4N－7

分散視覚化プログラム言語における インタラクティブ操作について

久野 英治　石原 昇　寺町 康昌　渋沢 進　小林 正樹
職業能力開発大学校　茨城大学

1 はじめに

PVM[4]が提供され、LAN環境があれば並列計算機システムを仮想的に利用でき、様々な並列プログラムの研究が促進されつつある。その一方で、並列プログラムの分割やそれに付随したスケジューリングなどの動作を解析するツール等の研究が各所で行なわれている。このようなツールは、プログラムの再分割や再スケジューリングを施して[3]ハードウェアの性能を効率的に引出せる。これらのツールの中には、プログラムをグラフで表現し、処理単位である粒度の大きさを様々に変えて（granularity packagingと言う）、実行環境の変化を見ることのできるものもある。しかし、これらはいずれもプログラム実行終了後のモニタの履歴やプロファイラによる実行過程の再現を行なった後に、granularity packagingなどや再配置の検討を行なうものである。

一方、プログラム記述力を高める手法として注目されている視覚化プログラミング(VP: Visual Programming)環境を、テキストによるプログラム開発支援やあるいは代わりプログラム開発[1][6]に用いている。さらにPrograph[1]は、作成したプログラム実行の過程を視覚で追うことが可能である。

これらに対して、本稿ではデータフローモデルをベースとしたDCVL(Dynamic Concurrent Visual Language)言語を提案する。この言語は分散環境におけるコンソール及びインタプリタを提供し、プログラムの部分実行は環境下の各計算機で行なわれ、コンソール上でインタプリタの動作をアニメーション表示する。また、グラフレベルで表現されたプログラムを動作中にインタラクティブにスケジューリング等を操作できるgranularity packaging機能を加える。これを実現することで、実行中ないし実行途上で負荷が偏る場合の分散を実現できることになり効率的にプログラムが動作することを期待できる。

2 DCVLの概要

データフローモデルはデータの通路となるエッジとオペレーションを格納するノードからなる[2]有向グラフで表現される。オペレーションは、ノードの入力エッジ上に全てのデータが揃ったところで実行され、実行結果を出力エッジにセットする。この結果、ノードは入力エッジに必要とされるデータが揃ったものから順次実行される。エッジとノードの接続は依存の流れとして表現される。従って、本モデルをベースとしたDCVLは、依存がオペレーションの流れに見い出せるノードの集合を分離して、これを分散（並列）処理の単位であるプロセスノードとして表現する。

以下、DCVLの規約を示し、次にDCVLインタプリタについて述べる。

2.1 規約

ここでは、アークを流れるデータの型及びノード内のオペレーションについて概観する。なお、ここの規約はインタプリタ上のコンソール（プログラム）ウィンドウで記述することを念頭に置いている。

まず、データ型として、整数型、固定小数点を用意（文字型及び論理型は整数型のサブセット）する。また、インタプリタ上ではデータを格納する共通の領域（エリアと呼ぶ）が暗黙のうちに設定され、数値インデックスであるアドレスないし、別名で定義してアクセスすることができるものとする。その結果、構造データアクセスについては、直接アドレスにインデックス操作という演算を施すものと、"名前"と"インデックス"を直接指定するものとに分けられるものとする。特に、意識しなければ名前のみの場合、インデックス値は0である。

次に、ノードの形状とオペレーションを分類する。表1.に示すように、表現レベルは二階層である。最も低レベルのオペレーションとしてプリミティブレベルが用意され、このレベルのノード等を組合わせて一つの集合としたものがハイブリッドレベルのノードとする。プリミティブレベルではエッジの入出力の数は予め定められた数を越えないものとする。

演算やエリアアクセスは二入力一出力となり、入出力機能は一入力一出力となる。定数ノードは直ちに出力エッジにセットされるため入力エッジはない。制御ノードとして用意されるオペレーションについて触れる。コピーは、エッジの出力先を増やすために用いられ、一入力二出力となる。マージは二入力一出力となる。いずれか一方のデータが到達したら直ちに出力する。三入力一出力のセレクタは論理値によって二入力のいずれかを選択して出力。スイッチは、二入力二出力だが、入力時の論理値によっていずれか一方の出力を選択する。なお、制御オペレーションは後述する繰り返しや条件分岐などの表現に吸収される場合がある。

レベル	種別	入/出	備考
primitive	四則演算	2/1	整数、固定小数点
	論理演算	2/1	論理
	エリアアクセス	2/1	整数、固定小数点
	入出力	1/1	
	定数	0/1	
	コピー	1/2	
	merge	2/1	
	selector	3/1	
	スイッチ	2/2	一方を選択出力
hybrid	マクロ	多/多	条件、場合分け 繰返し 構造データ操作 文字列操作 ノードの組合せ
	関数	多/多	
	プロセス	多/多	

表1: ノードの分類

次に、通常のテキストプログラム言語と同様なマクロ表現、関数表現及び分散処理単位であるプロセス表現を用意する。ノー

Interactive Manipulation on Distributed Visual Programming Language
KUNO, Eiji., ISHIHARA, Noboru., TERAMACHI, Yasuak
University of Industrial Technology
4-1-1 Hashimoto-dai,Sagamihara,Kanagawa 229,Japan
SIBUSAWA, Susumu., KOBAYASHI, Masaki
Ibaraki University
4-12-1 Naka-narusawa, Hitachi, Ibaraki 316, Japan

ド内部のグラフ表現はノード名の設定されたプログラムウィンドウに記述でき、これらを用意することにより、プログラムのスケーリングアップ問題をある程度回避できる[5]。なお、これらのノードにおけるアークの数は、いずれも多入力多出力を許すものとする。

