The CL-PVM Package

Liwei Li and Paul S. Wang[*]
Institute for Computational Mathematics
Department of Mathematics and Computer Science
Kent State University

January 24, 1996

Abstract

The CL-PVM package consists of a set of Common Lisp functions that interfaces Common Lisp (KCL, AKCL, or GCL) to the C-based library of PVM. CL-PVM also offers a set of tools to help use it effectively with Lisp and MAXIMA tasks. Documentation, on-line manual pages, and examples are also included. The software is available through public FTP at ftp.mcs.kent.edu in the directory /pub/wang/.

1 Introduction

Parallel Virtual Machine (PVM) is a software package that integrates a heterogeneous network of computers to form a single parallel/concurrent computing facility [2]. PVM consists of two parts: a run-time server and a set of library functions. A user sets up a *hostfile* that lists the names of the hosts constituting the *parallel virtual machine* (*pvm*). A PVM server runs on each host to help manage the *pvm*. Hosts can be added and deleted from the *pvm* dynamically. The *pvm* can be controlled from any constituent host either interactively from a *console* program or automatically from any *PVM tasks*.

A PVM task is an application program that runs on a *pvm* and can use the PVM library functions to interact with other tasks: sending and receiving messages, initiating subtasks, detecting errors, etc. The PVM

[*]Work reported herein has been supported in part by the National Science Foundation under Grant CCR-9503650 and CCR-9201800 Software Capitalization Supplement

version 3.0 library is written in C allowing direct calls from C programs. There is also a Fortran 77 interface to give F77 programs access to the PVM library.

CL-PVM provides a Common Lisp interface enabling Lisp-based programs to partake in PVM applications. A wide variety of useful Lisp programs exists including symbolic computation systems, expert systems, artificial intelligence systems, knowledge-based systems, and many more. With CL-PVM, the PVM library routines can be invoked interactively from the Lisp toplevel or from Lisp programs.

The CL-PVM package contains a set of Common Lisp functions that interfaces Common Lisp (KCL, AKCL, or GCL) to the C-based library of PVM [2]. Generally, there is one CL interface function to each PVM C library function. The CL function calls its C-based counterpart and relays data to and from the C function. This interface is complete and allows Lisp-based programs to run on a *pvm* and thus facilitates the combination of symbolic, numeric, graphics, and other useful systems in a distributed fashion (Fig. 1). CL-PVM also offers a set of tools to aid effective use of the package with Lisp and MAXIMA tasks. Documentation, on-line manual pages, and examples are also included.

CL-PVM is available by public FTP (`ftp.mcs.kent.edu`) in the directory (`/pub/wang/`). The package is described here. Please refer to [4] for more information on the design and implementation of the Lisp interface to PVM.

2 File Organization

The main directory of CL-PVM is `clpvm` and it contains six directories:

- `src/`: Lisp source code files for the interface functions to PVM.
- `lib/`: Compiled files of the `.lsp` source files.
- `report/`: This article and other useful documents.
- `tools/`: Useful commands.
- `examples/`: Complete examples to demonstrate and test the CL-PVM package.
- `man/`: On-line manual pages.

