Universidad de Talca
Facultad de Ingeniería

Computación Paralela y Distribuida

PVM: Parallel Virtual Machine

Este documento describe, de manera general, cómo usar en la práctica `PVM`. El código que se muestra ha sido probado en Linux, sin embargo, debería funcionar sin problemas en otro tipo de ambientes para los cuales `PVM` esté disponible, e incluso en sistemas heterogéneos compuestos por máquinas con distintas arquitecturas y sistemas operativos.

PVM

`PVM` permite utilizar una colección de computadores heterogéneos como una única máquina virtual paralela. El sistema administra en forma transparente el enrutamiento de mensajes, la conversión de datos y la planificación de tareas en la máquina virtual.
Los programas están compuestos por una serie de tareas que cooperan entre sí. Estas tareas acceden a recursos de `PVM` a través de una biblioteca estándar, que contiene funciones para iniciación y finalización de tareas así como para comunicación y sincronización entre tareas.
El sistema `PVM` se compone de dos partes. La primera parte es un *daemon*, llamado `pvmd3`, el cual reside en todas las máquinas que forman la máquina virtual. Cuando un usuario desea ejecutar una aplicación, lo primero que debe hacer es crear la máquina virtual, iniciando `PVM`. Luego, la aplicación se puede ejecutar desde cualquiera de los computadores en el sistema. Múltiples usuarios pueden configurar máquinas virtuales y cada uno de ellos puede ejecutar múltiples aplicaciones simultáneamente en su configuración.
La segunda parte de `PVM` es una biblioteca de funciones que ofrece un completo repertorio de primitivas. Existen funciones para enviar y recibir mensajes, iniciar procesos, coordinar tareas y modificar la máquina virtual. Adjunto se encontrará un listado con las funciones que forman la interfaz C de la biblioteca `PVM`.
El modelo de computación de `PVM` se basa en la noción de que una aplicación consiste de múltiples tareas. Cada tarea es responsable de una parte de la carga de trabajo computacional de la aplicación. Las tareas se identifican mediante un entero llamado el identificador de tarea y abreviado `TID`. Los mensajes se envían y reciben utilizando estos *tids*, los cuales son asignados por `PVM`.

Programa de ejemplo

El ejemplo que se muestra a continuación es una aplicación `PVM` que se ha separado en dos tipos de programa. Un programa *maestro* es ejecutado por el usuario en uno de los computadores. Este programa provoca la ejecución de programas *esclavos* en otros computadores de la máquina virtual y son éstos los que llevan a cabo el trabajo. Una vez que han finalizado reportan sus resultados al *maestro*. Para efectos del ejemplo, los programas *esclavos* simplemente envían un mensaje con información local al *maestro*, el cual la desplegará por la pantalla.

El *maestro* hace lo siguiente:

- Determina cuáles computadores forman parte de la máquina virtual
- Inicia un proceso *esclavo* en cada computador
- Recolecta los resultados enviados por los *esclavos*
- Despliega en pantalla la información recibida

El *esclavo* hace lo siguiente:

- Determina el nombre del computador en que se está ejecutando
- Determina la hora en ese computador
- Envía un mensaje al *maestro* con la información anterior

Programa *maestro* (`master.c`):

```
#include <stdio.h>
#include <pvm3.h>

main() {
   struct pvmhostinfo *hostp;
   int result, check, i, nhosts, narch, stid;
   char buf[64];

   pvm_setopt(PvmRoute, PvmRouteDirect);  /* canal para comunicacion */

   gethostname(buf, 20);                  /* obtiene nombre del maestro */
   printf("El programa maestro se ejecuta en %s\n", buf);

   /* Obtiene y despliega la configuracion de la maquina paralela */
   pvm_config(&nhosts, &narch, &hostp);    /* Obtiene configuracion */
   printf("Se encontraron los siguientes computadores en la maquina paralela:\n");
   for (i=0; i<nhosts; i++)
      printf("\t%s\n", hostp[i].hi_name);

   /* Ejecuta los procesos en todas las maquinas */
   for (i=0; i<nhosts; i++) {
      check = pvm_spawn("slave", 0, PvmTaskHost, hostp[i].hi_name, 1, &stid);
      if (!check) printf("No se pudo iniciar proceso en %s\n", hostp[i].hi_name);
   }

   /* Espera las respuestas de todas las maquinas e imprime su contenido */
   result=0;
   while (result<nhosts) {
      pvm_recv(-1, 2);        /* Espera por mensaje de respuesta */
      pvm_upkstr(buf);        /* Desempaca el mensaje */
      printf("%s\n", buf);    /* Despliega el contenido en pantalla */
      result++;               /* Siguiente mensaje */
   }

   pvm_exit;                  /* Maestro finalizando */
}
```

