar in a computer for interpreting english dialogue.
9. The man called 'John' presents an overview of a grammar and sentences, that are used in a computer system.
10. The annotations provide important information for other parts of the system that interpret the expression in the context of a dialogue.
11. This paper presents an explanatory overview of a large and complex grammar that is used in a computer for interpreting english dialogue.
12. This paper presents an explanatory overview of a large and complex grammar, that is used in a computer for interpreting dialogues.
13. Its procedures allow phrases to inherit attributes from their constituents and sentences to get attributes from the larger phrases which are constituents of the context.
14. Its procedures called 'P' allow phrases to inherit attributes called 'a' from their constituents and sentences to get attributes from the larger phrases which are constituents of the context.

これらの文に対して，単語数，解析木の数，64台のプロセッサを用いた時の解析時間，reduction数，台数効果を表1，表2に示す．前者が純粋な文脈自由文法に対する値であり，後者が制約を与えた場合の値である．尚，解析時間の欄の括弧内の値は，SGLRパーザによるものである．

表1で，最後の文の台数効果が計測不能である理由は，1台のプロセッサを用いた場合に，Multi-PSIのメモリが不足して解析できなかったことによる．また，理想値が計測できていないものは，ワークステーションのメモリ不足により，その文が解析できなかったことによる．

文	語数	木	時間(sec)	reductiou	台数効果
1	7	3	.131(.017)	798	1.1
2	7	14	.206(.040)	4908	1.9
3	9	66	.569(.800)	24980	3.3
4	10	26	.550(.760)	27138	3.1
5	15	2737	18.1(564)	1448787	8.9
6	15	90	.765(1.64)	48615	5.2
7	19	356	6.86(90.6)	805434	8.1
8	19	1240	7.25(109)	728699	12.3
9	20	2873	70.0(7935079)	4207369	7.8
10	21	926	7.93(13920)	780492	7.25
11	21	22541	150(16814220)	12180235	9.88
12	23	1608	24.8	1768683	10.41
13	25	82294	973	96014714	計測不能

表 1: 実験結果(CFG)

文	語数	木	時間(sec)	reduction	台数効果
1	7	2	.228(.020)	1482	1.45
2	7	2	.307(.060)	11132	2.12
3	9	2	.333(.080)	19603	3.01
4	10	1	.412(.100)	51129	2.39
5	15	22	1.528(.640)	441987	4.08
6	15	1	.304(.080)	9186	2.10
7	19	1	.936(.120)	139143	1.69
8	19	1	1.467(.240)	376687	2.61
9	20	2	1.38(.300)	375348	2.62
10	21	3	1.16(.260)	151894	2.33
11	21	2	1.22(.340)	381459	3.15
12	23	2	1.06(.180)	304941	2.92
13	25	24	5.82(1.78)	1777670	3.43
14	29	98	74.0(16.9)	37240904	4.55

表 2: 実験結果(Augmented CFG)

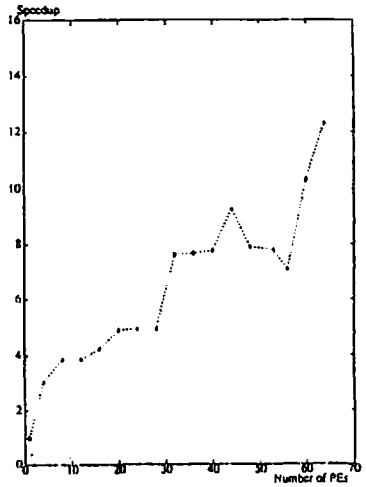

図 4: 文 8 の解析に於ける台数効果の推移

純粋な文脈自由文法に対する解析では，最良値として文8に対する解析が12.3倍の台数効果(プロセッサ数1台の場合と比べた時のプロセッサ数64台の時の処理速度の向上率)を得，制約を与えて曖昧性を漸進的に解消した場合は，文14に対して4.55倍の台数効果を得ている．

使用するプロセッサ数に対する台数効果の推移を明らかにするために，文8と文12に対して，プロセッサ数と台数効果の関係をグラフにしたものを図4，図5に示す．このグラフはどちらもプロセッサ数が増加するにつれ，台数効果も向上する傾向を示しているが，台数効果はプロセッサ数に正比例せず，台数が増加するにつれて，正比例した場合の直線から離れている．これは，プロセスの粒度が小さくなる一方で，負荷分散のオーバヘッドが大きくなるためであると思われる．また，台数効果が途中で増減しているのは，負荷分散が不確定な動作をすることと，実行時のGCの影響と思われる．

一般のワークステーションのProlog上に実現されたSGLRを用いた逐次処理との比較では，曖昧性の数が数十程度の場合はSGLRの方が速く，100以上の場合は圧倒的にPGLRの方が速いという結果を得ている．理想的には並列処理では一つの解析木を作る時間で処理を終えることができるため，逐次処理とは計算時間のオーダが異なり，曖昧性の多い文においては並列処理の方が速いのは期待通りの結果と言える．

