Utilisation de PVM et de XPVM

Eric Goubault

21 janvier 1998

1 Configuration de PVM

Vous avez accès à une partition disque sans quota sous `/users/pvm/LOGIN`. Attention à ne pas laisser trop d'exécutables chez vous! Si vous choisissez d'utiliser cette partition, il faut remplacer les `$HOME` dans tous les fichiers qui suivent (`xpvm_hosts`, `Makefile.aimk`) par `/users/pvm/LOGIN`.

Vous trouverez dans `~goubaul1/pvm_install.tar` une installation prédéfinie pour commencer vos TDs. Récupérez là:

```
> cp ~goubaul1/pvm_install.tar .
> tar -xvf pvm_install.tar
...
> cd pvm
> ls
bin         include     lib         src         xpvm_hosts
```

Le répertoire `bin` contient des répertoires par architecture possible. En effet, on doit avoir un exécutable différent du même programme pour toutes les architectures d'une machine PVM.

Le répertoire `src` contient les sources des programmes PVM. C'est là que vous allez programmer.

Le fichier `xpvm_hosts` est un fichier de configuration de la machine PVM (par défaut). Faites,

```
> cp xpvm_hosts ../.xpvm_hosts
```

Ainsi le fichier de configuration sera automatiquement reconnu par `xpvm` (mais pas automatiquement par `pvm`).

Ce fichier est comme suit,

```
* ep=$HOME/pvm/bin/$PVM_ARCH
& radius
& femur
& phalange
```

`ep` indique de considérer ce qui suit comme étant l'endroit en machine où chercher les exécutables. Les noms sur les lignes suivantes sont des noms de stations que l'on veut pouvoir inclure dans la configuration de la machine pvm dès son démarrage (marche sous xpvm).

2 Compilation d'un programme C/PVM

Pour compiler un programme PVM, il faut,

- compiler le programme C correspondant pour chaque architecture des machines entrant dans la composition de la machine PVM,
- faire le lien avec les bibliothèques PVM,
- ranger le (ou les) exécutable(s) dans des répertoires accessibles aux démons `pvmd3` selon les architectures.

Tout cela est géré par des `makefile`.

Les `makefile` de pvm sont très proches de ceux utilisés ordinairement en `C`. Seule la commande pour les utiliser change: elle s'appelle ici `aimk`.

Exemple de `Makefile` (fourni),

```
SDIR = $(HOME)/pvm/src
#BDIR = $(HOME)/pvm3/bin
BDIR = $(SDIR)/../bin
XDIR = $(BDIR)/$(PVM_ARCH)

OPTIONS = -O
CFLAGS = $(OPTIONS) -I../include -I$(PVM_ROOT)/include $(ARCHCFLAGS)

LIBS = -lpvm3 $(ARCHLIB)
GLIBS = -lgpvm3

LFLAGS = $(LOPT) -L$(PVM_ROOT)/lib/$(PVM_ARCH)

default: hello hello_other

hello: $(SDIR)/hello.c $(XDIR)
$(CC) $(CFLAGS) -o $@ $(SDIR)/hello.c $(LFLAGS) $(LIBS)
mv $@ $(XDIR)

hello_other: $(SDIR)/hello_other.c $(XDIR)
$(CC) $(CFLAGS) -o $@ $(SDIR)/hello_other.c $(LFLAGS) $(LIBS)
mv $@ $(XDIR)
```

Le `makefile` plus haut se contente essentiellement de compiler différents programmes exemples (voir section 5) avec la librairie `pvm3` puis de bouger le code compilé dans le répertoire que pvm connaît comme renfermant les exécutables (cf `ep` de la section 1).

