

## **PVM "Parallel Virtual Machine" Les aspects communication**

**Gilles Requilé**

*Laboratoire de Mécanique et Génie Civil - CNRS URA 1214 - Université Montpellier II  
requile@lmgc.univ-montp2.fr*

### **INTRODUCTION**

Le système **PVM** ("Parallel Virtual Machine") **est un logiciel** constitué d'un ensemble de programmes et de bibliothèques permettant d'utiliser des machines de calcul hétérogènes, insérées dans un environnement de réseaux basés sur le protocole INTERNET (IP), pour mettre en oeuvre des technologies **de calcul parallèle et distribué**.

PVM est issu d'un projet interne de l'Oak Ridge National Laboratory (ORNL) aux USA en 1989. La première version a servi de base en 1991 à un projet plus vaste commun à trois universités des USA et soutenu par plusieurs programmes tant gouvernementaux qu'industriels [1].

Ces techniques de calcul parallèle sont maintenant utilisées dans pratiquement toutes les disciplines scientifiques. La **mise à disposition des codes sources de PVM dans le "domaine public"** ainsi que le fort développement des réseaux de télécommunications, en particulier la couverture mondiale réalisée par les protocoles IP et leur présence sur toutes sortes de plates-formes, ont été des éléments déterminants dans son succès.

Deux autres phénomènes ont facilité sa popularisation :

- d'une part, l'accès aux machines spécifiques à multiprocesseurs reste toujours très onéreux et d'une complexité importante ; les logiciels permettant d'en utiliser les ressources sont encore fortement "propriétaire" (bien que dans ce dernier domaine les recherches sur des langages "portables" avancent rapidement) ;
- d'autre part la banalisation des stations de travail, dont la puissance ne cesse d'augmenter, et qui sont devenues d'un abord "simple" et d'un prix attractif.

**La force de PVM est d'utiliser** cet existant maintenant bien connu des utilisateurs - système d'exploitation, **langages de programmation normalisés** (FORTRAN, C) - et de n'introduire essentiellement que deux concepts non directement familiers aux scientifiques : le passage de message, l'utilisation de la mémoire distribuée (hormis bien sûr les techniques liées à l'algorithmique parallèle elle-même).

Dans cet exposé nous nous bornerons à décrire les aspects communication (protocoles ...).

### **PRINCIPES DE PVM**

PVM est implanté aussi bien sur les machines monoprocesseur (dans ce cas la technique de la mémoire distribuée est utilisée) que sur des machines multiprocesseurs à architecture explicitement parallèle (dans ce cas la gestion de la mémoire partagée est possible).

Afin de simplifier et pour mettre en lumière les mécanismes propres aux communications nous ne considérerons dans cet exposé que des machines monoprocesseur interconnectées via des réseaux IP, ce qui est le cas courant d'un groupe de station de travail et nous

utiliserons la terminologie d'UNIX pour parler des fonctions du système d'exploitation, bien que d'autres systèmes d'exploitation supportent PVM.

Posons aussi, très schématiquement, que la parallélisation d'un programme complexe consiste à découper certaines étapes en séquences indépendantes dont l'exécution peut être, à un moment donné, répartie entre les différentes machines disponibles. Chaque machine possède son propre système d'exploitation, ses propres ressources : disque, mémoire, processeur ... **Le processeur sera identifié par un nombre entier.** Une table sera créée pour assurer la **correspondance entre adresse de machine au sens IP et ce numéro de processeur.**

Ces **séquences indépendantes de programme** sont, dans la terminologie PVM, dénommées **tâches**. Chaque tâche a deux fonctions :

- réaliser les **traitements** prévus par la séquence de programme sur la partie des données qui lui sont affectées à ce moment là,
- assurer les **communications** avec le programme principal (lui-même une tâche) et éventuellement les autres tâches (échange des données initiales et des données traitées, informations de synchronisation ...).

Une **tâche PVM** sera, sur la machine, constituée par un **process au sens UNIX**. La machine virtuelle va donc fournir les moyens pour **identifier chaque tâche** indépendamment du système d'identification des process : un TID ("**Task Identifier**") lui sera attribué ; schématiquement ce sera la concaténation du numéro du processeur et du PID du process.

