2D－4　共有メモリ型並列計算機上のFleng処理系におけるオーバーヘッドの定量的解析

馬場 恒彦，荒木 拓也，田中 英彦
{tbaba,araki,tanaka}@mtl.t.u-tokyo.ac.jp
東京大学工学系研究科*

1　はじめに

Committed Choice型言語 Flengは単一代入変数とサスペンド・アクティベイトの機構により，非定型的な問題に対しても並列度を最大限抽出し実行可能な言語である．そのFleng処理系の一つとして，並列共有メモリ計算機上で動作する処理系が稼働している [1]．しかし，問題として一部のアプリケーションで十分な台数効果が得られないことがあり，スケジューリング方式を改善することが必要である．

そこで，本研究では並列共有メモリ計算機上のFleng処理系において，速度向上を阻害している要因について定量的解析を行なうことにより，改善すべき点について解析を行なった．本稿では，その結果について述べる．

2　研究の背景

2.1　細粒度並列処理言語 Fleng

Flengのプログラムは以下のような定義節を並べたものである．

$$Head : -Goal_1, Goal_2, \ldots, Goal_n$$

:-の左側をヘッド部，右側をボディ部という．Flengの計算単位はゴールと呼ばれるものであり，実行可能なゴールが与えられるとゴールと同じヘッド部を持つ定義節が選択される (ヘッドユニフィケーション)．選択されたゴールはボディ部の各ゴールに展開される (リダクション)．展開されたゴールはそれぞれ並列にヘッドユニフィケーション，リダクションが行なわれる．図1にFlengの実行モデルを示す．

2.2　並列共有メモリ計算機上のFleng処理系

共有メモリ計算機上のFleng処理系は，SunOS 5.Xのスレッドライブラリによりスレッドの生成・切替を行ない，マルチスレッドによる並列実行を実現している．また，メモリ空間は全てのスレッドで共有し，GC (Garbage Collection) は一括して全スレッド同時に実行している．メモリ空間やゴールスタックが共有されるため，正当性

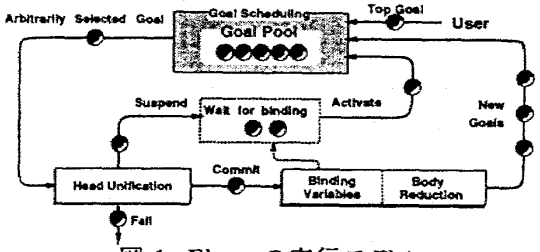

図 1: Flengの実行モデル

を保つためには，アクセス時に排他制御が必要とされる．

スケジューリング方式　共有メモリ型並列計算機上の処理系では，各スレッド毎のゴールキュー (LGS(Local Goal Stack)) と全スレッドによって共有する負荷分散用のゴールキュー (GGS(Global Goal Stack)) の二種類を用い，次の負荷分散によるスケジューリングを実行している (図2)．

Enqueue: 通常時はLGSに対して，新たなゴールをプッシュする．空き (停止している) スレッドがある場合は，LGSの底のゴールを1つGGSにプッシュし，代わりに新たなゴールをLGSにプッシュする．

Dequeue: LGSにゴールが有る場合はLGSからポップする．LGSが空の場合はGGSからポップする．

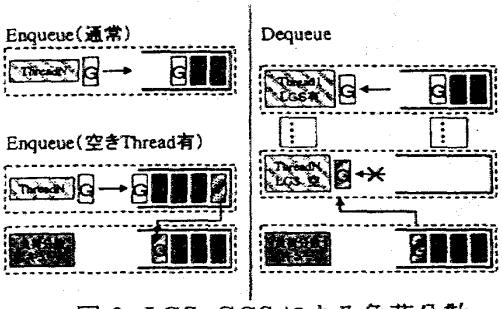

図 2: LGS, GGSによる負荷分散

問題点　ピボット選択を行なわない Gauss 法による連立一次方程式の解を求めるプログラム (以下，Gauss) を行列サイズ 80 × 80 に対して実行すると，4スレッド時に速度の低下が生じる (図3のNormal)．これはシステムタイムの急激な増加が原因となっている (表1)．

これは全ての排他制御機構をBusy Waitに置換することにより，スレッドの起動および停止のコスト (システムタイム) を削減することができる (図3のBusy Wait)．

*Quantitative Analysis of Overheads in Fleng Run-time System on Shared Memory Parallel Computer
Tsunehiko BABA, Takuya ARAKI, Hidehiko TANAKA
University of Tokyo, Graduate School of Engineering,
7-3-1 Hongou, Bunkyo-ku, Tokyo 113, Japan

