```
> cat hostfile
```

```
# Deux stations bi-processeurs superscalaires IBM G30
* lo=dupont dx=$PVM_ROOT/lib/pvmd \
          ep=bin/$PVM_ARCH wd=$PWD sp=1000
mira.idris.fr
vega.idris.fr
& mach1.idris.fr
& mach2.idris.fr
& mach3.idris.fr
& mach4.idris.fr
# Cinq noeuds d'une machine MPP IBM SP2
* lo=dupuis dx=bin/mon_pvmd \
          ep=pvm3/bin/$PVM_ARCH sp=3000
p01.rhea.cnusc.fr
p02.rhea.cnusc.fr
p03.rhea.cnusc.fr
p04.rhea.cnusc.fr
p05.rhea.cnusc.fr
# Un supercalculateur vectoriel CRAY C94
atlas.idris.fr lo=mowgli dx=$PVM_ROOT/lib/pvmd \
          sp=2000
# Une machine MPP CRAY T3E
aleph.idris.fr lo=rbid001 sp=4000
```

```
> pvmd3 hostfile &
```

```
1  program qui_je_suis
2    implicit none
3    include 'fpvm3.h'
4    integer, allocatable, dimension(:) :: tids
5    integer :: nprocs, MonTid, TidDuPere, numt, info
6
7    call PVMFMYTID(MonTid)
8    call PVMFPARENT(TidDuPere)
9
10   if( TidDuPere == PVMNOPARENT ) then
11
12     read(*,*) nprocs
13     allocate( tids(0:nprocs-1) )
14     tids(0) = MonTid
15     call PVMFSPAWN('QuiJeSuis',PVMTASKDEFAULT,  &
16                    '*', nprocs-1, tids(1), numt)
17     if( numt /= nprocs-1 ) stop
18
19   end if
20   print *,'Mon TID: ',MonTid, &
21           ' ; celui du pere: ',TidDuPere
22
23   call PVMFEXIT(info)
24 end program qui_je_suis
```

```
1  program machines
2   implicit none
3   include 'fpvm3.h'
4   integer, parameter :: nprocs=6, nmach=4
5   integer, dimension(nprocs) :: tids
6   integer, dimension(nmach)  :: vrai
7   character(len=14),dimension(nmach):: machine
8   integer                    :: info, bonne, numt, k
9   integer                    :: MonTid,TidDuPere,etat
10
11  call PVMFMYTID(MonTid)
12  call PVMFPARENT(TidDuPere)
13  if( TidDuPere == PVMNOPARENT ) then
14    tids(1)=TidDuPere ; bonne = 0 ; k = 1
15    machine(1)='mach1.idris.fr'
16    machine(2)='mach2.idris.fr'
17    machine(3)='mach3.idris.fr'
18    machine(4)='mach4.idris.fr'
```

```
1     do i = 1, nmach
2      call PVMFMSTAT(machine(i), etat)
3      if(etat == 0) then
4        bonne = bonne + 1
5        vrai(bonne) = i
6        call PVMFADDHOST(machine(i), info)
7      endif
8     enddo
9     if (bonne == 0) then
10      print*,'Aucune machine n''est disponible'
11      call PVMFEXIT(info)
12      stop
13    endif
14    do i = 2, nprocs
15      call PVMFSPAWN('MonProg', PVMTASKHOST, &
16                  machine(vrai(k)),1,tids(i),numt)
17      if ( k == bonne ) then
18        k = 1
19      else
20        k = k+1
21      endif
22    enddo
23   endif
24   call PVMFEXIT(info)
25 end program machine
```

```
1  program qui_je_suis
2    implicit none
3    include 'fpvm3.h'
4    integer, allocatable, dimension(:) :: tids
5    integer :: nprocs,MonTid,TidDuPere,numt,info,rang
6
7    call PVMFMYTID(MonTid)
8    call PVMFPARENT(TidDuPere)
9    call PVMFCATCHOUT( 1, info)
10   call PVMFJOINGROUP('UNIVERS', rang)
11
12   if( TidDuPere == PVMNOPARENT ) then
13     read(*,*) nprocs
14     allocate( tids(0:nprocs-1) )