Exemple: sur une DEC (tarse):

```
goubaul1@tarse ~ > cd pvm/src
goubaul1@tarse ~/pvm/src > aimk
```

```
making in ALPHA/ for ALPHA
cc -O -I../include -I/usr/local/pvm/include -DHASSTDLIB -o hello /users/profs/go
ubault/pvm/src/hello.c  -L/usr/local/pvm/lib/ALPHA -lpvm3
mv hello /users/profs/goubault/pvm/src/../bin/ALPHA
cc -O -I../include -I/usr/local/pvm/include -DHASSTDLIB -o hello_other /users/pr
ofs/goubault/pvm/src/hello_other.c  -L/usr/local/pvm/lib/ALPHA -lpvm3
mv hello_other /users/profs/goubault/pvm/src/../bin/ALPHA
```

Puis sur une HP (royce):

```
goubaul1@royce ~ > cd pvm/src
goubaul1@royce ~/pvm/src > aimk
making in HPPA/ for HPPA
        cc -O -I../include -I/usr/local/pvm/include -DSYSVSIGNAL -DNOWAIT3 -DNOG
ETDTBLSIZ -DRSHCOMMAND=\"/usr/bin/remsh\" -DFDSETISINT -o hello /users/profs/gou
bault/pvm/src/hello.c -Wl,-aarchive -L/usr/local/pvm/lib/HPPA -lpvm3
        mv hello /users/profs/goubault/pvm/src/../bin/HPPA
        cc -O -I../include -I/usr/local/pvm/include -DSYSVSIGNAL -DNOWAIT3 -DNOG
ETDTBLSIZ -DRSHCOMMAND=\"/usr/bin/remsh\" -DFDSETISINT -o hello_other /users/pro
fs/goubault/pvm/src/hello_other.c -Wl,-aarchive -L/usr/local/pvm/lib/HPPA -lpvm3

        mv hello_other /users/profs/goubault/pvm/src/../bin/HPPA
```

3 La console PVM

```
goubaul1@tarse ~/pvm/src > pvm ~/.xpvm_hosts
```

Les différentes commandes à la console sont

- `conf` permet de lister les différentes machines composant la machine pvm. Par exemple,

  ```
  pvm> conf
  1 host, 1 data format
                      HOST     DTID     ARCH   SPEED
    tarse.polytechnique.fr    40000    ALPHA    1000
  ```

- `add` permet d'ajouter une station à la machine pvm. Par exemple,

  ```
  pvm> conf
  1 host, 1 data format
                      HOST     DTID     ARCH   SPEED
    tarse.polytechnique.fr    40000    ALPHA    1000
  pvm> add radius
  1 successful
                      HOST     DTID
                    radius    80000
  ```

```
pvm> conf
2 hosts, 1 data format
                    HOST     DTID     ARCH   SPEED
  tarse.polytechnique.fr    40000    ALPHA    1000
                  radius    80000    ALPHA    1000
pvm> add royce
1 successful
                    HOST     DTID
                   royce    c0000
pvm> conf
3 hosts, 2 data formats
                    HOST     DTID     ARCH   SPEED
  tarse.polytechnique.fr    40000    ALPHA    1000
                  radius    80000    ALPHA    1000
                   royce    c0000     HPPA    1000
```

- `delete` permet de faire exactement le contraire et d'enlever une machine à la configuration pvm.