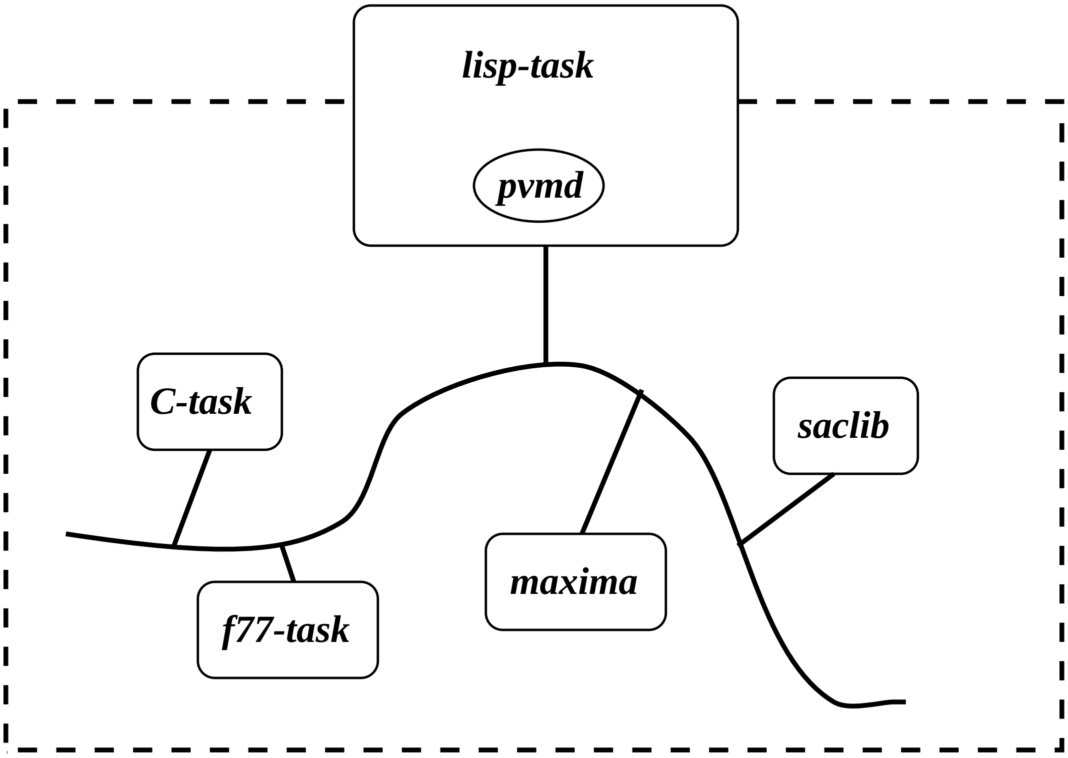

Figure 1: Tool Integration via PVM

After installing CL-PVM, first compile the `.o` files for the desired architectures under `lib/`. Then perform tests by using the interface interactively and by running the examples provided.

3 Interactive Use

Being interactive, Lisp provides an easy way to test CL-PVM. Whenever a user invokes Lisp the file `init.lsp`, if present in the current directory, is loaded into Lisp. Thus, CL-PVM can be loaded into Lisp automatically this way. For example,

```
;;;;;;;;;;;;;;;; init.lsp  ;;;;;;;;;;;;;;;;;;;;;
(si::faslink "/usr/local/clpvm/lib/SUN4/clpvm.o"
             "/usr/local/lib/libpvm3.a -lc")
```

loads both the `libpvm3.a` library routines and their Lisp interfaces into Lisp. The `clpvm.o` file is produced by compiling `clpvm.lsp` with the Common Lisp compiler (the **compile** function in CL). Once the CL-PVM interface is loaded, Lisp commands such as

```
(pvm-spawn "hello_other" "" 0 "condor.mcs.kent.edu" 1 'pids)
```

can be issued interactively. See [4], found in the `report/` directory, for an example of interactive testing and a sequence of Lisp commands to use.

4 Lisp Processes as PVM Tasks

Interactive use is fine, but PVM applications usually require non-interactive tasks. A Lisp process can be turned into a PVM task by writing a simple shell script. Consider the `hello.lsp` program with a top-level call `(hello ...)`. The following *csh* script can be used:

```
#!/bin/csh
set logfile = /tmp/${USER}LOG.$$
set gcl_w_pvm = /usr/local/gcl/gcl+pvm
set gcl_dir = /usr/local/gcl/
echo $0 $argv INVOKED >! $logfile

set cmd = ( hello )
foreach a ( $argv )
    set cmd = ( $cmd \"$a\" )
end

$gcl_w_pvm $gcl_dir >>&! $logfile <<EOF
(if (null (load "$HOME/pvm3/local_disk_bin/hello.o"
        :if-does-not-exist nil))
        (progn (princ "loading hello.o failed.") (bye) ))
( $cmd )
( bye )
EOF
echo FINISHED >>! $logfile
```

The executable `$gcl_w_pvm` is a version of GCL with CL-PVM codes already loaded.

The files `hello.lsp` and `hello_other.lsp` in the `examples/lisp/` subdirectory of this package show Lisp codes that uses CL-PVM. The `hello.o` file in the above script refers to the Lisp compiled code produced from `hello.lsp`.

This particular script takes one command-line argument indicating the host where to spawn the `hello_other` task (see `hello.lsp`). The command-line argument processing shown works in general for any number of arguments. Using a shell script to invoke a Lisp process has another advantage: the program is architecture independent.

Executable shell scripts like the one shown have a common structure and can be automatically generated given appropriate data. The **pvm_cltask** tool included in the package does just that.