15     tids(0) = MonTid
16     call PVMFSPAWN('QuiJeSuis',PVMTASKDEFAULT,&
17                    '*',nprocs-1, tids(1), numt)
18   end if
19
20   print *,'Mon TID: ',MonTid, '; Mon rang: ',rang,&
21           '; TID du pere ',TidDuPere
22
23   call PVMFLVGROUP(info)
24   call PVMFEXIT(info)
25 end program qui_je_suis
```

```
1  program point_a_point
2   implicit none
3
4   include 'fpvm3.h'
5   integer, parameter :: etiquette=100
6   integer    :: valeur,rang,nprocs,bidon,info
7   integer    :: tid_recepteur,tid_emetteur
8
9   call PVM_INIT(rang,nprocs)
10
11  if (rang == 1) then
12    valeur=1000
13    call PVMFGETTID('UNIVERS', 5, tid_recepteur)
14    call PVMFPSEND(tid_recepteur,etiquette,valeur,1,&
15                   INTEGER4,info)
16  elseif (rang == 5) then
17    call PVMFGETTID('UNIVERS', 1, tid_emetteur)
18    call PVMFPRECV(tid_emetteur,etiquette,valeur,1,&
19                   INTEGER4,bidon,bidon,bidon,info)
20    print *,'Moi, processus 5, j''ai reçu ', &
21            valeur,' du processus 1.'
22  end if
23
24  call PVMFEXIT(info)
25 end program point_a_point
```

```
1  program anneau
2   implicit none
3   include 'fpvm3.h'
4   integer,parameter :: etiquette=100
5   integer :: nprocs,rang,valeur,bidon,info
6   integer :: num_proc_precedent,num_proc_suivant
7   integer :: tid_proc_precedent,tid_proc_suivant
8
9   call PVM_INIT(rang, nprocs)
10
11  num_proc_precedent=mod(nprocs+rang-1,nprocs)
12  num_proc_suivant=mod(rang+1,nprocs)
13
14  call PVMFGETTID('UNIVERS',num_proc_suivant, &
15                  tid_proc_suivant)
16  call PVMFPSEND(tid_proc_suivant,etiquette,  &
17           rang+1000,1,INTEGER4,info)
18  call PVMFGETTID('UNIVERS',num_proc_precedent, &
19                   tid_proc_precedent)
20  call PVMFPRECV(tid_proc_precedent,etiquette,  &
21           valeur,1,INTEGER4,bidon,bidon,bidon,info)
22
23  print *,'Moi, processus ',rang,', j''ai reçu ', &
24         valeur,' du processus ',num_proc_precedent
25  call PVMFEXIT(info)
26 end program anneau
```

```
1  program heterogene
2    implicit none
3    include 'fpvm3.h'
4    integer,parameter :: etiquette=100
5    integer           :: rang,nprocs,bufid,info,genre
6    integer           :: tid_recepteur,tid_emetteur
7    character(len=7)  :: nom
8    real              :: poids
9    call PVM_INIT(rang,nprocs)
10   if (rang == 1) then
11      nom='chameau' ; genre=1010 ; masse=100.8
12      call PVMFINITSEND(PVMDATADEFAULT, bufid)
13      call PVMFPACK(STRING, nom, 1, 1, info)
14      call PVMFPACK(INTEGER4, genre, 1, 1, info)
15      call PVMFPACK(REAL4, poids, 1, 1, info)
16      call PVMFGETTID('UNIVERS', 5, tid_recepteur)
17      call PVMFSEND(tid_recepteur, etiquette, info)
18   elseif (rang == 5) then
19      call PVMFGETTID('UNIVERS', 1, tid_emetteur)
20      call PVMFRECV(tid_emetteur, etiquette, bufid)
21      call PVMFUNPACK(STRING, nom, 1, 1, info)
22      call PVMFUNPACK(INTEGER4, genre, 1, 1, info)
23      call PVMFUNPACK(REAL4, poids, 1, 1, info)
