PVM — PARALLEL VIRTUAL MACHINE
ADMINISTRATION SYSTÈME

Philippe Marquet
marquet@lifl.fr
Laboratoire d'informatique fondamentale de Lille
Université des sciences et technologies de Lille
France

Maîtrise d'informatique

Janvier 1998

Installation de PVM : point de vue de l'utilisateur

Les grandes lignes pour me faire une belle petite installation pour utiliser PVM

1. Pouvoir se connecter sur la ferme :
   `rlogin farm?.lifl.fr` ? de 1 à 16
   Mettre à jour mon `.login` et mon `.cshrc` en conséquence.

2. Définition de l'environnement PVM à ajouter dans mon `.cshrc`

```
setenv PVM_ROOT ~/pvm3

if (-r ${PVM_ROOT}/lib/pvmgetarch) then
  setenv PVM_ARCH `${PVM_ROOT}/lib/pvmgetarch`
  set path = ($path ${PVM_ROOT}/lib)
  setenv MANPATH "${MANPATH}:${PVM_ROOT}/man"

  alias pvmgs ~/pvm3/lib/${PVM_ARCH}/pvmgs
  setenv pvmbin ~/pvm3/bin/$PVM_ARCH
  setenv pvmout /tmp/pvml.5408
else
  unsetenv PVM_ROOT
endif
```


3. Créer mon installation propre de PVM ; elle sera située chez moi dans `~/pvm3`
   On suppose que « l'installation système » de PVM est dans `/usr/local/pvm3`

```
cd
mkdir pvm3
cd pvm3
foreach name (conf doc examples include lib man)
  ln -s /usr/local/pvm3/$name
end
mkdir bin
```

4. Test de l'installation de base
   On doit pouvoir exécuter les commandes suivantes sans erreur :
   - Vérification de la définition de l'environnement PVM
     `echo $PVM_ROOT`
   - Vérification de la même chose à distance
     `rsh farm? 'echo $PVM_ROOT'`

   Tant que cela ne marche pas, il faut regarder
   - du coté du `.rhosts`
   - si `PVM_ROOT` est bien défini dans le `.cshrc`
   - si le `.cshrc` ne produit pas d'entrée/sortie (`stty`, `echo`...)

5. On lance PVM
   `pvm`
   Sous le prompt de la console, on regarde la configuration, on ajoute une machine :
   ```
   pvm> conf
   ....
   pvm> add farm?-giga
   ...
   pvm> conf
   ```
   On peut ensuite laisser tourner le daemon PVM
   ```
   pvm> quit
   ```
6. On construit une application PVM, par exemple
   ```
   cd ~/pvm3/examples
   make hello
   make hello_other
   ```


7. On exécute cette application :
   ```
   ~/pvm3/bin/ALPHA/hello
   ~/pvm3/bin/$PVM_ARCH/hello
   $pvmbin/hello
   ```
   Il affiche `hello from "farm?-giga"`, c'est gagné
8. Fin
   Il faut quitter proprement PVM
   ```
   pvm
   pvm already running
   pvm> halt
   libpvm [t40001]: mxfer() EOF on pvmd sock
   ```
   Ce message indique que c'est bien fini !

Compilation d'une application PVM

- Problème : les exécutables doivent se trouver dans `~/pvm3/bin/$PVM_ARCH`
- Une solution :
  - Sources et `Makefile` dans le répertoire courant
  - Répertoire `$PVM_ARCH` pour les objets
    (posibilité d'avoir plusieurs architectures différentes)
  - Compilation de `toto.f` :
    ```
    f77 -I${PVM_ROOT}/include -c toto.f
    mv toto.o $PVM_ARCH/toto.o
    f77 -o toto $PVM_ARCH/toto.o $PVM_ROOT/lib/$PVM_ARCH/libfpvm3.a \
         $PVM_ROOT/lib/$PVM_ARCH/libpvm3.a
    mv toto $PVM_ROOT/bin/$PVM_ARCH
    ```

- Création du `Makefile`
  - ◇ Définition de variables :

```
# Il est nécessaire d'avoir positionné les variables shell
# PVM_ROOT : la racine de pvm3
# PVM_ARCH : l'architecture pour laquelle compiler
#
IncludePath=${PVM_ROOT}/include
PvmLib=${PVM_ROOT}/lib/${PVM_ARCH}/libpvm3.a
PvmFLib=${PVM_ROOT}/lib/${PVM_ARCH}/libfpvm3.a
PvmGLib=${PVM_ROOT}/lib/${PVM_ARCH}/libgpvm3.a
BinDir=${HOME}/pvm3/bin/${PVM_ARCH}
ObjDir=${PVM_ARCH}

FFLAGS=-I${IncludePath}
# Ne pas changer l'ordre des librairies dans la définition suivante
LDLIBS=${PvmFLib} ${PvmLib}
LDFLAGS=
LINK=f77
```


- ◇ Règles de compilation :

```
toto : ${BinDir}/toto
        @echo "$@ done"
${ObjDir}/toto.o : toto.f toto.h
        ${FC} ${FFLAGS} -c toto.f
        mv `basename $@` $@
${BinDir}/toto : ${ObjDir}/toto.o
        ${LINK} ${LDFLAGS} -o `basename $@` ${ObjDir}/toto.o ${LDLIBS}
        mv `basename $@` $@
```

