Laboratory of Computer Science Research Technical Report LCSR-TR-257, January 1996

The FM and PL Libraries Documentation

Javier Elices

Department of Computer Science
Rutgers University
Hill Center, Busch Campus
Piscataway, NJ 08855
email: elices@cs.rutgers.edu

Abstract

Building complex SPMD code in an efficient and portable way is nowadays a challenge, especially when there is no uniformity of tools and libraries across platforms. The Fast Messages (FM) and the Portability Library (PL) where both designed to provide the basis of an abstract enough framework for C, so that problems can be coded and ported to any supported platform with no more than a few changes in the makefiles and a recompilation. The FM library provides a message passing communications library built around the Berkeley Active Messages library. The PL library provides the primitives for host to node communication for problem initialization and results collection, as well as other miscellaneous and potentially non-portable primitives. This technical report contains the documentation for both libraries.

1 Introduction

It is unrealistic to think that a complex framework coded in an SPMD paradigm can survive long enough to be useful, unless it is written in a portable way that allows it to migrate to different platforms. Massively parallel machines have a very short life span compared to other platforms and become obsolete —and no longer usable— in a few years. Even so, new ideas in algorithm design have to be coded and tested, because it is only the empirical evidence what makes them acceptable with a good degree of confidence. That is the motivation behind the development of the Fast Messages (FM) and the Portability Library (PL) libraries. Both of them put together provide the basis of an abstract enough framework in which to code message passing SPMD code in C.

Fast Messages (FM) is a communication library based on services provided by the Berkeley's Active Messages (AM) library or any other similar library. It provides higher level primitives than those in the AM library, all of them intended for communication between processes (processors on parallel machines.) It provides facilities for sending and receiving unicast messages as well as a simple interface for dealing with errors that may occur during the user computation. As of this writing, the FM library has been implemented and tested under the following AM-like primitive versions, machines and operating systems:

- A TCP/IP version of Berkeley's AM, that works over TCP/IP on Sun machines running Solaris. This version is not intended for performance —which it would not achieve— but rather, for testing purposes. A version of the program under development can be compiled and run with this version until the program is ready for a "production" version based on a real parallel machine.
- A CM-5 version of CMAML, that works on CM-5 machines running CMost.

The Portability Library (PL), which really started standing for Parallel/Portable Library, is a library of primitives for those services that, although provided in most or all systems, are achieved in different ways or through different system-specific function calls. By grouping all those functions together and assigning

them a single name across systems, portability can be preserved as soon as a version of the PL library is available. As of this writing, the PL library has been implemented and tested under the following machines and operating systems:

- An emulated version, that works on Sun machines running Solaris. This version is not intended for performance —which it would not achieve— but rather, for testing purposes. A version of the program under development can be compiled and run with this version until the program is ready for a "production" version based on a real parallel machine.
- A CM-5 version, that works on CM-5 machines running CMost.

The complete documentation for both libraries follows. The rest of this document is also online at the URL:

http://www.prolangs.rutgers.edu/software

where it will be kept up-to-date if there are changes in the future.