5 Additional Tools

At ICM/Kent, we follow an organizational scheme where a centrally installed PVM root directory is shared by all users. Each user locates his/her PVM applications in a standard directory (`$HOME/pvm3/` by default) and the `local_disk_bin` subdirectory therein is assumed to be stored in a file system local to a host. Anyone adopting the standard setup can also take advantage of a set of tools that make PVM much easier to use. See [6] for details on the organization and the available tools.

A subset of the tools relates to CL-PVM and is included in this package. The tools will work without modification when you adopt the standard organization [6]. Minor changes can be made to customize these tools for other PVM organizations.

Among these tools are

- **pvmlc** — a command to compile and distribute CL-coded files for PVM tasks on all specified hosts ussed in the form:

  pvmlc `[ -U` *compiler* `] [ -H` *hostfile* `]` *file*`.lsp ...`

- **pvm_cltask** — a command used to generate a PVM executable task in the form of a *csh* script when given a list of Lisp object files. It is used in the form

  pvm_cltask `[ -L` *lisp* `] [ -N` *executable* `]` *file*`.o ...`

The script automatically invokes a user-specified Lisp function and passes to it any command line arguments. The generated script can be edited and later distributed by **pvm_distrib**.

- **pvm_distrib** — a command to distribute PVM executables to specified hosts used in the form

  pvm_distrib `[ -T` *architecture* `] [ -H` *hostfile* `]` *a.out1* ...

These tools and their documentations are included with the CL-PVM package.

6 Using PVM from MAXIMA

With CL-PVM, it is rather simple to run MAXIMA as a PVM task since the symbolic computation system is built on top of AKCL. You can load the CL-PVM interface into MAXIMA and produce the **maxima+pvm** command to run MAXIMA-based PVM tasks. This can be done, for example, with

```
(si::faslink "/usr/local/lib/clpvm/lib/clpvm3.o"
                         "/usr/local/lib/libpvm3.a -lc")
(si:save-system "maxima+pvm")
```

The CL-PVM package also includes

- **pvmmc** — a command to compile `.lsp` files written for MAXIMA to run on a *pvm*

- **pvm_maximatask** — a command to generate maxima-based PVM tasks.

Examples can be found in the `examples/maxima/` subdirectory.

7 On-line Manual Pages

Manual pages accessible by the UNIX **man** command are included in the `man/` directory. These are written by Liwei Li who also wrote most of the Lisp interface. There is one man page for each Lisp interface routine contained in the `man/man3/` subdirectory. The `man1` subdirectory contains documentation for the shell-level tools.

8 Related Work at Kent

The Symbolic Computation Group at ICM/Kent is working on many interesting projects. Three related items are worth mentioning here:

- A set of tools to help compile and manage PVM tasks in C, F77, and Common Lisp in general is available [6].
- A set of functions that interfaces the symbolic package SACLIB [1] to PVM is available.
- For efficient transfer of mathematical data among distributed processes in a heterogeneous environment, such as that represented by a *pvm*, the MP protocol [3] has been developed. MP 1.0 is being readied for release in Summer of 1996.

References

[1] B. Buchberger *et al. SACLIB 1.1 User's Guide.* RISC-Report No. 93-19, Linz, Austria, 1993.

[2] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek and V. Sunderam, *PVM 3 User's Guide and Reference Manual*, Oak Ridge National Laboratory, 1993.

[3] S. Gray, N. Kajler, and P. Wang, "MP: A Protocol for Efficient Exchange of Mathematical Expressions," Proceedings, ISSAC'94, Oxford UK, ACM Press, pp. 330-335, July, 1994.

[4] L. Li and P. S. Wang, "CL-PVM: A Common Lisp Interface to PVM," Proceedings[1], PVM User's Meeting, Pittsburgh, May 7-10, 1995. (Also ICM Technical Report ICM-199509-01, `http://www.mcs.kent.edu/cgi-bin/publish/access`).

[5] P. Wang, *An Introduction to ANSI C on UNIX*, PWS Publishing Co. Boston, MA., 1991.

[6] P. S. Wang, "PVM Guide," ICM/Kent Technical Report (ICM-199509-02), Institute for Computational Mathematics, Kent State University, `http://www.mcs.kent.edu/cgi-bin/publish/access` (1995).

[1] `http://www.cs.cmu.edu/Web/Groups/pvmug95.html`

[7] R. Wilensky, *Common LISPcraft*, W. W. Norton & Co., N.Y., 1986.

[8] T. Yuasa and M. Hagiya, “Kyoto Common Lisp Dictionary”, Kyoto University, KCL on-line documentation” 1986.