- ◇ Autres entrées utiles pour le `Makefile` :

```
#       .. initialisations ..
init :  ${ObjDir} ${BinDir}
        @echo "initialisations faites"

${ObjDir} :
        mkdir -p $@
${BinDir} :
        mkdir -p $@
```

Initialisation :

`make init`

Environnement système pour PVM

- L'utilisateur doit avoir
  - Un compte sur chacune des machines de la ferme
  - (– Avec le même nom d'utilisateur et le même numéro d'id)
  - La possibilité de lancer des `rsh` entre toutes les machines
  - Un répertoire `~/pvm3/bin` contennant les mêmes exécutables
  - (– Une installation de PVM identique sur toutes les machines)
- Une solution simple :
  - Installation des « pages jaunes » (NIS)
    Partage de `passwd`
  - Équivalencage de toutes les machines (`/etc/hosts.equiv`)
    `rsh` autorisés
  - Montage des « home directories »
    Partage de `~/pvm3/bin`
  - Montage de `/usr/local`
    Partage de l'installation de PVM

Autres solutions

- Pas de « pages jaunes »
  Réplication de `/etc/passwd` sur toutes les machines
  Cohérence de ces fichiers (création de compte, modification de « passwd »)
- Pas d'équivalencage des machines
  `~/.rhosts` de chaque utilisateur
- Pas de montage des « home directories »
  Recopie par l'utillisateur des exécutables lors de chaque compilation
- Pas de montage de `/usr/local`
  Réplication de l'installation de PVM sur l'ensemble des machines

Nettoyage

- Problème : impossible de savoir pour l'administrateur système si un processus PVM tournant est dû à
  - Une application s'exécutant, ou à
  - Un oubli de quitter proprement PVM
- Nettoyage systématique par l'utilisateur
- Utilisation d'un « shell script »
- Nettoyer : taper `halt` sous la console PVM
- Il peut y avoir des problèmes, par exemple si une machine ne répond plus
- Ce que l'on peu faire :
  - `halt`
  - Destruction des fichiers `/tmp/pvm*`
  - Destruction des sockets dans `/tmp/`
- S'il reste des processus PVM ils se termineront sur une erreur

- `killpvm` : exemple de script (`csh`)

```
while ($1 != "")
  echo Checking $1 .....
  rsh $1 -l $USER '                                                      \
  if ($PVM_ROOT/lib/$PVM_ARCH/pvm != `which pvm` && $PVM_ROOT/lib/pvm != `which pvm`) then
    echo PVM is not properly installed on $HOST \!                       \
  else                                                                   \
    echo halt | pvm                                     >& /dev/null  \
    if ($?TMPDIR) then                                                   \
      find $TMPDIR/* -type s -exec /bin/rm -f {} \;  >& /dev/null  \
      /bin/rm -f $TMPDIR/pvm*                           >& /dev/null  \
    endif                                                                \
    find /tmp/* -type s -exec /bin/rm -f {} \;        >& /dev/null \
    /bin/rm -f /tmp/pvm*                                >& /dev/null \
  endif                                                                  \
  '
  shift
end
```


Implémentation de PVM

- Implémentation de PVM sur machines Unix

(• Implémentations spécifiques sur machines multiprocesseurs)

- Grandes lignes :
  - Utilisation des *sockets* Unix pour les communications inter-processus
  - Protocoles TCP et UDP
- Les daemons
  - Un daemon `pvmd` tourne sur chaque nœud d'une machine virtuelle pour chaque utilisateur
  - Les daemeons appartiennent à l'utilisateur
  - Fichier de log du daemon : `/tmp/pvml.`uid
  - Création des *sockets* dans `/tmp`

- Messages
  - Les tâches et les daemons émettent des messages de longueur arbitraire
  - Ces messages peuvent être codés
    - ◇ Machines de formats différents
    - ◇ Utilisation de XDR
  - Communications daemon/daemon
    - ◇ Utilisent UDP (perte, duplication, non conservation de l'ordre des messages)
      $\Rightarrow$ Acquittement, réémission
    - ◇ Découpage des longs messages
    - ◇ Avantage : extensibilité (partage de *sockets*)
  - Communications daemon/tâche et tâche/tâche
    - ◇ Utilisent TCP, ou
    - ◇ Utilisent les *stream-socket* pour les communications «locales »

- Limitations :
  - Mémoire : allocation dynamique des buffers, messages en attente dans le daemon
  - Processus
  - Descripteurs de fichiers (limitation par processus) :
    - ◇ Daemon
      Un par tâche PVM (*socket* TCP)
      Un par tâche PVM « spawn » depuis une autre tâche (« output » redirigé sur un pipe vers le daemon)
      Quelques descripteurs supplémentaires (log...)
    - ◇ Tâche
      Un par tâche avec laquelle elle est en communication directe (*socket*)
      Quelques descripteurs supplémentaires (communication avec le daemon, I/O...)