5W−03

スティール評価法のためのSSTマシンによる並列Schemeシステムの実現*

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 Scheme 処理系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．スティール評価法のための仮想並列マシン SST マシンが伊藤により提案されている．本稿では，SST マシンをマルチスレッドに基づく C ライブラリとして実現した SST マシンライブラリについて説明し，それを用いて作成された並列 Scheme 言語 PaiLisp のコンパイラについて報告する．

2 スティール評価法と SST マシン

ETC（Eager Task Creation）によるプロセス過剰生成問題を解決する効率のよい並列評価法として導入されたスティール評価法 [1] は，並列評価したい式を共有情報と共に SST（Stealable Stack）のトップから入れ（push），ホームプロセッサは SST のトップから式を取り出し（pop）て評価し，空きプロセッサはホームプロセッサの SST のボトムから式と共有情報を取り出し（steal）て式を評価する並列評価法である．SST マシンは，スティール評価法が SST に対する操作によって実現されることに着目し，また，スティール評価法を様々な並列構文やアーキテクチャが異なる並列計算機に対しても系統的に実現するために提案された仮想並列マシンである．**図 1** に SST マシンの構成図を示す．

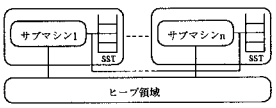

図 1: SST マシンの構成

3 SST マシンライブラリ

SST マシンを C 言語による処理系やターゲットマシンのアセンブリから利用できるように，マルチスレッドに基づく C ライブラリである SST マシンライブラリとして実現した．SST マシンを sstm_t 型，SST を sst_t 型，SST ポインタを ssp_t 型，SST のコピーを csst_t 型の C の構造体として SST マシン命令群を実装した．SST マシンの命令は，SST に対する操作が成功したか否かを返り値（1 または 0）によって判別できるように，int 型の整数を返す C 関数として実現した．SST マシンライブラリにおいて実現されている SST マシン基本操作命令の一部を**表 1** に示した．さらに，実行効率がよくコンパクトな SST マシンライブラリを実現するため，**表 2** に示すメモリ管理関数，**表 3** に示すスケジュールされたスティール命令実行関数，**表 4** に示す SST マシン実行管理関数などを導入している．SST マシンライブラリの利用により，SST マシンを並列処理系の実現に汎用的に利用できる．

表 1: SST に対する基本操作命令（一部）

int SST_push(sst_t sst, data_t data) sst のトップの要素の上にデータ data を入れ，新しく置いた data を指すようにトップポインタを上げる．
int SST_pop(sst_t sst, data_t* reg) sst に要素がある場合，トップ要素をレジスタ reg に入れ，sst からトップ要素を除いてトップポインタの位置を 1 つ下げる．SST が空の場合，NULL ポインタを reg に入れる．
int SST_steal(sst_t sst, data_t* reg) 空きサブマシンによって実行される命令であり，sst に要素がある場合，ボトム要素をレジスタに入れ，sst からボトムの要素を除いてボトムポインタが指す位置を 1 つ上げる．sst が空の場合，reg に NULL ポインタを入れる．
int SST_ref(sst_t sst, int m, data_t* reg) sst の要素が m 個以上の場合，sst のトップから m 番目の要素をレジスタ reg に入れ，sst の要素が m 個より少ない場合，reg に NULL ポインタを入れる．
int SST_del(sst_t sst, int m, data_t* reg) sst の要素が m 個以上の場合，sst のトップから m 番目の要素をレジスタ reg に入れてその要素を sst から除き，sst の要素が m 個より少ない場合，reg に NULL ポインタを入れる．
void SST_flush(sst_t sst, int m) sst の要素数が m 個以上の場合，トップから m 個の要素を一括削除し，sst の要素数が m 個より少ない場合，sst の全ての要素を除去する．

表 2: メモリ領域の解放に関する関数

void SST_pfree(ssp_t ssp) SST ポインタ ssp に割り当てたメモリ領域を解放する．
void SST_copy_free(csst_t copy) SST のコピー copy に割り当てたメモリ領域を解放する．

表 3: スケジュールされたスティール命令実行関数

int SSTM_steal(sstm_t sstm, sst_t sst, data_t* reg) SST マシン sstm の中の sst を有するサブマシン以外のサブマシンの SST から要素をスティールし，レジスタ reg に入れて 1 を返す．全ての SST が空の場合，いづれかの SST からスティールできるまで待つ．
int SSTM_checksteal(sstm_t sstm, sst_t sst, data_t* reg) SSTM_steal と同様の動作を行うが，全ての SST が空の場合，reg に NULL ポインタを入れて 0 を返す．

表 4: SST マシンの実行管理に関する関数（一部）

sstm_t SSTM_init() SST マシンの実行を開始するため，新しく確保した SST マシン構造体を返す．
sst_t SST_alloc(sstm_t sstm) 新しく確保した SST の構造体を SST マシン sstm に登録してから返す．
int SSTM_subcreate(void* start, void* a) start(a) を実行するサブマシンのスレッドを開始する．

*Design and Implementation of a Parallel Scheme System Using the SST Machine for the Steal-Help Evaluation Strategy