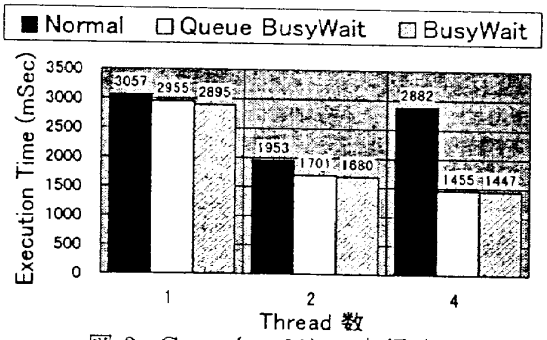

図 3: Gauss(n=80) の実行時間

表 1: Gauss(n=80) の system time(mSec)

スレッド数	1	2	4
Busy Wait	24	32	18
Queue Busy Wait	24	14	18
Normal	20	20	3332

しかし, 通常, 共有メモリ型並列計算機では他のプログラムが CPU を使用するため, Busy Wait は実用的でなく, 占有できない場合にはかえって速度の低下を招くことになる. このため, スケジューリング方式を改善する必要がある.

そこで, 現在のスケジューリング方式を改善するために. 各排他制御機構のどの処理が処理時間に影響を与えているかについて定量的に解析し, 改善すべき点を把握することが必要である.

3 定量的解析

どの排他制御機構がシステムタイムを増大させているのかを解析するために, 各排他制御機構を Busy Wait 機構に置換した場合の速度向上の測定を行なった. プログラムは Gauss(サイズ 80 × 80) を, 計算機は Sun Enterprise 3000(4CPU 構成) を用いた.

Busy Wait に置換した排他制御機構のうち, キューに関する排他制御機構のみを置換した場合の実行時間を図3に示す. Busy Wait に近い速度が得られており, キューの排他制御機構において, システムタイムが増大していることがわかる.

キューの排他制御機構は次の 5 つの場合に行なわれる. 各機構の影響を解析するため, 各システムタイムを 2 スレッド, 4 スレッド時について測定を行なった.

1. En/Dequeue 時の GGS の排他制御 (Queue Lock)
2. 排他制御終了時のキュー解放通知 (Queue Signal)
3. GGS が空の時の Enqueue 待ち (Queue Empty)
4. GC の実行待ち (GC Suspend)
5. GC 終了時のキュー解放通知 (GC Signal)

その結果, GC に関する最後の 2 つの場合については Busy Wait によるシステムタイムの増加は見られなかった. 一方, 他の 3 つの場合が増加したシステムタイムに

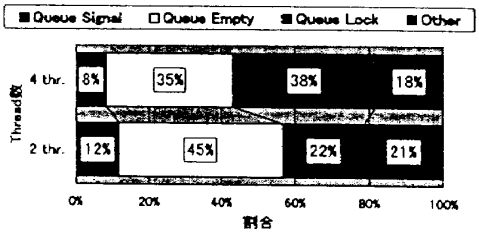

図 4: Sytem Time の増加に占める各排他制御機構の割合

占める割合を求めると図4のようになる. (図中の Other は計測により実行スケジュールが変動した影響である.)

2 スレッド時, 4 スレッド時ともに GGS への Enqueue 時の排他制御 (Queue Lock) 及び, GGS からの Dequeue 時の Enqueue 待ち (Queue Empty) が大きな割合を占めているのがわかる. よって, スレッドが停止している状況が多く, GGS に対するアクセスが頻繁に生じていることが示される.

これは, 空きスレッドが存在する場合にはゴールのサイズに関わらず各スレッドから GGS にゴールがプッシュされるため, プッシュされたゴールのサイズが小さい場合, 動き出したスレッドが再びすぐに停止することが生じるためや, ゴールをプッシュしたスレッドの LGS 内部のゴール数が少ない場合にも同様ににプッシュしてしまうため, 空きスレッドが動き出す代わりに停止してしまうことが起きるためである.

この問題を避けるためには, 各スレッドがすぐに停止しないように十分な処理量を持ったゴールのみを GGS にプッシュすることが必要である. そのためには, Fleng では動的にスケジューリングが行われるため, 動的に実行されるゴールの実行時間を予測・算出することが必要である.

4 おわりに

本稿では, 共有メモリ型並列計算機上の Fleng 処理系において実行速度向上を阻害するオーバーヘッドについて, スケジューリング方式に着目し, 定量的解析を行なった. その結果, キュー処理部の排他制御部分が大きな要因となっており, スケジューリング方式を改善するためにゴールの実行時間を予測・算出することが必要であることを示した. 今後の課題として, その機構を付加したスケジューリング方式の開発があげられる.

参考文献

[1] 馬場恒彦, 荒木拓也, 田中英彦: 共有メモリ型並列計算機上への Fleng 処理系の実装及び評価, 97-PRO-14, Vol.97, No.78, pp.67-72, August,1997.