24   end if
25   call PVMFEXIT(info)
26 end program heterogene
```

```
1  program distribution
2    implicit none
3    include 'fpvm3.h'
4    integer, parameter :: etiquette=100
5    integer            :: rang,nprocs,bufid,info
6    integer            :: tid_emetteur,tids(3)
7    real               :: A(500,600), B(600)
8
9    call PVM_INIT(rang,nprocs)
10   if (rang == 0) then
11      call random_number(a)
12      call PVMFINITSEND(PVMDATARAW, bufid)
13      call PVMFPACK(REAL4, A(4,1), 600, 500, info)
14      call PVMFGETTID('UNIVERS', 1, tids(1))
15      call PVMFGETTID('UNIVERS', 3, tids(2))
16      call PVMFGETTID('UNIVERS', 5, tids(3))
17      call PVMFMCAST(3,  tids, etiquette, info)
18   elseif (rang==1 .or. rang==3 .or. rang==5) then
19      call PVMFGETTID('UNIVERS', 0, tid_emetteur)
20      call PVMFRECV(tid_emetteur, etiquette, bufid)
21      call PVMFUNPACK(REAL4, B, 600, 1, info)
22      print *,'Moi, processus ',rang,', j''ai reçu',&
23                   ' A(4,:) dans B(:)'
24   end if
25   call PVMFEXIT(info)
26 end program distribution
```

```
1  program non_bloquant
2   implicit none
3   include 'fpvm3.h'
4   integer, parameter     :: na=256,nb=200
5   integer, parameter     :: m=2048,etiquette=1111
6   real, dimension(na,na):: a
7   real, dimension(nb,nb):: b
8   real, dimension(m,m)   :: c
9   real, dimension(na)    :: pivota
10  real, dimension(nb)    :: pivotb
11  integer                :: rang,nprocs,info, &
12                            drapeau,bufid,oldval &
13                            tid_emetteur,tid_recepteur
14
15 !*** Initialisation PVM
16  call PVM_INIT( rang, nprocs )
17
18 !*** Initialisation des tableaux
19  call random_number(a)
20  call random_number(b)
21
22 !*** Choix du protocole de communication
23  if (rang == 0 .or. rang == 1) &
24    call PVMFSETOPT(PVMROUTE,PVMROUTEDIRECT,oldval)
```

```
25  if (rang == 0) then
26    call random_number(c)
27  !*** J'envoie un gros message
28    call PVMFINITSEND(PVMDATAINPLACE, bufid )
29    call PVMFPACK(REAL4,c,m*m,1,info)
30    call PVMFGETTID('UNIVERS', 1, tid_recepteur)
31    call PVMFSEND(tid_recepteur, etiquette, info)
32  !*** Ce calcul modifie le contenu du tableau C
33    c(1:nb,1:nb) = matmul(a(1:nb,1:nb),b)
34  elseif (rang == 1) then
35    drapeau = 0 ; bufid = 0
36    call PVMFGETTID('UNIVERS', 0, tid_emetteur)
37    do while ( bufid == 0 )
38  !*** Je reçois le gros message, si arrivé
39      call PVMFNRECV(tid_emetteur, etiquette, bufid)
40      if ( bufid > 0 ) then
41  !*** Le message est reçu, je le décompacte
42        call PVMFUNPACK(REAL4,c,m*m,1,info)
43  !*** Ce calcul dépend du message précédent
44        a(:,:) = transpose(c(1:na,1:na))
45      elseif( bufid == 0 .and. drapeau == 0) then
46  !*** Ce calcul est indépendant du message
47        call sgetrf(nb, nb, b, nb, pivotb, info)
48        drapeau = 1 ; end if ; end do ; end if
49  call PVMFEXIT(info)
50  end program non_bloquant
```

```
 ...
