超並列V言語とそのマルチスレッド実行方式の概要

3D−8

日下部茂　高橋英一　谷口倫一郎　雨宮真人

九州大学総合理工学研究科

1　まえがき

V言語はデータフロー向き関数型言語 Valid[1] を拡張したもので，依存関係のないものは全て並行動作することを前提とした言語である．計算の進行する順序を規定するのは依存関係だけである．依存関係を持つものの間で頻繁な同期が必要となるが，データフロー同期方式により同期は暗黙のうちに行なわれ，プログラマが明示的に同期を指定する必要はない．ある計算に必要な値がまだ求まっていない場合は，その値が決まるまで実行が中断し，値が求まった時点で自動的に同期がとられ計算が再開する．

V言語では，自律して並列に動作するプロセスとしてagentインスタンスを生成し，並列オブジェクト指向風の計算を行なうことができる．agentインスタンスはカプセル化された内部状態を持ち，お互いの間でストリームを通じてデータをやり取りしながら計算を進める．インスタンス間の同期もデータフロー同期方式によって行なわれる．送信側では通信路に次々にデータを流し，受信側では通信路の出口からデータを読みとって計算を進める．データが到着していなければ読みとり側のプロセスは待たされ，データが到着すると待ちが解かれ処理が進む(メッセージ駆動 or データ駆動)．

ストリームの要素をメッセージとみなせば並列オブジェクト指向と考えることが出来るが，処理内容がメッセージだけで決定するなら，Validの枠組での関数呼び出しで済む．ここでagentを導入したのは，並行に動作し内部状態を持ったプロセスと，それらの間の自由度の高い通信を容易に記述することが目的である．出来る限り関数性を保持するため，agentの内部で状態を保持するためには，再帰で明示的に状態をフィードバックする．

本稿ではagentを中心にV言語の特徴とその処理系，並列実行モデルの概略について述べる．V言語プログラムは命令レベルの細並列処理も可能だが，ここでは種々の並列計算機上での実現を意識したマルチスレッド並列実行モデルについて述べる．

2　V言語の特徴

ここではV言語の，Validと共通する部分は省きagentの記述に関連した事項を中心にV言語の特徴を述べる．

2.1　agent定義

```
agent-def::="agent"name"("parameters")"interface"="body-exp
name ::= ident
parameters ::= ⟨ ident ⟩...
interface ::= channel-part [export-part]
channel-part ::= {"channel" ⟨ ident ⟩ ... }
export-part ::= {"export" ⟨ ident ⟩ ⟨ ident ⟩ ...}
```

ここで⟨ x_0 ⟩... はxの0回以上の繰り返し，⟨ x_0 ⟩⟨ x_1 ⟩... はxの1回以上の繰り返し，[ ]は囲まれたものが省略可能であることを示す．parametersはagentインスタンス生成時に渡す値で各インスタンスに固有のものである．interfaceのchannel-partは入力ストリームの変数を，export-partは出力ストリームの変数を宣言するためのもので，その変数のスコープはインスタンス内である．またparameters，interfaceの変数には型指定を行なう(省略時はリスト型)．body-expには正しい式であれば任意の式が可能だが，通常はparametersを通して渡された値によりインスタンス内の環境を初期化し，状態値とストリームを再帰的に処理する式が書かれる．

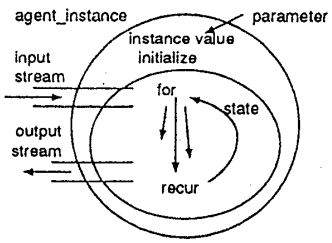

図1: agent

2.2　メッセージパッシングとストリーム結合

send式を用いてそのagentインスタンスの入力ストリームの最後尾にデータをアペンドすることが出来，自由度の高い通信が可能である．複数のsendは暗黙のうちにマージされる．入力ストリームをメッセージキューとみなせば，非同期メッセージパッシングに対応する．

```
x=send(agent_instance,message)
```

これによりagent_instanceのストリームにmessageが送られる．この場合，xに依存する箇所が継続点として指定され，結果が返されるまでxに依存する計算は中断し，結果が返ってくると計算は再開する．

出力ストリームであるexport変数を他のagentインスタンスの入力ストリームであるchannel変数に接続することによってagentインスタンス間を結合することができる．このときexport側のインスタンスの出力ストリームと結合先のagentインスタンスの入力ストリームは共有され，export側で値を書き込めば結合先のagentインスタンスではその値を読むことが出来る．超並列処理の場合，ある程度規則的な通信トポロジーの問題構造を持つ場合が多いと考えられるが，このような方法で問題の持つ構造を反映した記述が容易になる．

3　コンパイラ

コンパイルは2つのフェーズからなっている．フェーズIではV言語のソースプログラムをデータ依存関係にもとづき，Datarolモデル[2]を拡張した中間言語表現に変換する．中

Massively Parallel Programming Language "V" and It's Multi-threaded Implementation

Shigeru KUSAKABE, Eiich TAKAHASHI, Rin-ichiro TANIGUCHI and Makoto AMAMIYA

Graduate School of Engineering Sciences, Kyushu University

間言語は，各命令に対応するノードが，依存関係を表すアークで結ばれた半順序関係のフローグラフである．フェーズIIではグラフを次の節で述べる並列実行モデルをベースにターゲットマシンのアーキテクチャの特徴を考慮して命令列をスケジューリングしターゲットマシン用のコーディングに変換する．

