Rinnakkaistaminen heterogeenisessä tietokoneympäristössä

Parallel Virtual Machine (PVM)

Sami Saarinen
(sbs@csc.fi)

9. kesäkuuta 1993

Tämä dokumentti löytyy `compress`oidussa muodossa CSC:n koneilta hakemistosta `/usr/local/bin/pvm3/doc` nimellä `pvm3_kalvot_090693.ps.Z`.

Mikä heterogeeninen ympäristö ?

- **Laskentaympäristö, jossa useampi kuin yksi tietokone osallistuu tehtävän ratkaisemiseen.**
- **Tietokoneet voivat olla saman tai eri valmistajan laitteita.**
- **Ratkaistava tehtävä saattaa olla rinnakkaistettu suoraan eri koneille, yhtäaikaa toimivaksi *tai* se voi olla esimerkiksi kahdesta peräkkäisestä toiminnosta (laskenta + grafiikka) koostuva kokonaisuus.**
- **Kuorman jakamisen kannalta saattaa olla viisasta ajaa vektoroituvat osat superkoneella ja skalaariosat työasemilla.**

Parallel Virtual Machine (PVM)

- **Aliohjelmakirjasto ja laskentaympäristö, jolla voidaan yhdistää usean UNIX-koneen tai prosessorin laskentakapasiteetti yhdeksi (rinnakkaiseksi) laskentaresurssiksi.**
- **Kehitetään yhteistyönä Oak Ridge National Laboratoryssä (ORNL), University of Tennesseessä ja Emory Universityssä (Atlanta).**
- **Ensiversio ilmestyi vuoden 1992 alussa.**
  Ensikokeiluja CSC:ssä kesällä 1992.
- **Nykyversio on PVM 3.1, mutta usein näkee vielä käytettävän "vanhaa" versiota 2.4.2.**
- ***Public Domain*-tuote, joka on saatavissa `netlib`istä.**
- **CSC:ssä asennettu versio 3.1 Craylle, Convexille, Finsunille ja Azalealle sekä Iris-työasemille.**

Al Geist

Weicheng Jiang

Vaidy Sunderam

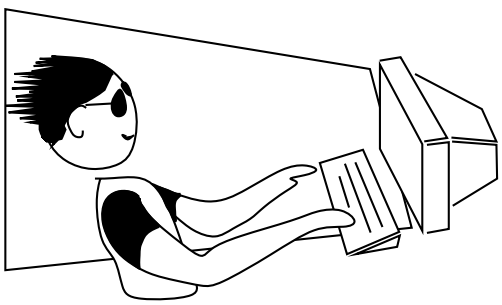
Keith Moore

Jack Dongarra

Adam Beguelin

Carolyn Aebischer

Robert Manchek



Team PVM

Vendor Support

Several vendors have announced that they will supply and support PVM on their systems:

Cray Research Inc.

