Projet de parallélisme et concurrence

Cours de Parallélisme

—Maîtrise Informatique—

---

réalisation d'une super-console

L'objet de ce projet est de mettre en place une console PVM plus évoluée que la console fournie avec le système. Il faudra bien entendu que le logiciel fournisse les mêmes services que la console PVM mais en plus il faudra pouvoir lancer des commandes *shell* par exemple.

Le logiciel PVM permet de regrouper plusieurs machines d'architectures hétérogènes (machines séquentielles ou parallèles) reliées par un réseau d'interconnexion en une machine parallèle virtuelle. Cette dernière est alors vue côté utilisateur comme une seule machine mettant en commun des ressources pour la réalisation de calculs. La gestion de la machine virtuelle se fait essentiellement à partir de la console PVM qui offre les possibilités suivantes :

```
pvm> help
help   - Print helpful information about a command
Syntax:  help [ command ]
Commands are:
  add    - Add hosts to virtual machine
  alias  - Define/list command aliases
  conf   - List virtual machine configuration
  delete - Delete hosts from virtual machine
  echo   - Echo arguments
  export - Add environment variables to spawn export list
  halt   - Stop pvmds
  help   - Print helpful information about a command
  id     - Print console task id
  jobs   - Display list of running jobs
  kill   - Terminate tasks
  mstat  - Show status of hosts
  ps     - List tasks
  pstat  - Show status of tasks
  quit   - Exit console
  reset  - Kill all tasks
  setenv - Display or set environment variables
  sig    - Send signal to task
  spawn  - Spawn task
  trace  - Set/display trace event mask
  unalias - Undefine command alias
  unexport - Remove environment variables from spawn export list
  version - Show libpvm version
```

Il existe aussi une interface graphique XPVM qui permet de contrôler plus simplement la machine virtuelle. Très orientés utilisation de la machine virtuelle, ces deux produits ne sont d'aucune aide pour l'installation d'un programme sur plusieurs machines. Or, pour qu'un programme puisse s'exécuter au sein d'une machine parallèle virtuelle, il faut qu'il soit installé sur chacune des machines. L'objet du projet est donc de produire une *super-console* offrant toutes les possibilités de la console usuelle PVM mais permettant en plus de lister les fichiers d'autres machines, de copier un fichier d'une machine à une autre, bref d'exécuter des commandes distantes.

Globalement, il s'agira d'un côté de programmer une interface utilisateur ressemblant à l'interface d'un shell, et d'un autre côté de réaliser un démon (qui sera bien entendu un programme PVM) se chargeant de mettre en œuvre les commandes lancées à partir de la console. Le démon devra s'installer automatiquement sur chacune des machines par lecture d'un fichier de configuration (en tout point identique au fichier de description d'une machine virtuelle). Il devra aussi pouvoir recevoir un ordre de mise à jour si le fichier de configuration a été changé, auquel cas il lancera ou terminera des démons en fonction de la nouvelle configuration. Le rôle des démons sera de satisfaire les demandes de la console. Pour cela, ils devront s'échanger des messages suivant un protocole que vous définirez. Vous aurez aussi à mettre en place un protocole (identique) entre la console et le démon local (lancé sur la même machine que la console).

La console se chargera initialement de lancer le démon qui lui-même lancera les autres démons en fonction du fichier de configuration. Par ailleurs, la super-console devra avoir le même comportement que la console PVM, c'est-à-dire notamment lancer PVM s'il n'est pas déjà lancé. Il faudra prévoir que l'on puisse sortir de la super-console sans terminer PVM mais aussi sans pour autant tuer les démons de la super-console installés sur les différentes machines. Il faudra donc bien dissocier les commandes de sortie de sorte que tous les cas usuels soient possibles (on s'inspirera de la différence entre les commandes `quit` et `halt` de la console PVM).

Ensuite, elle proposera à l'utilisateur de rentrer des commandes. Elle devra notamment donner la possibilité d'exécuter des scripts shell. Par exemple, on pourra utilement rechercher un fichier sur un parc de machines, ou mieux encore installer automatiquement un programme sur plusieurs ordinateurs. Un exemple probant serait de programmer l'installation automatique de la super-console sur un ensemble de machines : il faudra envoyer le makefile, les sources, lancer la compilation, modifier le fichier de configuration puis informer le démon local de la modification du fichier, et ainsi de suite.

La commande `add` qui existe déjà au niveau de la console PVM et qui a pour but d'ajouter une machine à la machine virtuelle, aura au niveau de la super-console un effet double : ajouter la machine et contacter le démon local afin qu'il lance un démon sur la nouvelle machine et informe les démons existants de la modification de la configuration de la machine virtuelle. Les commandes pourront se présenter sous la forme :

```
superconsole> ls machine1:/tmp

   Ex00869          snd.19277          Ex06472          Ex13223
```

où machine1 désigne la machine sur laquelle sera lancée la commande ls. En fait, les noms de fichier devront intégrer au début un nom de machine séparé par ":". Dans le cas où ce nom est absent il est clair que le nom fera référence à un fichier de la machine locale :

```
superconsole> cp machine1:fichier machine2:fichier
superconsole> cp fichier machine2:fichier
```

où machine1 et machine2 désignent des noms de machines. Le transfert du fichier devra se faire au même titre que toutes les autres commandes par les démons. L'exécution d'un script sera le résultat d'une commande comme :

```
superconsole> machine1:install
```

Libre à vous en fin de compte de prévoir les commandes et options intéressantes !

1 Travail à rendre

Vous devrez rendre un **rapport** décrivant les fonctionnalités et les particularités de votre produit. Ce rapport devra contenir les sources de votre projet judicieusement commentés (*i.e.* commentés afin qu'une lecture rapide soit suffisante pour comprendre la structure du logiciel). Vous devrez aussi produire un **démon** superpvm_demon, la **console** superpvm et des **fichiers d'exemples** intéressants montrant les capacités de votre logiciel.

Le projet pourra déboucher sur un stage de fin d'année de maîtrise dont l'objet sera de reprendre ce travail dans l'optique d'en faire une contribution internationale pour les utilisateurs de PVM.

2 Informations supplémentaires

En ce qui concerne le logiciel PVM, voici quelques adresses ou *URL Internet* utiles :

1. Ftp, ftp netlib2.cs.utk.edu, cd pvm3/book, get pvm-book.ps
2. Email, netlib@ornl.gov, message : send pvm-book.ps from pvm3/book
3. Http, http: //www.epm.ornl.gov/pvm/pvm_home.html
4. Http, http: //www.netlib.org/pvm3/book/pvm-book.html