Advanced Capabilities in PVM 3.4

G. A. Geist *
Oak Ridge National Laboratory, USA

Abstract. This paper describes many of the new features available in the newly released version 3.4 of PVM (Parallel Virtual Machine). Under development for over a year, PVM 3.4 provides a big leap forward in capabilities over PVM 3.3 by adding communication context, message handlers, message box, user defined tracing, and a WIN32 port to the virtual machine model. This paper will describe how to take advantage of these new features to create sophisticated distributed computing applications. The reader is expected to already have an understanding of parallel and distributed computing and a knowledge of the basic features provided by the PVM software. Other programming topics covered in this paper include improving application performance, adding fault tolerance, building interactive applications, and debugging.

1 New features in PVM 3.4

1.1 Communication context

One of the main difficulties of writing libraries for message passing applications is that messages sent inside the application may get intercepted by the message passing calls inside the library. The same problem occurs when two applications want to cooperate, for example, a performance monitor and a scientific application or an airframe stress application coupled with an aerodynamic flow application. Any time there are two or more programmers writing different parts of the overall message passing application there is the potential that a message will be inadvertently received by the wrong part of the application. The solution to this problem is communication context.

In PVM 3.3 pvm_recv() requests a message from a particular source with a user chosen message tag (Either or both of these fields can be set to accept anything). PVM 3.4 adds communication context as a third field that a recv must match on before accepting a message and context can not be wild-carded. By default there is a base context that is a predefined constant. This default context allows existing PVM 3.3 applications to run unchanged and for different virtual machines to cooperate.

PVM 3.4 adds four new routines to manage communication contexts.

* This work was supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under Contract DE-AC05-96OR22464 with Lockheed Martin Energy Research Corporation
new_context = pvm_newcontext()
old_context = pvm_setcontext( new_context )
info = pvm_freecontext( context )
context = pvm_getcontext()

Pvm_newcontext() returns a system-wide unique context tag generated by the local daemon (in a way similar to the way the local daemon generates system-wide unique task IDs). Since it is a local operation, pvm_newcontext is very fast. The returned context can then be broadcast to all the tasks that are cooperating on this part of the application. Each of the tasks calls pvm_setcontext, which switches the active context and returns the old context tag so that it can be restored at the end of the module by another call to pvm_setcontext. Pvm_freecontext and pvm_getcontext are used to free memory associated with a context tag and to get the value of the active context tag respectively.

Spawned tasks inherit the context of their parent. Thus if you wanted to add context an existing parallel routine already written in PVM, it requires adding only four lines to the source.

```c
int mycxt, oldcxt;
/* near the beginning of the routine set a new context */
mycxt = pvm_newcontext();
oldcxt = pvm_setcontext( mycxt );

/* spawn slave tasks to help */
/* slave tasks require no source code change */
/* leave all the PVM calls in master unchanged */

/* just before exiting the routine restore previous context */
mycxt = pvm_setcontext( oldcxt );
pvm_freecontext( mycxt );

return;
```

1.2 Message handlers

PVM has always had message handlers internally, which were used for controlling the virtual machine. In PVM 3.4 the ability to define and delete message handlers has been raised up to the user level. The two new message handler functions are:

```c
mhid = pvm_addmhf( src, tag, context, *function );
pvm_delmhf( mhid );
```

Once a message handler has been added by a task, whenever a message arrives at this task with the specified source, message tag, and communication context, the specified function is executed. The function is passed the message ID so that it may unpack the message if desired. PVM 3.4 places no restrictions on the complexity of the function, which is free to make system calls or other PVM