Convex

DEC

IBM

Silicon Graphics

Intel

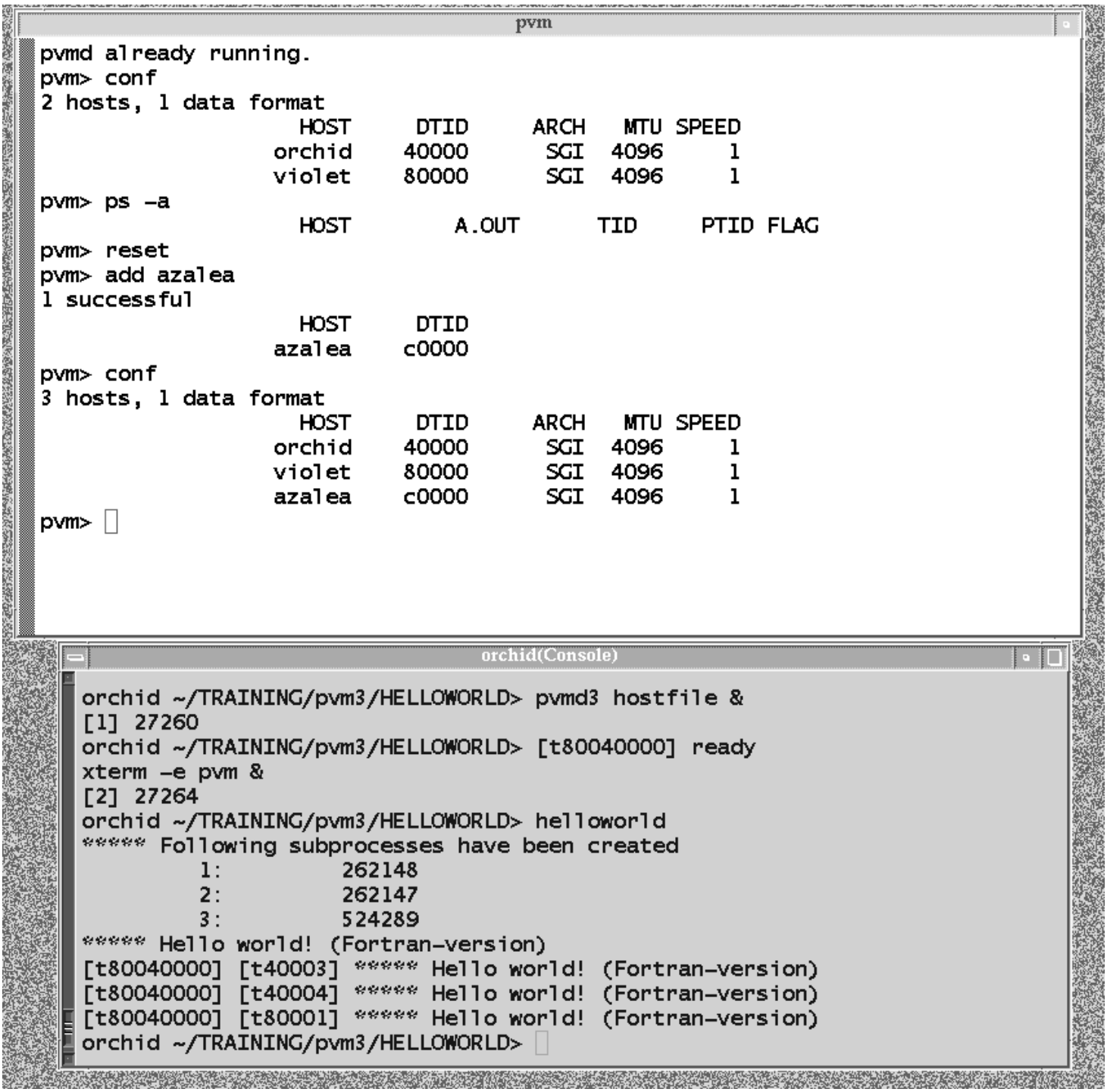
Thinking Machines Corp.

Miten PVM toimii ?