†Shinya Miyakawa, Takayasu Ito

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

4 並列 Scheme システムの実現

SST マシンライブラリとレジスタマシンを用いて並列 Scheme システムを実現した．実現する並列 Scheme 言語 PaiLisp[2] は，逐次 Scheme[3] にカーネル構文，**pcall**, **par**, **par-or**, **future**, **stealable** などの豊富な並列構文を拡張した並列言語である．PaiLisp を始めとする並列 Scheme では，リンク構造の環境と評価に当って必要な他の情報からなる共有情報を用いて式が評価されるので，共有メモリを用いた実現となる．サブマシンは，Abelson-Sussman のレジスタマシンを基にして，並列処理を実現するためのプロセスを扱えるように拡張し，評価スタックに対する操作命令を SST マシンの SST に対する操作命令を参考に設定し，C 言語によって実現した．**図 2** に，プログラムを入力して結果を出力するメインのサブマシンの構成図を示す．

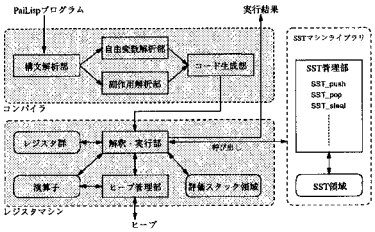

図 2: サブマシンの構成

PaiLisp プログラムは，コンパイラによりレジスタマシン命令と SST マシン命令からなる仮想マシンコードに変換される．仮想マシンコードは C コンパイラによりコンパイルされて実行される．例えば，並列関数適用式 **pcall** 式 [1] は，コンパイル環境とキーワードを用いたコンパイル関数によって**図 3** の仮想マシンコードに変換される．ホームプロセッサと空きプロセッサによる引数式の評価は次のように行われる．

- ホームプロセッサは SST から pop したラベルのルーチンを実行してから，カウンタを 1 つ減らす．
- 空きプロセッサはホームプロセッサの SST からラベルと共有情報のペアを steal し，共有情報の環境などの情報を用いてラベルのルーチンを実行してから，カウンタを 1 つ減らす．

カウンタが 0 になった時点でホームプロセッサは式 e_f を評価し，得られた関数を評価スタックにある引数式 $e_1, \ldots, e_n$ の評価値を用いて呼び出す．

他の並列構文も SST マシン命令とレジスタマシン命令を用いて実現されている．

```
C[(pcall e_f e_1 ... e_n), ctenv_0, return] =
1: push(reg(est), const("dummy"));
2: ...;
3: push(reg(est), const("dummy"));
{(1-3):引数式の数のダミー値を評価スタックに push する．}
4: push(reg(est), make_counter(n));
{ 初期値が引数の数のカウンタを評価スタックに push する．}
5: push(reg(est), reg(pid));
{ 親プロセスの名前 pid を評価スタックに push する．}
6: sp(reg(est), info);
{sp により info に入れた評価スタックポインタを共有情報とする．}
7: SST_push(reg(sst), cons(label(pcall_arg_n), reg(info)));
8: ...;
9: SST_push(reg(sst), cons(label(pcall_arg_1), reg(info)));
{(7-9):e_n,...,e_1 のラベルと共有情報のペアを SST に push する．}
10: load(cont, label(pcall_fun));
{ 関数呼び出しの pcall_fun ラベルを cont レジスタに入れる．}
11: jump(label(pcall_inline));
{SST の式を評価する pcall_inline ルーチンに jump する．}
pcall_arg_m
12: C[e_m, ctenv_1, next]; { 式 e_m を評価する．}
13: savep(info, n − m + 2 , val);
{ 式 e_m の評価値を評価スタックの指定した位置に保存する．}
14: pop(reg(est), cont);
15: jump(reg(cont));
{(14-15):評価スタックから pop したラベルに jump する．}
```

図 3: **pcall** 式の評価

5 並列 Scheme システムの性能評価

並列 Scheme システムは 8 プロセッサの共有メモリ型並列計算機 PrimePower600 上に実装されており，このシステムを用いた評価実験を行った．フィボナッチ数列の n 番目を求める fib(n)，fib(n) を m 回実行する fatwalk(n, m)，n 都市の巡回セールスマン問題を解く travsales(n)，一回探索の n クィーン問題を解く orqueen(n)，製品の生産と消費を n 回繰り返す生産者・消費者問題を解く pc(n) の逐次プログラムとその並列化版の実行結果を**表 5** に与える．「並列構文」は逐次プログラムの並列化に用いた構文である．「8 並列」実行時間は「逐次」実行時間と比較して，多数の細粒度プロセスを生成する fib でも 2.3 倍程度，その他の場合には 5 倍以上高速になっている．

表 5: ベンチマークプログラムの実行結果

プログラム名	逐次	並列構文	8 並列
fib(27)	0.544	**pcall**	0.230
fatwalk(20, 50)	1.114	**pbegin**	0.199
travsales(15)	1.822	**par**	0.281
orqueen(18)	2.423	**par-or**	0.480
pc(500)	0.360	kernel	0.015

[sec]

6 おわりに

本稿では，SST マシンライブラリとそれを用いた並列 Scheme システムの設計と実装の概要について述べた．並列 Scheme システムの評価実験を行い，SST マシンライブラリとレジスタマシンを用いて効率のよい並列 Scheme コンパイラが実現できることを報告した．

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] Kelsey, R., Clinger, W., Rees, J. et al., *Revised^5 Report on the Algorithmic Language Scheme*, ACM SIGPLAN Notices, 33(9), pp. 26–76, 1998.

[註] PrimePower600 は富士通株式会社の登録商標である．