- `spawn` permet de lancer une ou plusieurs copies d'un programme pvm. Par exemple `spawn -> hello` permet de lancer une copie du programme `hello` et de rediriger les sorties du programme (où que ce processus soit exécuté) vers la console pvm. `spawn -> 2 hello` permet d'en lancer deux copies. Exemple:

```
pvm> spawn -> hello
[1]
1 successful
t80001
pvm> [1:t80001] i'm t80001
[1:t80001] from tc0001: hello, world from royce
[1:t80001] EOF
[1:tc0001] EOF
[1] finished
pvm> spawn -> 10 hello
[3]
10 successful
t40009
t4000a
t4000b
t4000c
t80009
t8000a
t8000b
tc000a
tc000b
```

```
tc000c
pvm> [3:t4000d] EOF
[3:t40009] i'm t40009
[3:t40009] from t4000d: hello, world from tarse.polytechnique.fr
[3:t40009] EOF
[3:t4000c] i'm t4000c
[3:t4000c] from t4000f: hello, world from tarse.polytechnique.fr
[3:t4000f] EOF
[3:t4000c] EOF
[3:t8000a] i'm t8000a
[3:t8000a] from tc000e: hello, world from royce
[3:tc000a] i'm tc000a
[3:tc000a] from tc000d: hello, world from royce
[3:t8000a] EOF
[3:tc000d] EOF
[3:t4000e] EOF
[3:tc000a] EOF
[3:t8000c] EOF
[3:tc000e] EOF
[3:t8000d] EOF
[3:tc000c] i'm tc000c
[3:tc000c] from t8000c: hello, world from radius.polytechnique.fr
[3:t80009] i'm t80009
[3:t80009] from t8000d: hello, world from radius.polytechnique.fr
[3:tc000c] EOF
[3:t80009] EOF
[3:t4000b] i'm t4000b
[3:t4000b] from tc000f: hello, world from royce
[3:t4000b] EOF
[3:t80009] EOF
[3:t4000b] i'm t4000b
[3:t4000b] from tc000f: hello, world from royce
[3:t4000b] EOF
[3:t8000e] EOF
[3:t4000a] i'm t4000a
[3:t4000a] from t8000e: hello, world from radius.polytechnique.fr
[3:tc000f] EOF
[3:t4000a] EOF
[3:t8000b] i'm t8000b
[3:t8000b] from t4000e: hello, world from tarse.polytechnique.fr
[3:t40010] EOF
[3:tc000b] i'm tc000b
[3:tc000b] from t40010: hello, world from tarse.polytechnique.fr
[3:t8000b] EOF
[3:tc000b] EOF
[3] finished
```

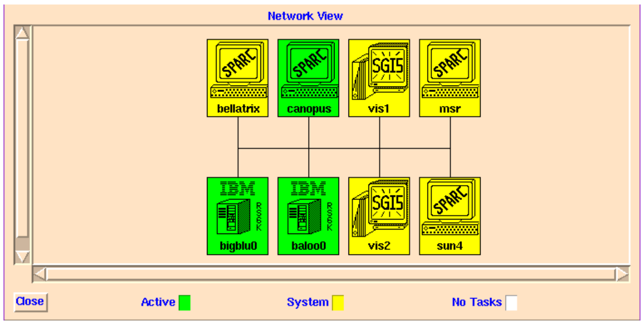

Figure 1: Machine pvm

- `quit` permet de quitter la console pvm sans arrêter les démons `pvmd`.
- `halt` quitte la console pvm et arrête les démons `pvmd`.

4 L'interface XPVM

xpvm est la version X-Windows de pvm. Faire `xpvm` au clavier pour démarrer la console xpvm, mais surtout pas `xpvm &`. Au démarrage, on voit la figure 1.

C'est la représentation de la machine PVM avec les noms et les types des différents ordinateurs la composant. Les menus de cette fenêtre permettent de contrôler les tâches (menu `Tasks`), de contrôler les machines (menu `Hosts`), d'arreter PVM (menus `Reset, Quit, Halt`) et d'avoir de l'aide (menu `Help`). Voir figure 2.

On peut ajouter et enlever des hôtes (`add` et `delete` de PVM) par le menu `Tasks`. On conseille dans les prochains TDs de commencer par tester les programmes sur un seul hôte (la machine sur laquelle vous vous êtes loggée) puis sur deux ou trois, choisies parmi les voisins immédiat. Cela permettra de ne pas surcharger certaines machines.

Le `spawn` de la console PVM se retrouve dans le menu `Tasks`.

Quand on exécute une tâche, on voit dans la partie basse de la fenêtre XPVM le déroulement dans le temps des différentes tâches, voir figure 3.

Chaque couleur représente le type d'activité d'une tâche ("computing", "overhead", "waiting", "messages"). Vous pouvez cliquer sur les flèches représentant les messages ou sur les intervalles de temps pour avoir plus d'informations. Vous pouvez également changer d'echelle de visualisation en cliquant-glissant sur cette fenêtre.

On récupère ce qu'affichent les différentes tâches en demandant dans le menu `View` les "Task Output", voir figure 4.

C'est ce que l'on voyait sous la console PVM (quand on avait mis l'option `->`).

