5R-03 スティール評価法のための SST マシンと並列構文の実現*

宮川 伸也　　伊藤 貴康†

東北大学大学院情報科学研究科‡

1 はじめに

スティール評価法[1]は，並列Lisp言語におけるプロセス過剰生成を抑制する効率のよい並列評価法である．スティール評価法を実現するためのSST（Stealable Stack）を備えた並列仮想機械SSTマシンが伊藤により提案されている．本稿では，SSTマシンの1つの設計とSSTマシンによる並列構文の実現方法を説明し，その実行結果を報告する．

2 PaiLisp とスティール評価法

PaiLispは，Schemeに**pcall, plet, par-or, future, stealable**などの豊富な並列構文を拡張した並列Lisp言語である．PaiLispを始めとする並列Lispでは，リンク構造からなる環境（データ）と評価に当って必要な他の情報から構成される共有情報を用いて式が並列評価されるので，共有メモリを利用した実現となっている．並列評価には，通常，プロセスqueueを用いてeagerにタスク生成を行うETC（Eager Task Creation）法が用いられるが，ETCはプロセス過剰生成による効率が低下するという問題がある．スティール評価法（SHE）は，並列評価したい式をSSTに入れ（push），ホームプロセッサはSSTのトップから式を取り出し（pop），空状態のプロセッサはSSTのボトムから順に式を取り出し（steal）て評価する方式であり，これによりプロセス過剰生成問題を解決することができる．PaiLisp/MT[2,3]によってスティール評価法の有効性が示されている．

3 SST マシンの設計

スティール評価法に基づく並列構文の系統的な実現法を与えるため，SSTとその基本命令によりタスク生成を行う並列仮想機械SSTマシンを設計した．

3.1 SST マシンの基本命令

SSTに対する基本操作命令を**表1**に与えた．表1の命令はPaiLispのスティール評価法に基づくタスク生成を実現するための十分な機能を備えている．更に，投棄計算とコンティニュエーションの効率のよい実現のために**表2**に示すSSTのflush-out[4]を行う命令とSSTの内容の移動とコピーの命令を導入している．

表1: SST に対する基本操作命令

SST_push(reg1,reg2)	SST に reg1 の内容を push し，その位置を reg2 に代入する．
SST_pop(reg)	SST のトップの要素を pop して reg に代入し，その要素がなければ偽の値を reg に代入する．
SST_steal(node,reg)	プロセッサ node の SST のボトムから取り出した要素を reg に代入し，その要素がなければ偽の値を reg に代入する．
SST_ref(ptr,n,reg)	SST の ptr から n 個下の位置にある要素を参照して reg に代入し，その要素がなければ偽の値を reg に代入する．
SST_delete(ptr,n,reg)	SST の ptr から n 個下の位置にある要素を取り出して reg に代入し，その要素がなければ偽の値を reg に代入する．

表2: SST の flush-out，移動，コピーに関する命令

SST_flush(ptr,n)	SST の ptr から n 個下の位置から上にある要素を一度に取り出す．
SST_flush_all()	SST の全ての要素を一度に取り出す．
SST_move(ptr,n)	SST の ptr から n 個下の位置から上にある要素を自身の SST に移動する．
SST_copy(ptr,n,reg)	SST の ptr から n 個下の位置から上にある要素のコピーを reg に代入する．
SST_restore(reg)	SST のコピー reg を現在の SST に復元する．

3.2 SST を用いた並列仮想マシンの実現

共有メモリを対象としたSSTマシンの構成を**図1**に与える．PQ（Process Queue）は，アクセス競合を

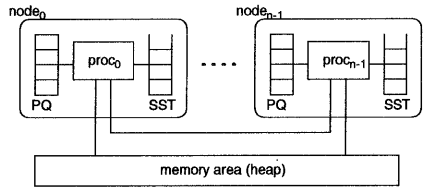

図1: SST マシンの構成

減少させるために，各プロセッサに割り当てた．

(1) 各プロセッサ上で動作するサブマシンは，Abelson-Sussman のレジスタマシンに基づく設計であり，レジスタと評価スタックを用いて式を評価する．環境は評価スタックを用いた shallow-binding により効率よく実現し，Scheme プログラムを実行するための 461 種類の命令を備えている．

(2) SST に対する命令の他，並列処理系として不可欠なプロセス制御，排他処理，各並列構文を実現するためのオブジェクトに関する命令を導入した．プロセス

---

* *The SST Machine and an Implementation of Parallel Constructs based Steal-Help Evaluation*

†Shinya Miyakawa, Takayasu Ito

‡Department of Computer and Mathematical Sciences, Graduate School of Information Sciences, Tohoku University

間通信は共有変数を用いた送信／受信により実現し，プロセス間のデータ移動は参照（ポインタ）の受け渡しにより効率よく実現している．

4 SST マシンによる並列構文の実現

並列構文は，SST マシン命令からなるコードに変換するコンパイル規則を与えることにより実現される．SST マシン命令は C 言語で実装されており，変換後のコードは，C コンパイラによってコンパイルして実行できる．例えば，PaiLisp で導入された **stealable** 構文 [1] は，以下のように実現される．