Programa *esclavo* (`slave.c`):

```
#include <stdio.h>
#include <pvm3.h>
#include <time.h>

main() {
   time_t now;
   char name[12], buf[60];
   int ptid;

   ptid = pvm_parent();            /* El ID del proceso maestro */
   pvm_setopt(PvmRoute, PvmRouteDirect);

   gethostname(name, 64);          /* Obtiene nombre de la maquina */
   now = time(NULL);               /* Obtiene hora de la maquina */
   strcpy(buf, name);              /* Coloca el nombre en el string */
   strcat(buf, "'s time is ");
   strcat(buf, ctime(&now));       /* Agrega la hora al string */

   pvm_initsend(PvmDataDefault);   /* Asigna buffer para el mensaje */
   pvm_pkstr(buf);                 /* Empaca el string en el buffer */
   pvm_send(ptid, 2);              /* Envia el buffer al maestro */

   pvm_exit;                       /* Esclavo finalizando */
}
```

Observaciones sobre el programa ejemplo

La función `pvm_setopt`, empleada en ambos programas, permite configurar opciones. En el ejemplo se establece el mecanismo de enrutamiento que, en la mayoría de los casos debe hacerse en la forma mostrada. La función `pvm_config` permite obtener información sobre la configuración de la máquina virtual. En este caso interesa conocer las máquinas participantes.

La ejecución remota de los *esclavos* se lleva a cabo empleando la función `pvm_spawn`. El primer parámetro indica el programa a ejecutar, el segundo son argumentos que se puede pasar al programa, el tercero es una bandera que en este caso indica que el cuarto parámetro contiene el nombre de la máquina donde se debe ejecutar el programa. El quinto parámetro indica la cantidad de tareas que debe iniciarse y el último se usa como parámetro de salida para conocer los identificadores globales de las tareas creadas.

La información enviada en los mensajes debe ser empaquetada y desempaquetada al recibirse, con el objetivo de manejar la posible heterogeneidad entre las máquinas participantes. Las funciones `pvm_pkstr` y `pvm_upkstr` empacan y desempacan strings, respectivamente. Las funciones para manipular otros tipos de datos se pueden consultar en la guía de referencia adjunta.

Para poder enviar mensajes es necesario inicializar un *buffer* empleando la función `pvm_initsend`. El envío y la recepción de mensajes se logran empleando las funciones `pvm_send` y `pvm_recv`, respectivamente. Para el caso del envío, el primer parámetro de `pvm_send` indica el identificador del destinatario, mientras que el segundo es un *tag* que ayuda a clasificar los mensajes. Al recibir un mensaje, es posible pasarle dos parámetros a `pvm_recv`. El primero se refiere al identificador del proceso del cual se quiere recibir un mensaje, y el segundo corresponde al *tag* que se está esperando. Para que el mensaje se reciba debe provenir del emisor especificado y debe tener el *tag* especificado. Un valor de -1 en cualquiera de los dos parámetros indica indiferencia, es decir, se recibirá de cualquier emisor o con cualquier *tag*.

Finalmente, la función `pvm_exit` notifica a la máquina paralela que el proceso ha finalizado su ejecución.

Configuración del ambiente

PVM requiere de cierta configuración para cada usuario. Es necesario crear un archivo .rhosts en el directorio HOME de cada usuario que quiera ejecutar programas bajo PVM. Este archivo debe contener una línea por cada computador que se desee forme parte de la máquina virtual, así como el nombre del usuario que se utilizará en cada una de ellas. En este caso, debería ser algo como:

```
maquina1 username
maquina2 username
maquina3 username
maquina4 username
```

Recuerde cambiar username por su identificador de usuario.
Además, debe crear el directorio pvm3/bin/LINUX en el directorio HOME y colocar ahí sus programas.

Configuración de la máquina virtual

Para configurar la máquina virtual debe ejecutarse el programa pvm en cualquiera de los computadores. El programa mostrará un *prompt* en el cual es posible ingresar comandos de configuración y de consulta, por ejemplo conf:

```
pvm> conf
```

En la guía adjunta puede consultarse los comandos disponibles desde la consola PVM. Por el momento, para la ejecución de programas son de interés:

`add hostname` — Para agregar el computador con nombre `hostname` a la máquina virtual

`delete hostname` — Para sacar el computador `hostname` de la máquina virtual

`conf` — Para ver la configuración actual

`ps -a` — Para ver los procesos en ejecución en la máquina virtual

`quit` — Para salir de la consola pero dejar activa la máquina virtual

`halt` — Para detener la máquina virtual y salir de la consola

También es posible ejecutar pvm pasándole como parámetro el nombre de un archivo con los computadores que participarán en la máquina virtual. De este modo no es necesario agregar un computador a la vez.
Una vez que se ha configurado la máquina virtual será posible ejecutar las aplicaciones desde cualquiera de los computadores que forman parte del sistema.

Compilación de Programas

```
cc -o prog prog.c -lpvm3
```

Para el caso del ejemplo mostrado anteriormente se debe hacer:

```
cc -o master master.c -lpvm3
cc -o slave slave.c -lpvm3
```

Posteriormente se ejecuta master en cualquiera de los computadores.