Dans le même menu on peut également demander un graphe du type d'activité des processus en fonction du temps ("Utilization versus Time"), voir figure

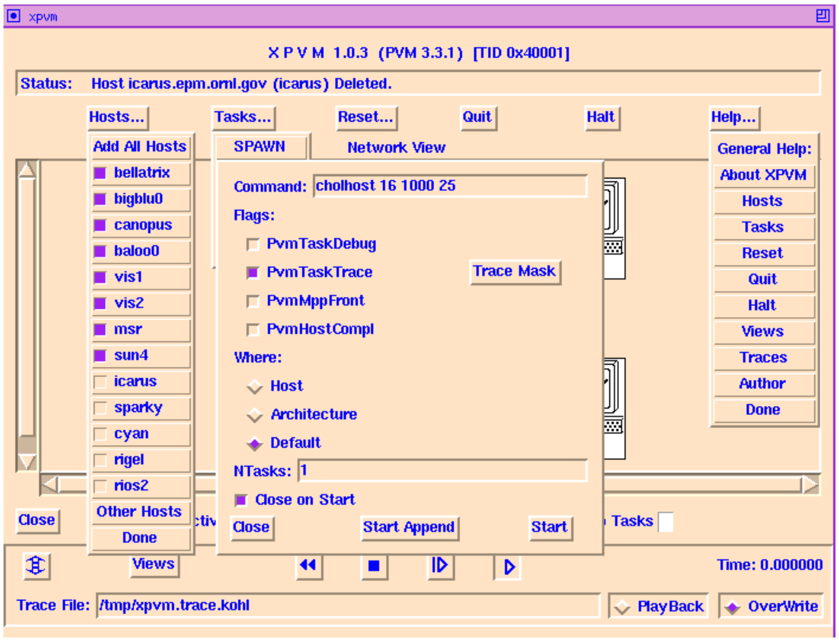

Figure 2: Menus XPVM

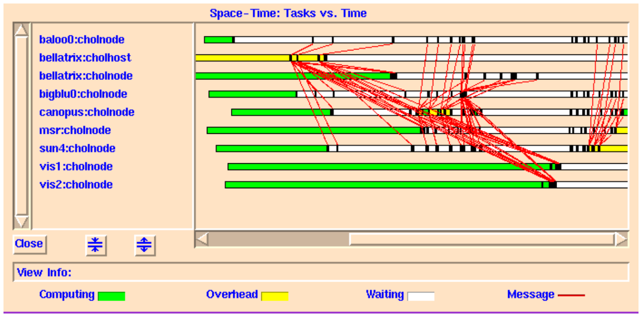

Figure 3: Activité des tâches

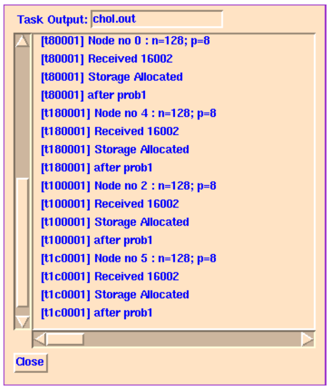

Figure 4: Task Output

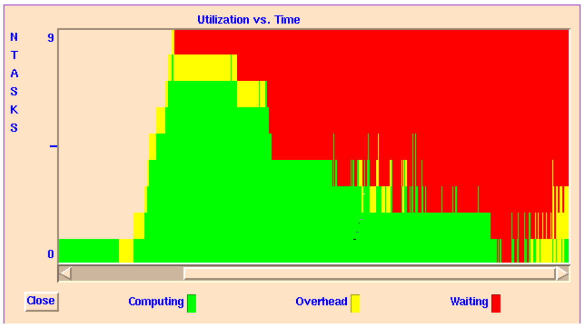

Figure 5: Utilization versus Time

5.

Ou encore, à voir tous les envois et les réceptions de messages entre les tâches (“Call Trace”), voir figure 6.

On peut tout avoir en même temps, voir figure 7.

5 Un programme exemple

Ces programmes vous sont fournis dans la distribution `pvm_install.tar`.