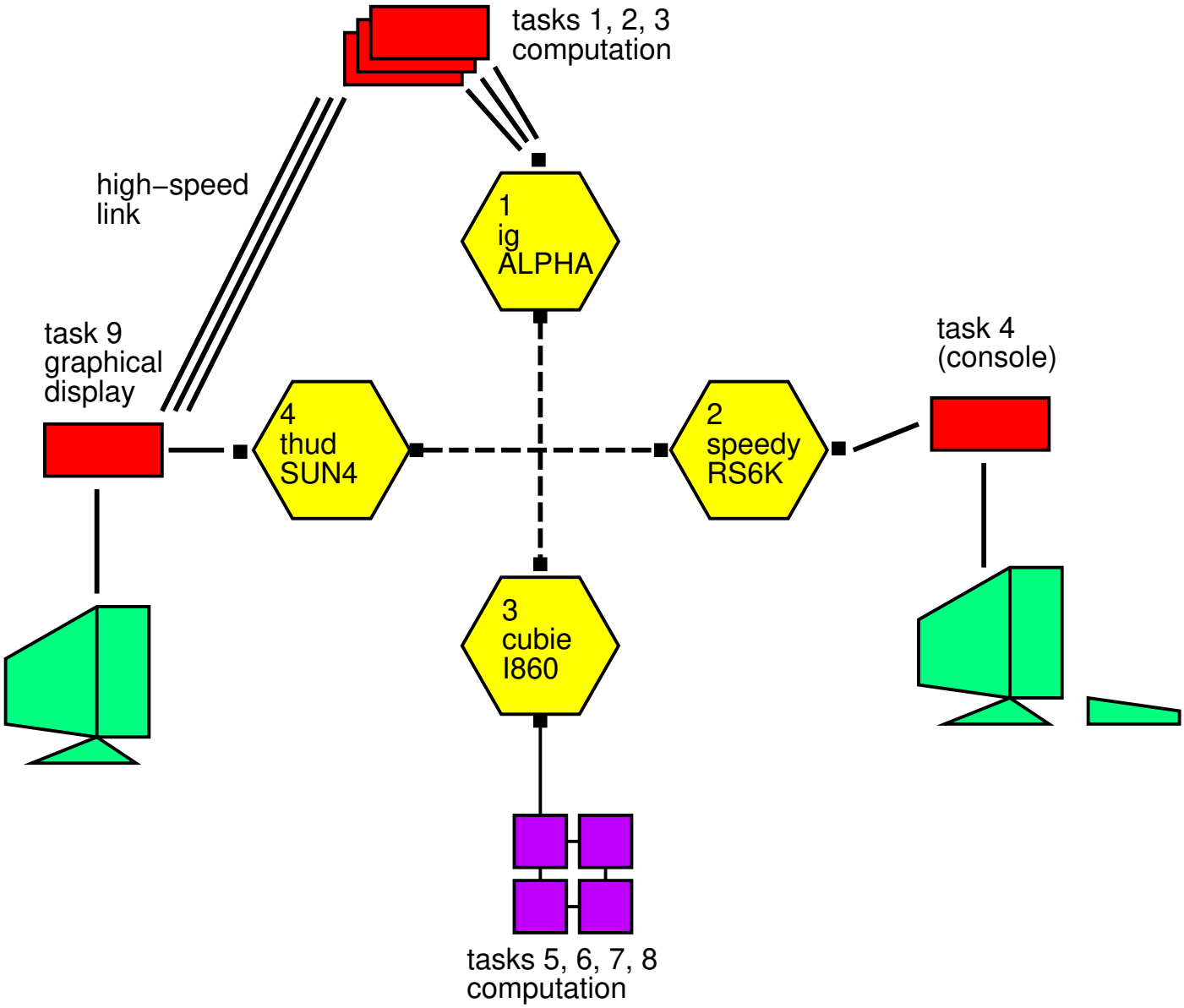
- **PVM-laskenta koostuu kahdesta peruskomponentista, jotka yhdessä muodostavat "kuvitellusti rinnakkaisen tietokoneen". Komponentit ovat:**

  1. **Käyttäjän C- tai Fortran-ohjelma(t), joihin on lisätty PVM-aliohjelmakutsuja.**
  2. **Kullakin koneella tausta-ajona pyörivistä ns. PVM-demoniohjelmista, jolla kontrolloidaan prosessien tekemää tiedonsiirtoa eri koneiden välillä ja hallitaan PVM-kompleksia.**

- **Käyttäjän PVM-ohjelmat (yksi tai useampia) kommunikoivat keskenään käyttäen ns. viestinvälitysmekanismia (message passing).**
- **Yhteydet PVM-ohjelmaosien välillä ovat suoria (yleensä `TCP/IP`-linkki) tai epäsuoria, PVM-demonien välityksellä tapahtuvia (`UDP`-linkki).**
- **Perustapauksissa käyttäjän ei tarvitse välittää viestintätavasta.**
- **PVM-demoniohjelmia voi kontrolloida PVM-konsoli-ikkunan (PVM-monitori) avulla.**


