Programming in PACLIB*

Hoon Hong
hhong@risc.uni-linz.ac.at

Wolfgang Schreiner
schreine@risc.uni-linz.ac.at

Research Institute for Symbolic Computation (RISC-Linz)
Johannes Kepler University, Linz, Austria

February 3, 1993

Abstract

This paper gives a short overview on PACLIB, a new system for parallel algebraic computation on shared memory computers. PACLIB has been developed as a professional tool for the simple design and efficient implementation of parallel algorithms in computer algebra and related areas. It provides concurrency, shared memory communication, non-determinism, speculative parallelism, streams and pipelining and a parallelized garbage collection. PACLIB has been implemented on a Sequent Symmetry multiprocessor and is portable to other shared memory machines and workstations.

1 Introduction

This paper gives a short overview on programming in PACLIB, a new library for parallel algebraic computation on shared memory computers. PACLIB has been developed as a professional tool for the simple design and efficient implementation of parallel algorithms in computer algebra and related areas.

PACLIB is a combination and extension of two public domain software systems:

1. **Saclib** [3] is a large library of algebraic algorithms written in the C language. These algorithms are based on a list-processing kernel that provides automatic garbage collection. Furthermore, the package supports exact arithmetic with arbitrary precision integers and rationals.

2. The **uSystem** [1] is a package that provides light-weight concurrency on mono-processor workstations and shared memory multiprocessors running under the UNIX operating system. The μSYSTEM provides an efficient management of *tasks* i.e. light-weight parallel processes.

*Sponsored by the Austrian Science Foundation (FWF) in the frame of the project S5302-PHY "Parallel Symbolic Computation".

PACLIB combines the list-processing and algebraic facilities of SACLIB with the concurrency possibilities of the μSYSTEM. In particular, PACLIB supports the following features:

- **Concurrent Tasks:** Any program function can be executed as a separate task; the scheduling of tasks and the balancing of load among processors is performed automatically.
- **Shared Memory Communication:** All data can be *shared* among tasks, i.e. task arguments and task results are passed by reference. No copying of data structures is required.
- **Non-Determinism:** A task may simultaneously wait for the results of various other tasks. The further computation depends on which task delivers its result first.
- **Task Abortion:** If by the result of one task the result of another task is not required any more, the superfluous subtask may be explicitly aborted.
- **Streams and Task Pipelines:** Tasks may be connected by *streams* of data and work in an overlapping fashion (pipelining).
- **Parallel Garbage Collection:** The garbage collector is based on a *mark and scan* scheme where the marking of of used cells and the reclamation of free cells are both parallelized.

PACLIB is a system still under evolution but we believe that the main features have already stabilized. The system has been implemented on a Sequent Symmetry shared memory computer with 20 processors running under the Dynix/ptx version of UNIX System V.3. However, it is portable to any other machine supported by the μSYSTEM package [1].

This short paper just gives an overview on the flavour of programming in PACLIB. A detailed description of the user interface can be found in [5]; the internal structure and the implementation of the system are described in [6]. We assume that the reader is familiar with the concepts of SACLIB; a description can be found in [3].

2 A Sample Program

A sample program shall demonstrate the main characteristics of a PACLIB program. In Figure 1 a parallel version of the school algorithm for the multiplication of two long integers a and b is presented.

This program splits b into subintegers of N digits each and to multiply a with each subinteger in parallel. `PIPROD` applies the PACLIB function `pacStart` to create n concurrent tasks executing the function `IPRODS` that multiplies a with b_0 and shifts the

```
#include <paclib.h>                  /* paclib definitions */
#define N 20                         /* digits per task */

Word PIPROD(Word a, Word b) {        /* multiply a with b in parallel */

  Word l, n, i, b0, b1;
  pacTask t;
  pacTaskList tlist;

  l = LENGTH(b);                     /* assume b is less than a */
  n = l / N + (l % N != 0);          /* number of multiplying tasks */
  tlist = NIL;                       /* list of multiplying tasks */

  for (i = 0; i < n; i++) {          /* start multiplying task */
    ADVANCE(N, b, &b0, &b1);         /* get b[0..N-1] and b[N..]  */
    t = pacStart(IPRODS, 3, a, b0, i*N); /* start task a*b0*BETA**(i*N) */
    tlist = COMP(t, tlist);          /* remember new task in tlist */
    b = b1;                          /* rest of b */
  }

  for (i = 0; i < n-1; i++) {        /* start adding tasks */
    b0 = pacWaitListRm(&tlist);      /* get result and remove task */
    b1 = pacWaitListRm(&tlist);      /* get result and remove task */
    t = pacStart(IADD, 2, b0, b1);   /* start adding task */
    tlist = COMP(t, tlist);          /* remember new task in tlist */
  }

  return(pacWaitListRm(&tlist));     /* return final result */

}

int pacMain(int argc, int *argv[]) { /* paclib main function */

  Word a, b;

  SWRITE("a = "); IREAD(a);
  SWRITE("b = "); IREAD(b);
  SWRITE("a*b = "); IWRITE(PIPROD(a, b));

}
```

Figure 1: Parallel Integer Multiplication

```
-------------------------------------------------------------------
PACLIB Parallel Algebraic Computation V1.01 (c) 1992 RISC-Linz
8 processors, 500000 cells, 50 free lists, 32000 bytes stack size.
-------------------------------------------------------------------
a = ...
b = ...
a*b = ...
-------------------------------------------------------------------
0 garbage collections, 0 cells and 0 arrays reclaimed.
Initialization time (TI):        4858 ms
Garbage collection time (TG):       0 ms
Actual execution time (TE):      2025 ms
Total program time (TI+TG+TE):   6883 ms
-------------------------------------------------------------------
```

Figure 2: PACLIB Program Output

result by $i * N$ `BETA` digits[1]. The descriptor t of each task is stored in *tlist* for further reference.