<b>S 1 bit</b>	<b>G 1 bit</b>	<b>N° processeur 12 bits</b>	<b>N° process 18 bits</b>
----------------	----------------	------------------------------	---------------------------

Figure 1 : *Format des TID (32 bits)*

S : TID pour pvmd ou pour tâche      G : multicast vers les tâches en groupe

On remarquera la possibilité de créer des **groupes de tâches** ce qui permet par exemple de réaliser des synchronisations. Ces groupes sont alors **adressables par multicast**.

Les mécanismes de communication inter-tâches sont donc essentiels dans PVM. Ils sont basés sur la technique de **passage de message** ("Passing Message") dont la taille n'est limitée que par les contraintes locales de chaque machine (mémoire ...). Des protocoles spécifiques, bâtis sur UDP et TCP, ont été conçus pour réaliser d'une manière optimale ces **communications inter-process**.

## LE SYSTEME PVM

Le logiciel PVM est composé de trois parties :

- le "daemon" (**pvmd**) lancé sur chaque machine (processeur) : la machine virtuelle est l'ensemble des pvmd explicitement mis en relations entre-eux par une déclaration de l'utilisateur. Ces "daemon", à l'écoute des requêtes externes et internes sont aussi chargés de router les messages entre tâches.  
**Les pvmd coopèrent entre-eux par l'intermédiaire de "sockets" UDP.**
- une librairie de primitives (**libpvm**) permet à l'utilisateur de gérer et d'ordonner lui-même la coopération des tâches (fonctions C ou sous-programmes FORTRAN) au sein de son programme : envoi et réception de messages, lancement de process, coordination des tâches, modifications de la configuration, ...  
**Les tâches coopèrent entre-elles par l'intermédiaire de "sockets" TCP.**
- une console, sorte de "shell", est une tâche créée par un programme (**pvm**) qui va servir à gérer la "vie" de l'ensemble des tâches de la machine virtuelle (comme le "shell" sous

UNIX permet de gérer la vie des process) : kill, status, ajout ou suppression de machines... **xpvm est une forme évoluée de cette console dans un environnement X11.**

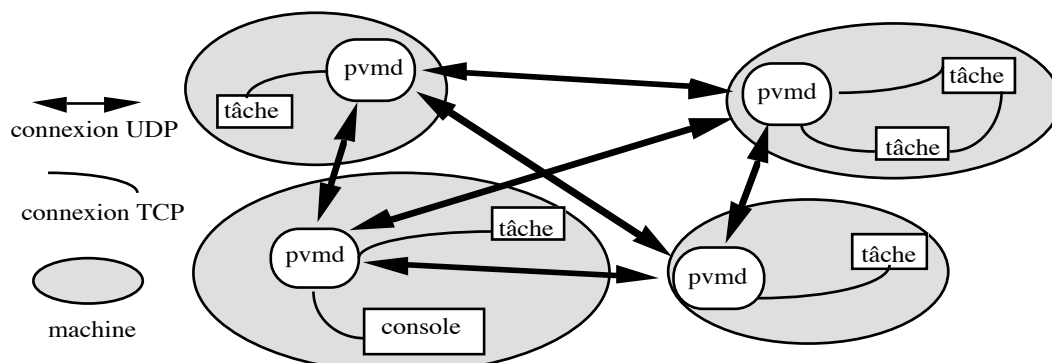


Figure 2 : Schéma des modes de connexion du système PVM

Les messages entre tâches exécutées sur deux processeurs différents transiteront par l'intermédiaire des "daemon" pvmd qui en assurent le routage.

## LES PROTOCOLES

Deux types de connexions sont établies avec des comportements différents :

- Les échanges **pvmd-pvmd** utilisent des connexions de type "**datagram**" sur **UDP** (nécessité d'un grand nombre de connexions simultanément ouvertes).

<b>adresse TID destination finale 32 bits</b>																
<b>adresse TID de l'émetteur 32 bits</b>																
<b>numéro de séquence</b>	<b>numéro d'acquittement</b>															
<table border="1"> <tr> <td>A</td><td>F</td><td>D</td><td>E</td><td>S</td> </tr> <tr> <td>C</td><td>I</td><td>A</td><td>O</td><td>O</td> </tr> <tr> <td>K</td><td>N</td><td>T</td><td>M</td><td>M</td> </tr> </table>	A	F	D	E	S	C	I	A	O	O	K	N	T	M	M	<b>non utilisé</b>
A	F	D	E	S												
C	I	A	O	O												
K	N	T	M	M												

Figure 3 : En-tête de paquet entre "daemon" pvmd

Les numéros de séquence et d'acquittement (sur 16 bits) sont incrémentés à chaque transmission à la fin de laquelle une mesure de "round-trip delay" est effectuée pour être utilisé, parmi d'autres critères (caractéristiques de la machine, ...), à établir la stratégie de répartition des tâches entre processeur.