曖昧性の数が少ない場合に，ワークステーション上での逐次処理の方が速いの理由は，PGLRのインプリメンテーションがSGLRより複雑であることによるためであり，計算機そのものの性能によるものではない．特に，PGLRの制約条件の評価機構はかなり重い．SGLRでは，Prologのゴールの成功/失敗により実行を制御しているが，PGLRでは，KL1ゴールの成功/失敗による実行制御を実現するため，OSの提供する「荘園」と呼ばれている制御機構を利用しており，この処理がオーバーヘッドとなっている．この点については今後改善する余地がある．

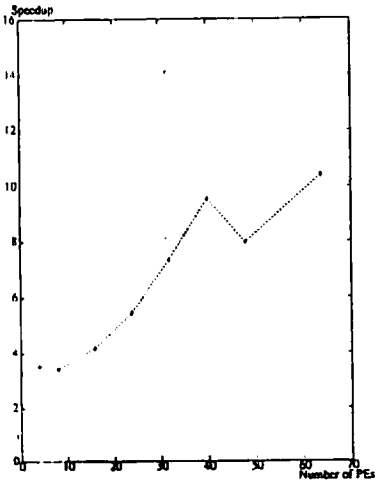

図 5: 文 12 の解析に置ける台数効果の推移

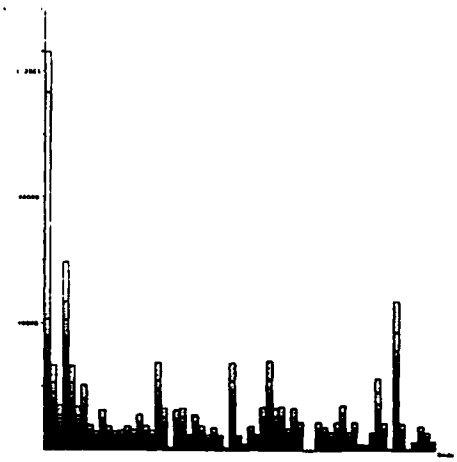

図 6: 負荷の総量の分布 (静的方式)

4.2 負荷の偏りの分析

ここでは，Multi-PSI の OS が有する ParaGraph というツールを用いて，純粋な CFG を用いた文 8 の解析における負荷の分布の様子を調べた結果を示し，静的方式，動的方式，融合方式のそれぞれの特徴を明らかにする．

図 6，図 7，図 8は，静的方式，動的方式，融合方式のそれぞれについて，各プロセッサの負荷の総量を示したものである．図の横軸がプロセッサ番号，縦軸が負荷の総量である．尚，縦軸の尺度はグラフごとに異なっている．

これらを比較すると，静的負荷分散において負荷の偏りが顕著であることがわかる．プロセッサによっては，全く稼働していないものも存在している．また，動的負荷分散において，グラフの右端のプロセッサの負荷が特に大きいのは，ここに PM が存在しているためである．このグラフは，PM に負荷が集中している可能性があることを示している．融合方式では，負荷に偏りがあるものの，特定のプロセッサのみに負荷が集中したり，逆に負荷が与えられなかったりする現象が生じていないと言う意味では，負荷分散に成功していると考えて良い．

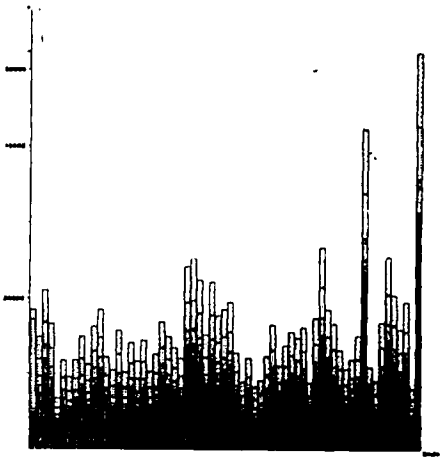

図 7: 負荷の総量の分布 (動的方式)

また，図 9，図 10，図 11は，静的方式，動的方式，融合方式における各プロセッサの負荷を一定の時間で区切って見た場合の様子を示している．各グラフの横軸が時間，縦軸がプロセッサ番号，黒いマスの濃度が負荷の重さを示している．尚，濃度の割り振りはグラフごとに異なっている．

静的方式では，処理が進むにつれて負荷の偏りが顕著になっており，一つのプロセッサ上の処理が全体の処理時間を支配していることが分かる．動的負荷分散では，負荷の時間的な偏りが少ない反面，処理の密度が希薄である．その理由は，PM の処理がボトルネックとなり，実行を中断するプロセスが多いことが予想される．融合方式では，処理の後半で負荷の偏りがかなり認められるものの，他と比べた場合は良い負荷分散が実現されていることが分かる．