After all tasks have been started, `PIPROD` iteratively waits for the results delivered by any two tasks in *tlist* and starts a new task that sums up the result integers by the function `IADD`. Therefore, not only the multiplications but also the various additon operations are performed in parallel. Finally, there remains a single task left that returns the total result to the caller.

The main function `pacMain` reads two integers a and b, calls the function `PIPROD` and prints the result of the multiplication. The program includes the file `paclib.h` that contains the prototypes of all PACLIB functions The program file `piprod.c` is compiled by the statement

```
pacc -o piprod piprod.c
```

that creates the executable file `piprod` that can be started by

```
piprod +p8 +t1
```

The command line argument `+p8` says that 8 processors shall be applied for the program; `+t1` triggers the generation of timing information. The output then looks similar to that presented in Figure 2.

[1]The functions `IPRODS` and `ADVANCE` are actually not part of SACLIB.

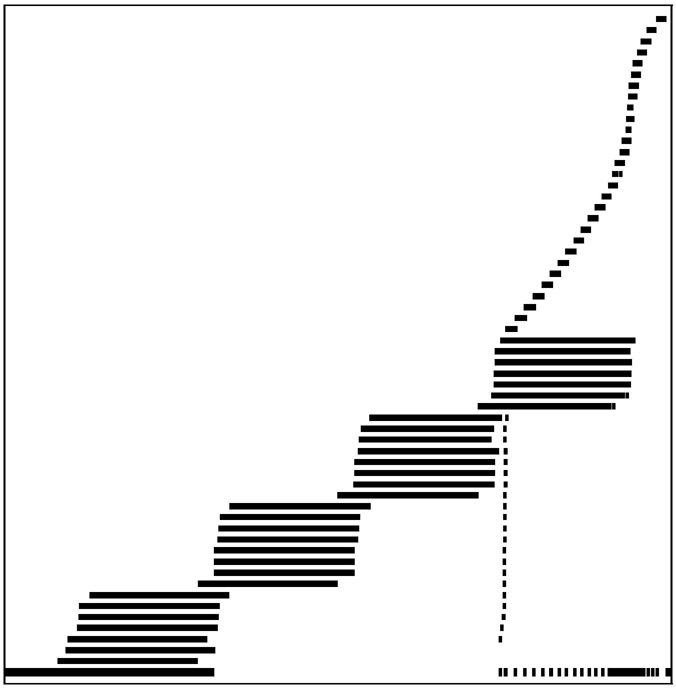

Figure 3: Time: 2025

The PACLIB kernel statistics is written into a file `pac.dat`. The command

```
pacgraph 250 250 <pac.dat >paclib.tex
```

visualizes these data by generating a LaTeX picture of size 250*250 points as displayed in Figure 3: each horizontal line represents one task and shows the time the task was actually active.

3 Related Work

Finally, we'd like to mention two systems for parallel algebraic computation developed by other groups and relate their work to ours:

- **PARSAC** [4] is a parallel implementation of SACLIB based on the operating system MACH. Similar to PACLIB, it supports light-weight concurrency (based on *threads*) and communication by shared memory. Its main difference to PACLIB is its memory management scheme that does not apply an explicit garbage collector: each thread has a separate free list space; when a thread terminates, it copies its result into the parent space and deallocates its own space for further use. PARSAC does not support non-determinism, explicit task abortion or streams.

- **MAPLE/LINDA** [2] is a parallel implementation of MAPLE based on the programming model LINDA. Each process runs a new copy of MAPLE that has its own address space and applies a separate garbage collection. Communication between processes is performed via the LINDA tuple space into which the result of a MAPLE process is copied. This is also the main difference to PACLIB where all tasks share a common address space and just object references are transmitted.

4 Acknowledgements

PACLIB was developed by the parallel computation group at RISC-Linz on an initiative of its leader Hoon Hong. The detailed design and the implementation of PACLIB were both performed by Wolfgang Schreiner. Andreas Neubacher provided as the maintainer of SACLIB valuable support. This work was sponsored by the Austrian Science Foundation (FWF) in the frame of the research project S5302-PHY "Parallel Symbolic Computation".

References

[1] Peter A. Buhr and Richard A. Stroobosscher. The μSystem: Providing Light-weight Concurrency on Shared-Memory Multiprocessor Computers Running UNIX. *Software — Practice and Experience*, 20(9):929–964, September 1990.

[2] Bruce W. Char. Progress report on a system for general-purpose parallel symbolic algebraic computation. In *Proceedings of the ISSAC'90, Tokyo, Japan, August 20–24*, pages 96–103, Department of Computer Science, University of Tennessee, Knoxville, TN 37996-1301, 1990. ACM Press, New York.

[3] George Collins, Hoon Hong, Mark Encarnation, Jeremy Johnson, Werner Krandick, and Andreas Neubacher. A SACLIB Primer. Technical Report 92-34, RISC-Linz, Johannes Kepler University, Linz, Austria, 1992. To appear.

[4] Wolfgang Kuechlin. A Parallel SAC-2 Based on Threads. Technical report, Computer and Information Sciences, Ohio State University, Columbus, Ohio, April 1990.

[5] Wolfgang Schreiner. PACLIB User Manual. Technical Report 92-32, RISC-Linz, Johannes Kepler University, Linz, Austria, May 1992.

[6] Wolfgang Schreiner. PACLIB — Structure and Implementation. Technical Report 92-33, RISC-Linz, Johannes Kepler University, Linz, Austria, 1992. To appear.