並列論理型言語処理系 KLIC の通信タイミング最適化

2D−2

福田 茂紀　近山 隆
東京大学　工学系研究科
fukuta@logos.t.u-tokyo.ac.jp

1　はじめに

並列論理型言語 KL1 は、データフロー制御を用いているため、通信・同期処理の記述が容易である反面、処理実行順序を動的に決定しつつ処理を進める必要があるため、ある程度実行が進むまでワーカ間通信内容を決定できない。そこで KL1 の処理系 KLIC では通信量を減らすために、各データは必要になるまで送信されない。このため、通信待ち合わせのオーバヘッドが、他の並列逐次型処理系に比べて生じやすかった。

この問題に対処するため、KLIC は使用する可能性のある作成済データをすべて送信するモードを用意しているが、この場合は逆に無駄なデータ送信が多くなり、通信バッファも多く必要とする。

そこで今回は、KL1 プログラムの静的解析を行ない、必要なデータのみ一括して送信を行なう処理系を作成した。本稿では、この処理系と従来の KLIC による実行時間を比較し、データ待ち合わせのオーバヘッド削減の有効性を示す。

2　並列論理型言語処理系 KLIC の通信方式

KL1 は、プログラム全体をゴールと呼ばれる部分処理にわけ、これらゴールがすべて並列動作することを前提として作られた言語である。この特徴のため、プログラマが並列実行の仕様変更するのが容易になるが、通信機構はどのゴールに対するワーカ間通信にも対応できなければならない。この問題に対処するために KL1 では、データフロー同期機構を導入して同期を自動化し、プログラマによる同期の誤りが入り込む可能性を排除している。

2.1　Lazy 通信

論理型言語の処理は非決定的に動作するため、用いるデータの内容や量を予測することは容易ではない。また、KL1 によるプログラムでは、深く構造化されたデータを多用する。そこで KLIC は通常、具体的なデータが必要になるまで通信を行なわない。

たとえば別ワーカの持つ input(1, 2, data(3, 4)) という 3 引数のファンクタ構造を参照する際は、まず「3 引数の input というファンクタ構造」であることと、非構造データ (ここでは 1, 2 という整数) のみを送信する。そして、具体的に受信側ワーカで内部データが必要になったときに、data(3, 4) という内部の構造データを送信する。

これを Lazy 通信と呼ぶ。Lazy 通信は参照されないデータを無駄に送信することがないが、データ送信回数が増え、それに伴うデータ要求通信や、データ待ち合わせ処理のオーバヘッドが大きい。

2.2　Eager 通信

KLIC には通常の方式とは別に、送信可能なデータはすべて送ってしまう通信方法も用意されており、プログラム実行時に使い分けることができる。この方式では、構造データを送る際に、作成済みの内部データは一括して送信する。

こちらを Eager 通信と呼ぶ。Eager 通信は使用されないデータまでも送信してしまうことがあり、データ送信量が不必要に増えるてしまう。さらに送信データの構造が深い時にも、そのすべてを一括して送信するために、通信バッファをより多く必要とする。

3　ワーカ間通信最適化手法

3.1　モード解析方法

今回の解析においては、プログラムに出現する変数の位置をノードとし、必ず受信側ワーカで実行されるゴールに参照されるノードを Eager に通信すべきであるとした。このとき、受信側ワーカ以外で実行され得るゴールは、データを間接的に送信している可能性があるため、参考にしなかった。

3.2　通信手法

今回の実験によるワーカ間通信は、各ノード毎に Eager 通信を行なうか、Lazy 通信を行なうかを処理系に与えた。

図 1に示すようなゴールを別ワーカに通信する場合、図のように引数内部の構造を再帰的に解析し、すべての構造データの通信方法を指定する。

この図の場合、実線で示されている通信が Eager に、点線で示されている通信が Lazy に行なわれることを示している。

Optimizing timing of message passing on KLIC
Shigeki FUKUTA and Takashi CHIKAYAMA
School of Engineering, The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656, JAPAN