Le maître (`hello.c`),

```
#include <stdio.h>

/* pour avoir acces a la librairie PVM: */
#include "pvm3.h"

main()
{
```

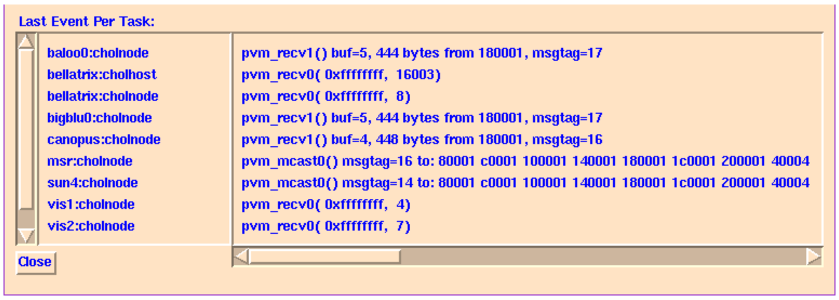

Figure 6: Call Trace

Figure 7: Un écran XPVM au complet

```
int cc, tid;
char buf[100];

/* pour raccrocher le programme au d\'emon pvmd3 */
printf("i'm t%x\n", pvm_mytid());

/* pour lancer un esclave en parallele */
cc = pvm_spawn("hello_other", (char**)0, 0, "", 1, &tid);

/* s'il s'est bien lance... */
if (cc == 1) {
/* recevoir un message de n'importe qui */
cc = pvm_recv(-1, -1);
/* pvm_bufinfo(cc, (int*)0, (int*)0, &tid); */
/* decompacter ce message et ranger une chaine de */
/* caracteres dans la variable buf */
pvm_upkstr(buf);
printf("from t%x: %s\n", tid, buf);

} else
printf("can't start hello_other\n");

/* deconnecter le processus du d\'emon pvmd3 */
pvm_exit();
exit(0);
}
\ed{verbatim}

Le programme esclave ({\tt hello_other.c}),

\begin{verbatim}
/* pour avoir acces a la librairie PVM: */
#include "pvm3.h"

main()
{
int ptid;
char buf[100];

/* trouver le processus parent */
ptid = pvm_parent();

strcpy(buf, "hello, world from ");
gethostname(buf + strlen(buf), 64);

/* initialiser un tampon de communication */
```

```
pvm_initsend(PvmDataDefault);
/* y ranger la chaine de caracteres buf */
pvm_pkstr(buf);
/* envoyer le contenu de ce tampon */
pvm_send(ptid, 1);

/* deconnecter le processus du d\'emon pvmd3 */
pvm_exit();
exit(0);
}
```

6 TD

- Réaliser l'installation de pvm et la tester avec le programme exemple fourni. Essayer xpvm.
- Pour ceux qui auraient fini: au choix,
  - Reécrire le programme de calcul de pi (du cours) en utilisant des groupes pvm.
  - Un petit exercice pour implémenter un tri parallèle (d'après T. Jensen). Soit un tableau $T = (T_i)_{1 \leq i \leq 2N}$ d'entiers. On le suppose réparti sur deux processeurs, soit divisé en deux tableaux $T^1 = (T_i^1)_{1 \leq i \leq N}$ et $T^2 = (T_i^2)_{1 \leq i \leq N}$. Le but est de classer T, c'est à dire d'avoir les propriétés suivantes,
    * T^1 est classé (ordre croissant),
    * T^2 est classé (ordre croissant),
    * tous les éléments de T^1 sont plus petits que tous les éléments de T^2.

    On peut donc classer en parallèle T^1 sur le processeur 1 et T^2 sur le processeur 2, par exemple par un tri par insertion (ou à bulle). Puis on programmera une “fusion” des deux tableaux T^1 et T^2. Pour cela, on fera transiter des données en itérant le procédé suivant,
    * passer $Y = T^1(N)$ au processeur 2 et passer $Z = T^2(1)$ au processeur 1. Si $Y > Z$ alors ranger Z dans le tableau $(T_i^1)_{1 \leq i \leq N-1}$ (respectivement Y dans le tableau $(T_i^2)_{2 \leq i \leq N}$).
    * si $Y \leq Z$ alors stop.