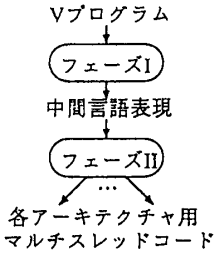

図 2: compile

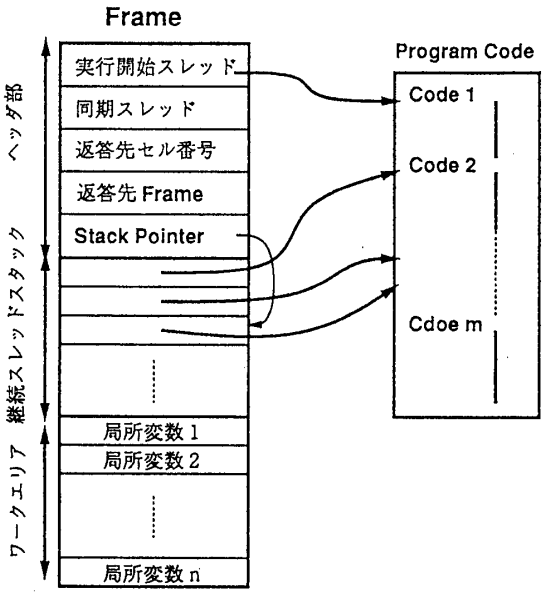

図 3: Frame の構造

4 並列計算モデル

V 言語の並列実行モデルは，Valid の並列計算機上でのマルチスレッド実行方式をベース [3] に Datarol モデルを拡張したマルチスレッド実行モデルである．V 言語の agent を実現するため考慮しなければならない主要な点は以下の通りである．

- agent インスタンスの生成
- ストリームの受渡し
- 実行スケジューリング
- 状態の保持

agent インスタンスは Frame として動的に生成される．図3に示すように Frame は，ヘッダ部，継続スレッドスタック，ワークエリアより構成される．

ヘッダ部は，実行開始スレッド，同期スレッド，ストリーム情報，要求側 Frame へのポインタなどを保持する．実行開始スレッドは，プログラムコードセグメントの実行開始アドレスで，同期スレッドは返答値によって，実行可能とすべき要求側 Frame のスレッドコードアドレスである．継続スレッドスタックは，自 Frame 内での実行可能なスレッドのコードアドレスを保持し，1 つの Frame 内でのスレッド実行制御に用いる．ワークエリアは局所変数を保持する．

send 式によるメッセージのストリームを受けるインスタンスにはメッセージのデータの取り込みや継続点の保持処理などを行なうメッセージハンドラを用意する．メッセージの送信側は，データとともに継続点としての Frame，同期スレッドをメッセージとして受信側のハンドラに送る．

結果値の返答時には，要求側に対して同期スレッド，Frame アドレスおよび返値の情報を含むメッセージを返信する．要求側では，上記のメッセージを受け取る (メッセージバッファから取り出す) と，以下のように同期スレッドを実行する．(1)Frame アドレスが示す Frame のをカレント Frame とする．(2) メッセージから返値を Frame へ転送．(3)join 処理．

同期変数は同期の対象とするスレッドの数に初期化されており，join 時に-1 される．join 時，もし，同期変数が 0 になれば，次スレッドを継続スレッドスタックへ push する．

インスタンスのストリームを結合し共有したストリームを通して通信を行なう場合，ストリーム要素の型情報，ストリームの共有パターンとターゲットアーキテクチャによっていくつかのストリームの実現方法が考えられる．例えば上述の send 式の場合と同様の方式をとっても，この場合は通信相手が特定されており，特にストリーム上を単純なデータが流れる場合は共有ストリームのハンドラの実現コストは高くない．またIストラクチャ上にストリームを実現できる場合はメモリ上で同期をとることが出来る．

状態値は明示的にフィードバックし，ストリームの要素は先頭から着順に再帰ループで処理する．意味上は繰り返しを unfold して Frame を割り当て，再帰変数として渡される状態値などのデータ依存関係が許す限りストリームの複数要素に対する計算をオーバーラップさせることも可能だが，実現コストの問題で 1 つの Frame 内のループで処理する方式をとっている．

5 まとめ

超並列 V 言語の agent を中心に記述とそのマルチスレッド並列実行方式について述べた．本並列実行方式をベースに，Sequent Symmetry，AP1000 上に各アーキテクチャ特性を考慮した V 言語の実装を行なっていく予定である．

参考文献

[1] 長谷川，雨宮.“データフローマシン用関数型言語 Valid” 電子情報通信学会論文誌 D Vol.J71-D No.8 1532 (1988).

[2] 立花，谷口，雨宮. “データフロー解析による 関数型言語 Valid のコンパイル法”，情報処理学会誌 Vol.30, No.12, p.p.1628-1638 (1989)

[3] 高橋，谷口，雨宮.“関数型プログラムの疎/密結合並列計算機上の実行スケジューリング手法”，情報処理学会 PRG 研究会 (SWoPP'93) 発表予定

[4] D. E.Culler, et.al.“Fine-grain Parallelism with Munimal Hardware Support: A Compiler-Controlled Threaded Abstract Machine.” Proc. of 4th Int. Conf. on Architectural Support for Programming Languages and Operating Systems, (1991)