図 2 は，コンパイル時環境とキーワードによるコンパイル関数を用いて表わした (**stealable** e) に対するコードである．(**stealable** e) を評価したホームプロセッサは，式 e，環境，状態，親プロセスの識別子，SST ポインタ，排他変数から構成される **stealable** オブジェクト（SO）を生成する（1 行目）．式 e の自由変数の値を保存した（2 ～ 4 行目）SO を SST に積み（5 行目），その SO を **stealable** 式が返したホルダとして扱う．ホームプロセッサとスティールプロセッサによる評価は次のように行われる．

```
C[(stealable e), ctenv, label] =
1: make_so(label_stealable, n, pid, obj);
2: R[x1, ctenv]; set_so_env(obj, 1, val);
3: ...
4: R[xn, ctenv]; set_so_env(obj, n, val);
5: SST_push(obj, val); set_so_SST(obj,val);
6: load(fun, label); jump;
label_stealable
7: C[e, [x1, ..., xn]@{}, return];
```

図 2: **stealable** 式の評価

- ホームプロセッサは，**stealable** ホルダを本来の値に置換するため，例えば，引数が返した **stealable** ホルダに対しては関数呼び出しの直前に，**図 3** のルーチンを実行する．まず，SO を SST から取り出し（1 行目），式 e を遂次評価する（2 行目）．SO が既に steal されていた場合，その状態が obtained ならば，ホルダをその評価値に置き換える（6-8 行目）．それ以外は，状態を waiting に変えてスティール評価が終了するまで待つ（10-14 行目）．

```
1:  so_SST(obj,val); SST_delete(val, 0, val);
2:  branch(val, label_stealable_check1);
3:  so_label(val, fun); jump;
label_stealable_check1
4:  sstm_mutex_lock(obj);
5:  so_state(obj, val); eq(val, OBTAINED, val);
6:  branch(val, label_stealable_check2);
7:  sstm_mutex_unlock(obj);
8:  so_label(val, val);
9:  load(fun, label_stealable_check0); jump;
label_stealable_check2
10: set_so_state(obj, WAITING);
11: sstm_mutex_unlock(obj);
12: push(obj);
13: push(label_stealable_resume);
14: sstm_process_suspend();
```

図 3: **stealable** における逐次評価

- 空状態のプロセッサは，PQ に実行待ちのプロセスが無い場合，各プロセッサの SST を調べる．SO を steal すると，式 e を環境の下で評価した後，SO の状態が waiting ならば，親プロセスを再開させ，initilized ならば obtained に変更して値を保存する．

pcall や **future** などの SST マシンによる実現も同様に行われている．

5 SST マシンの性能評価

SST マシンは 6 プロセッサの共有メモリ型並列計算機 DEC7000 上に実現されており，このシステムを用いて評価実験を行った．**表 2** にフィボナッチ数の計算を各種並列構文を用いて並列化したプログラムの実行結果を与える．並列処理やメモリ管理に関しては，PaiLisp/MT コンパイラ [3] はインタプリタの実行系を呼び出しているのに対して，SST マシンによる実現ではコンパイラのための直接的実行ルーチンを用いることもあり，全体的に高速である．

表 3: SST マシンの実験結果

	SST マシン	PaiLisp/MT
遂次	0.526	1.254
pcall	0.208	0.527
stealable	0.221	（未実装）
future (SHE)	0.370	0.633

実行時間 [sec]

6 おわりに

本稿では，SST マシンの設計と実装の概要を述べ，例として **stealable** の実現法を述べた．SST マシンによる実用的な PaiLisp コンパイラの実現のため，PaiLisp の全ての並列構文の SST マシンによる（最適化を取り入れた）処理系作成を進めている．

参考文献

[1] T. Ito, *Efficient evaluation strategies for structured concurrency constructs in parallel scheme system*, LNCS, Vol.1068, pp22-52, Springer, 1996.

[2] 川本真一，伊藤貴康，スティール評価法を備えた PaiLisp システムの実現とその評価，情報処理学会論文誌，Vol.39, No.3, pp.692-703, 1998.

[3] T. Ito, S. Kawamoto, M. Umehara, *A multi-threaded implementation of PaiLisp interpreter and compiler using the steal-help evaluation strategy*, IPSJ Advanced Information Technology in Japan Series, Advanced Lisp Processing Techonology, pp.21-46, Gordon & Breach, 2000. (Also, available at http://www.ito.ecei.tohoku.ac.jp/PaiLisp/)

[4] T. Ito, T. Yuasa, *Notes on Parallel Evaluation Strategies and their related Issues*, Parallel and Distributed Computing for Symbolic and Irregular Applications, pp82-97, World Scientific, 2000.