Les bits SOM et EOM permettent de gérer la fragmentation (premier et dernier paquet du message) indispensable aux longs messages.

Le bit DAT indique que le paquet véhicule des données et que le numéro de séquence est valide et le bit ACK indique que le numéro d'acquittement est valide.

Le bit FIN permet au pvmd d'informer qu'il va mettre fin à la connexion.

- Les échanges **tâche-tâche ou pvmd-tâche** utilisent des connexions de type "**stream**" sur **TCP** (nécessité d'une connexion fiabilisée).

On remarque ici l'absence de mécanisme d'acquittement. Seuls les bits EOM et SOM permettent de gérer la fragmentation comme précédemment.

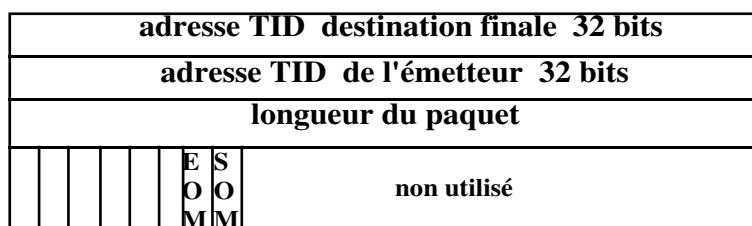


Figure 4 : En-tête de paquet entre "daemon" pvmd et tâche ou entre tâche et tâche

En ce qui concerne les données véhiculées via les messages, PVM permet d'opérer une conversion dans le mode de représentation. En effet les plates-formes constituant la machine virtuelle peuvent être hétérogènes ; le **protocole XDR** peut alors être employé. L'utilisateur devra explicitement utiliser, dans son programme, les primitives adéquates destinées à encoder (ou décoder) le message.

## CONFIGURATION ET MISE EN OEUVRE

Le système PVM est installé directement sous le répertoire de l'utilisateur. Le premier "daemon" pvmd est lancé manuellement par l'utilisateur en même temps que la console pvm. Ensuite, si l'on simplifie, l'ajout de machine est fait par pvmd qui utilise **rsh** : ceci nécessite la présence du fichier **.rhost** sur la machine distante. Si l'on ne souhaite pas utiliser ce mécanisme, il est possible d'employer **rexec**, qui impose alors une validation de l'opération par le mot de passe de l'utilisateur sur la machine concernée.

Bien sur, une version du système PVM devra exister sur la machine distante, où l'utilisateur devra pouvoir être reconnu et identifiable ("login" et "password"). La situation la plus simple consiste à travailler sur des machines homogènes, à ne posséder qu'une seule version de PVM et à rendre visible le répertoire de l'utilisateur sur toutes les machines, via NFS.

La liste ainsi que les caractéristiques de connexion des machines insérables dans la machine virtuelle seront indiquées dans un fichier situé dans le répertoire de l'utilisateur, en outre un certain nombre de variables d'environnement seront initialisées via le "shell". Le "resolver" devra être en service sur toutes les machines concernées. L'environnement X11 est fortement conseillé pour utiliser les outils annexes au système PVM lui-même : console XPVM, HeNCE (outil d'aide à la parallélisation)... Une abondante littérature est disponible [2] et [3] en plus du manuel [1] pour l'assistance à la mise en oeuvre.

## REFERENCE, DISTRIBUTION ET DOCUMENTATIONS

[1] Al Geist, Adam Geguelin, Jack Dongarra, Weilcheng Jiang, Robert Manchek, Vaidy Sunderman **"PVM : Parallel Virtual Machine. A users' Guide and Tutorial for Networked Parallel Computing"** The MIT Press Cambridge, Massachusetts, London, England (1994). *Disponible en POSTSCRIPT (voir [2] ci-après).*

[2] **Où trouver PVM :**

A la source : <http://www.netlib.org/pvm3/index.html>  
 Par e-mail : echo "send index from pvm3" | mail netlib@ornl.gov  
 En France (mirror): <ftp://ftp.irisa.fr/pub/mirrors/netlib/pvm3>

[3] **Groupe de discussion sur les news :** comp.parallel.pvm