マクロ表現には、システムが提供する繰返し、条件分けや場合分けが用意される。この表現を用いて先に述べた制御オペレーションが省略可能となる。ここで用いられる条件は、マクロノードを開くと、少なくとも二つのウィンドウが得られ、そこに条件を記述しなければならない。また、必要に応じてそのフレーム内にグラフプログラムを記述する。他には、名前付き構造データアクセスや文字列操作(接続、比較、コピーと分離)もシステムから提供されるものがある。それ以外は自ら名前を定義したマクロを設定することが可能である。手続き表現についてはマクロ表現と設定は変わらないが、実装レベルで異なるものとするためにノードの表現形態は変えてある。先に述べたように分散(並列)処理の単位であるプロセスノードも同様に形態の表現のみ変えるものとする。このノードのウィンドウには、プロセス名だけでなく計算機識別子を設定できるが未定義状態の場合は、コンソールが動的設定する。なお、エッジによって接続された異なるプロセスノードが異なる計算機間に跨った場合、エッジを解したデータ交換は計算機間の通信規約に従うことになる。従って、できるだけ接続されるエッジの数が少ないにこしたことはない。

2.2 DCVL インタプリタ

前述した規約に従って記述されたプログラムを直接実行してくれるのがDCVLインタプリタである。このインタプリタは、以下のようなソフトウェアサブモジュールからなるプログラム開発環境である。なお、インタプリタはプログラムコンソールだけでなく利用可能となる計算機あるいはプロセッサが個別に常時表示され、稼働中との区別が容易にできる。

- エディタ：ノードとアークの組合せてプログラムを作成でき、コンソールウィンドウと一致。
- インタプリタ：エディタで作成したプログラムを実行する。
- デバッガ：エディタで作成したプログラムをデバッグする環境を提供する。

インタプリタ及びデバッガはプログラム実行時における振舞いをアニメーション機能によって表現できる。また、プログラム実行前や実行中にインタプリタと異なる計算機にプロセスノードが送られる場合、設定が完了するまで動作できないことに注意が必要である。

これらの環境を開発するプラットホームは、IBM PC/ATベースのNEXTSTEPである。インタプリタの分散環境をMachのスレッドを用いて実現する。詳細は後述するが、インタプリタにはインタラクティブに分散処理単位のブロック化操作を行なって、分散実行による動作確認とインタラクティブに操作したプログラムの実行効率などを比較できるようにする。

3 動的な粒度パッケージング

データフローモデルはエッジの接続による依存関係の度合の高いものほど処理単位として一つにまとめやすい。その一方で、まとめられた処理の中に他のプロセスノードの結果待ちとなるノードがあると、まとめたプロセスノード全体の稼働率を落す可能性もある。

ここでは、DCVLインタプリタに加える動的なgranularity packaging機能(以下、pkgと呼ぶ)について述べる。この機能は、実際には手で行なうスケジューリングであり、実行中のプログラムエディットも意味する。変更内容は変更前のそれと論理的に同等であることを保証しなければならない。また、分散環境であることとDCVLがインタプリタであることから、この提案に至った。しかし、実行中の有効な処理に至る時間のみを換算することで以下で実現する機能の自動化する際の予測値が得られるのではないかと期待している。

具体的な機能は以下の通りである。

- プロセスノードの識別子書換えと、その結果生じるプロセスノードの転送
- プロセスノードの分割。この場合、一方のノードは他の識別子を与えるべき
- 二つ以上のプロセスノードの統合。一つに定められた識別子とプロセスノード名が与えられる。統合するまでの間、プロセスノード(n)と言う名称が自動的に設定される。
- プロセスの分割と、分割されたプロセスノードと他のプロセスノードとの統合。これまで、述べた処理が終了するまで続く。

いずれの処理も、他の計算機/プロセッサ間のプロセス及び関連データの移転が発生し、その間関連するプロセスノードの処理が停止ないし抑制される。また、異なる識別子間に跨るエッジはプロセッサ間通信に切替えられ、その逆の場合は単なるデータ交換処理に変わる。

4 おわりに

現在、DCVLインタプリタは分散環境で手操作でpkgを行なうため大きなトラフィックを発生することがない。そこで、この可能性があるが、pkgを実時間モニタと組み合わせての自動化の検討を行なっていきたい。

参考文献

[1] The Gunakara Sun Systems Ltd., *Prograph Reference*, TGS Systems Limited, Canada, Jan. 1992.

[2] Daniel D. Hils, *Visual Languages and Computing Survey: Data Flow Visual Programming Languages*, Journal of Visual Languages and Computing, Academic Press Ltd., Mar. 1992, pp. 69 - 101.

[3] Kai Hwang, *Advanced Computer Architecture: Parallelism, Scalability, Programmability*, McGraw-Hill, Inc., 1993, pp. 61 - 70.

[4] Al Geist, et. al., *PVM: Parallel Virtual Machine - A Users' Guide and Tutorial for Networked Parallel Computing*, The MIT Press, 1994.

[5] Margaret M.Burnett, et. al., *Scaling Up Visual Programming Languages*, IEEE Computer, Mar. 1995, pp. 45 - 54.

[6] Takayuki D. Kimura, *Object-Oriented Dataflow*, Proceedings 11th IEEE Int. Sym. on Visual Languages, Darmstadt, Germany, Sep. 1995, pp. 180 - 186.

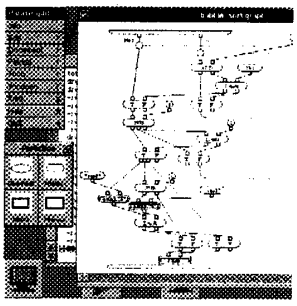

図 1: DCVL イメージ