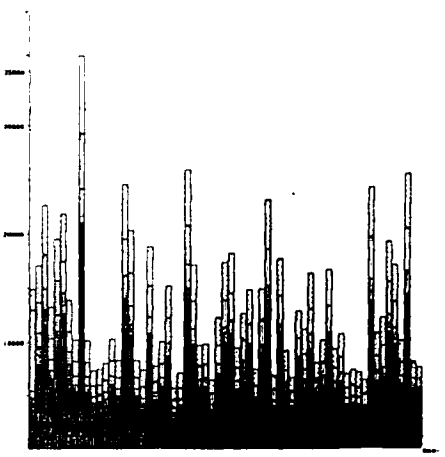

図 8: 負荷の総量の分布 (融合方式)

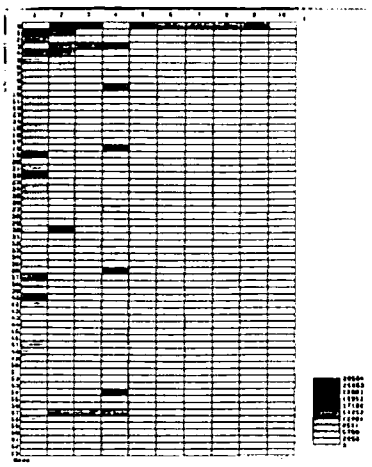

図 9: 負荷の時間的な分布 (静的方式)

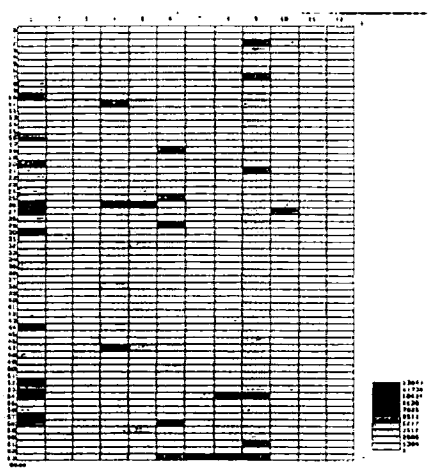

図 10: 負荷の時間的な分布 (動的方式)

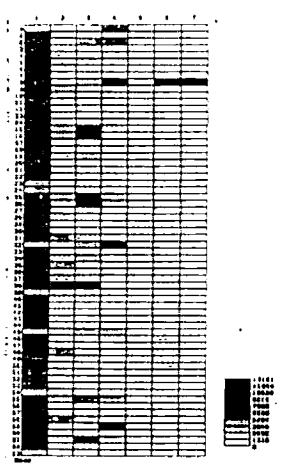

図 11: 負荷の時間的な分布 (融合方式)

5 結論と今後の課題

本研究では，一般化 LR 法に基づく並列構文解析を実際の並列計算機で行い，その性能を測定した．負荷分散の方式として，静的方式と動的方式とを融合した方式を採用することにより，64 台のプロセッサを用いて最大 12 倍という比較的良好な台数効果を得た．また，一般のワークステーションを用いた構文解析と比較により，並列処理の有用性が明らかとなった．

本研究では，一般の探索問題に適用できる形の負荷分散方式を採用しているため，LR パーザ固有の性質を十分に考慮していない．すなわち，パーザが最後の入力語を読み込んだ後に生ずるレデュース動作が，並列実行可能な多くのプロセスを生ずるにもかかわらず，本方式ではこれを負荷分散していない．このため，実行の後半でかなり大きな負荷の偏りを生じていることが，プロセッサの稼働状況の分析により明らかとなった．

今後，この問題をさらに改善することにより，より高い台数効果を得る負荷分散方式を実現できる可能性がある．

謝辞

本研究を進めるにあたり，日頃から御協力を頂いた田中研究室の皆様，ならびに，Multi-PSI を使用させて頂いた ICOT に感謝致します．

参考文献

[1] 沼崎浩明，田中穂積．論理型言語に基づく効率的な並列一般化 LR 構文解析．The Logic Programming Conference, pp. 191–198, 1990.

[2] 寿崎かすみ，他．マルチ PSI における並列構文解析プログラム PAX の実現および評価．並列処理シンポジウム, pp. 343–350, 1989.

[3] 山崎重一郎．並列自然言語解析における並列協調の効果について．KL1 Programming Workshop, pp. 141–146, 1991.

[4] 沼崎浩明，田中穂積．SGLR:逐次型一般化 LR パーザの Prolog による実現．情報処理学会論文誌, 32(3):396–403, 1991.

[5] 池田朋男，沼崎浩明，田中穂積．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．情報処理学会第 41 回全国大会, pp. 123–124, 1990.

[6] 古市　昌一，瀧　和男，市吉伸行．疎結合型並列計算機上での動的負荷分散方式とその評価．KL1 Logic Programming Workshop, pp. 1–9, 1990.

[7] 高倉伸．Prolog による英語のボトムアップ構文解析．1984. 東京工業大学工学部卒業論文．