 elseif (rang == 1) then
   drapeau = 0 ; bufid = 0
   do while ( bufid == 0 )
!*** Je teste l'arrivée d'un message
     call PVMFPROBE(-1,etiquette,bufid)
     if ( bufid > 0 ) then
!*** Un message est arrivé
       call PVMFBUFINFO(bufid,taille,etiquette, &
                        tid_emetteur,info)
       call PVMFRECV(tid_emetteur,etiquette,bufid)
!*** Le message est reçu, je le décompacte
       call PVMFUNPACK(REAL4,c,taille/4,1,info)
!*** Ce calcul dépend du message précédent
       a(:,:) = transpose(c(1:na,1:na))
     elseif( bufid == 0 .and. drapeau == 0) then
!*** Ce calcul est indépendant du message
       call sgetrf(nb, nb, b, nb, pivotb, info)
       drapeau = 1
     end if
   end do
 end if
```

```
1  program bcast
2    implicit none
3    include 'fpvm3.h'
4    integer :: nprocs=4,etiquette=100
5    integer :: rang,valeur,bufid,info, &
6                bidon,tid_emetteur
7
8    call PVM_INIT(rang, nprocs)
9    call PVMFBARRIER("UNIVERS", nprocs, info)
10
11   if (rang == 2) then
12     valeur=rang+1000
13     call PVMFINITSEND(PVMDATADEFAULT, bufid)
14     call PVMFPACK(INTEGER4, valeur,1, 1, info)
15     call PVMFBCAST("UNIVERS", etiquette, info)
16   else
17     call PVMFPGETTID("UNIVERS", 2, tid_emetteur)
18     call PVMFPRECV(tid_emetteur,etiquette, valeur,&
19             1,INTEGER4,bidon,bidon,bidon,info)
20     print *,'Moi, processus ',rang,', j''ai reçu ',&
21           valeur,' du processus 2'
22   end if
23   call PVMFBARRIER("UNIVERS", nprocs, info)
24   call PVMFLVGROUP("UNIVERS",info)
25   call PVMFEXIT(info)
26 end program bcast
```

```
1  program scatter
2   implicit none
3
4   include 'fpvm3.h'
5   integer,parameter :: nb_valeurs=128
6   integer           :: nprocs=4,etiquette=100
7   integer           :: rang,longueur_tranche,i,info
8   real,allocatable,dimension(:) :: valeurs,donnees
9
10  call PVM_INIT(rang, nprocs)
11  call PVMFBARRIER("UNIVERS", nprocs, info)
12
13  longueur_tranche=nb_valeurs/nb_processus
14  allocate(donnees(longueur_tranche))
15
16  if (rang == 2) then
17    allocate(valeurs(nb_valeurs))
18    valeurs(:)=(/(1000.+i,i=1,nb_valeurs)/)
19  end if
```

```
20  call PVMFSCATTER(donnees,valeurs,longueur_tranche,&
21         REAL4,etiquette,"UNIVERS",2,info)
22
23  print *,'Moi, processus ',rang,', j''ai reçu ', &
24    donnees(1),' à ',donnees(longueur_tranche), &
25    ' du processus 2'
26
27  call PVMFBARRIER("UNIVERS", nprocs, info)
28  call PVMFLVGROUP("UNIVERS",info)
29  call PVMFEXIT(info)
30 end program scatter
```

```
> cat donnee
4
> scatter < donnee
Moi, pr. 0, j'ai reçu 1001. à 1032. du pr. 2
Moi, pr. 1, j'ai reçu 1033. à 1064. du pr. 2
Moi, pr. 3, j'ai reçu 1097. à 1128. du pr. 2
Moi, pr. 2, j'ai reçu 1065. à 1096. du pr. 2
```

```
1  program gather
2   implicit none
3
4   include 'fpvm3.h'
5   integer, parameter :: nb_valeurs=128
6   integer            :: nprocs=4, etiquette=100
7   integer            :: rang,longueur_tranche,i,info
8   real, dimension(nb_valeurs)     :: donnees
9   real, allocatable, dimension(:) :: valeurs
10
11  call PVM_INIT(rang, nprocs))
12  call PVMFBARRIER("UNIVERS", nprocs, info)
13
14  longueur_tranche=nb_valeurs/nb_processus
15  allocate(valeurs(longueur_tranche))
16
17  valeurs(:)=(/(1000.+rang*longueur_tranche+i, &
18                i=1,longueur_tranche)/)
19
20  call PVMFGATHER(donnees,valeurs,longueur_tranche,&
21        REAL4,etiquette,"UNIVERS",2,info)
```

```
22  if (rang == 2) &
23     print *,'Moi, processus 2', j''ai reçu ', &
24             donnees(1),
25             ' ... ',donnees(longueur_tranche+1), &
26             ' ... ',donnees(nb_valeurs)'
27
28  call PVMFBARRIER("UNIVERS", nprocs, info)
29  call PVMFLVGROUP("UNIVERS",info)
30  call PVMFEXIT(info)
31 end program gather
```

```
> cat donnee
4
> gather < donnee
Moi, pr. 2, j'ai reçu 1001. ... 1033. ... 1128.