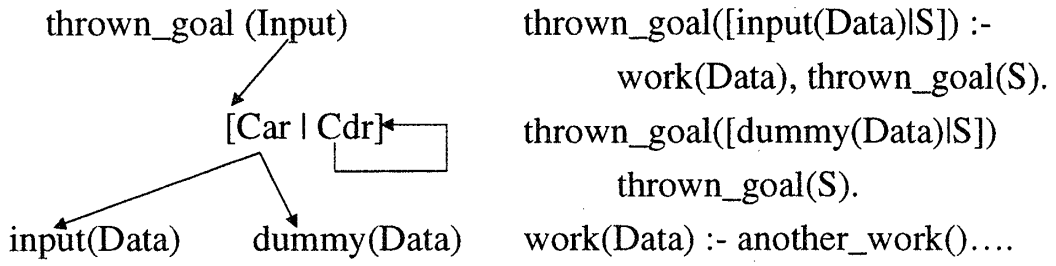

図 1: アクセスパターングラフ

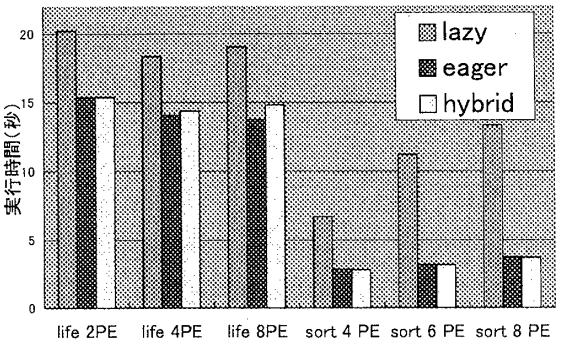

図 2: 実行時間比較

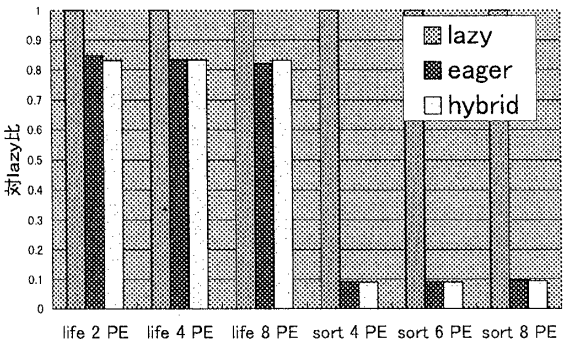

図 3: メッセージ数比較 (lazy 通信を 1 とする)

4 実験結果・評価

本稿では、二つのプログラムに対して従来の KLIC による実行時間と新方式による実行時間を示す。

一つは、一定の深さの二分木の葉にデータを作成し、それらを別ワーカでソートするプログラムである。こちらは、一度に送信しうるデータが多く、ごく単純なプログラム例である。

もう一つは、コンウェイのライフゲームの計算を y 軸方向に分割して並列に行なうプログラムである。こちらは、一度に送信するデータが少なく、比較的複雑なプログラム例である。

4.1 実験結果

テストを行なったプログラムのうち二つのプログラムの結果を、以下に示す。

なお、テストは計算機は Sun Ultra Enterprise E4000 , OS は Solaris 2.5.1 で行なった。並列動作は共有メモリ計算機上でメッセージパッシングを行なっている。

実験結果を、図 2と図 3に示す。多くの場合、実行時間は eager ¡ hybrid ¡ lazy となっている。これはアクセスパターンを参照するオーバヘッドがかかるためだと思われるが、ごく単純なプログラムに対しては、通信内容を十分に解析でき、 eager よりわずかに早くなるケースもあった。これはアクセスパターン参照によるペナルティよりも送信データ削減の効果の方が大きかったためと思われる。

4.2 実験結果の考察

KLIC の Lazy 通信は通信単位が高々 100Bytes 程度の小さなデータが多いため、 KLIC が現在対応している通信方法 (PVM や shared memory による通信) は多くの場合、通信帯域:throughput は十分に大きいと考えられる。より計算に与える影響が大きいと考えられるのは、通信遅れ:latency であり、 Eager 通信はこの通信遅れを大幅に小さくすることを期待できる。

今回の新通信方式は、 Eager 通信の問題である送信量の増加を抑えることに成功し、従来の Eager 通信では実行できないサイズのプログラムに対しても適用できた。

解析処理にも通信ポリシーによって、なるべく Eager に送信する解析視点、なるべく Lazy に送信する解析視点などが考えられ、各プログラムに対する最適な通信ポリシーはさらに考察の余地がある。

参考文献

[1] 上田 和紀: 平成 9 年度　委託研究ソフトウェアの最終成果報告書 (9) KL1 プログラム静的解析系. http://www.icot.or.jp/AITEC/FGCS/funding/97/saishuu09.html

[2] 関田 大吾: Inside KLIC. http://www.icot.or.jp/AITEC/COLUMN/KLIC/inside/master.html