```

```
1  program reduce
2   implicit none
3   include 'fpvm3.h'
4   integer :: nprocs=4,etiquette=100
5   integer :: rang,valeur,info
6   external PVMSUM
7
8   call PVM_INIT(rang, nprocs)
9   call PVMFBARRIER("UNIVERS", nprocs, info)
10  if (rang == 0) then
11     valeur=1000
12  else
13     valeur=rang
14  endif
15
16  call PVMFREDUCE(PVMSUM,valeur,1,INTEGER4, &
17                  etiquette,"UNIVERS",0,info)
18
19  if (rang == 0) &
20     print *,'Moi, processus 0, j''ai pour ', &
21             'valeur de la somme globale ',valeur
22  call PVMFBARRIER("UNIVERS", nprocs, info)
23  call PVMFLVGROUP("UNIVERS",info)
24  call PVMFEXIT(info)
25 end program reduce
```

```
1  program monde
2   include 'fpvm3.h'
3   integer, parameter :: m=100,etiquette=100
4   integer            :: nprocs=8,rang,info
5   real, dimension(m)         :: A
6   real, dimension(m,nprocs) :: B
7
8   call PVM_INIT(rang, nprocs)
9   call PVMFBARRIER('UNIVERS', nprocs, info)
10
11  a(:)=real(rang)
12
13  call PVMFGATHER(B,A,m,REAL4,etiquette,'UNIVERS', &
14                  2,info)
15
16  if(rang == 2) print *,B(:,:)
17  call PVMFBARRIER('UNIVERS', nprocs, info)
18  call PVMFLVGROUP('UNIVERS', info)
19  call PVMFEXIT(info)
20 end program monde
```

```
1  program GroupePair
2   include 'fpvm3.h'
3   integer,parameter :: m=16,etiquette=100
4   integer           :: nprocs=8,rang,info,tid_de_2
5   integer           :: rang_de_2_dans_pair, &
6                        nprocs_pair,rang_dans_pair
7   real, dimension(m)           :: A
8   real, dimension(m,nprocs/2) :: B
9
10  call PVM_INIT(rang, nprocs)
11 !*** Initialisation du vecteur A
12  A(:) = real(rang)
13 !*** Le nombre de processus de rang pair
14  nprocs_pair = int(nprocs/2) + 1
```

```
15  if( mod(rang,2) == 0 ) then
16    call PVMFJOINGROUP('PAIR', rang_dans_pair)
17    call PVMFBARRIER('PAIR', nprocs_pair, info)
18    call PVMFGETTID('UNIVERS', 2, tid_de_2)
19    call PVMFGETINST('PAIR',tid_de_2, &
20                rand_de_2_dans_pair)
21
22    call PVMFGATHER(B,A,m,REAL4,etiquette, &
23                 'PAIR',rand_de_2_dans_pair,info)
24
25    if(rang_dans_pair == rand_de_2_dans_pair) &
26        print *,B(:,:)
27    call PVMFBARRIER('PAIR', nprocs_pair, info)
28    call PVMFLVGROUP('PAIR', info)
29  end if
30
31  call PVMFBARRIER('UNIVERS', nprocs, info)
32  call PVMFLVGROUP('UNIVERS', info)
33  call PVMFEXIT(info)
34 end program GroupePair
```