pvm
pvmd already running.
pvm> conf
2 hosts, 1 data format
HOST DTID ARCH MTU SPEED
orchid 40000 SGI 4096 1
violet 80000 SGI 4096 1
pvm> ps -a
HOST A.OUT TID PTID FLAG
pvm> reset
pvm> add azalea
1 successful
HOST DTID
azalea c0000
pvm> conf
3 hosts, 1 data format
HOST DTID ARCH MTU SPEED
orchid 40000 SGI 4096 1
violet 80000 SGI 4096 1
azalea c0000 SGI 4096 1
pvm>
orchid(Console)
orchid ~/TRAINING/pvm3/HELLOWORLD> pvmd3 hostfile &
[1] 27260
orchid ~/TRAINING/pvm3/HELLOWORLD> [t80040000] ready
xterm -e pvm &
[2] 27264
orchid ~/TRAINING/pvm3/HELLOWORLD> helloworld
***** Following subprocesses have been created
1: 262148
2: 262147
3: 524289
***** Hello world! (Fortran-version)
[t80040000] [t40003] ***** Hello world! (Fortran-version)
[t80040000] [t40004] ***** Hello world! (Fortran-version)
[t80040000] [t80001] ***** Hello world! (Fortran-version)
orchid ~/TRAINING/pvm3/HELLOWORLD>

Version 3 Details



10 vaihetta PVM-laskennassa

Vaihe 1: **Tehdään PVM-aliohjelmakutsuja sisältävät C- tai Fortran-ohjelmat.**

Vaihe 2: **Luodaan niistä ajoversiot kullekin niistä konearkkitehtuureista, joissa PVM-ohjelmia tai -ohjelmanosia aiotaan käyttää. Ohjelmat käännetään ja linkataan seuraavasti (*Silicon Graphics*):**

```
f77 prog.f -lfpvm3 -lpvm3 -lsun -o prog.exe  # Fortran-77
cc -I/usr/local/include prog.c -lpvm3 -lsun -o prog.exe # C
```

Vaihe 3: **Viedään ajoversiot hakemistoihin, joista ne aiotaan seuraavaksi käynnistää.**

Vaihe 4: **Luodaan "`hostfile`", jossa määritellään ne koneet, joiden halutaan aktiivisesti/passiivisesti osallistuvan PVM-kompleksiin. Esimerkki "`hostfile`"-tiedostosta:**

```
* dx=/usr/local/bin/pvmd3
azalea
finsun ep=bin/SUN4:/tmp/username/bin
& convex.csc.fi  ep=bin/CNVX
```

Vaihe 5: **Käynnistetään PVM-demonien taustaprosessit komennolla**

```
% pvmd3 hostfile &
```

Vaihe 6: **Käynnistetään PVM-konsoli-ikkuna (PVM-monitori) erilliseksi X-ikkunaksi:**

```
% xterm -e pvm &
```

Vaihe 7: **Annetaan PVM-konsoli-ikkunassa komennot** `conf`, `ps -a` **sekä** `reset`.
Lisätietoja muista komennoista saa komennolla `help`.

Vaihe 8: **Käynnistetään PVM-ohjelma, joka puolestaan käynnistää muut PVM-ohjelmanosat toisissa koneissa.**

Vaihe 9: **Jos halutaan ajaa jokin toinen PVM-sovellus, toistetaan (tarvittaessa) vaiheet** $1 \ldots 3$ **sekä jatketaan vaiheesta** 7.

Vaihe 10: **Kun PVM-sovellus on päättynyt ja halutaan lopettaa PVM-istunto, annetaan PVM-konsoli-ikkunassa komento** `halt`.

PVM:n `hostfile`-tiedosto

Tiedostolla kerrotaan PVM-demoneille:

- **mitkä koneet aikovat osallistua PVM-laskentaan,**
- **mistä löytyvät ajokelpoiset ohjelmat kullakin koneella (`ep=` parametri UNIX-polulla),**
- **mitä PVM-demonia käytetään (`dx=` parametri),**
- **mitä käyttäjätunnusta käytetään (`lo=` parametri),**
- **ja annetaanko PVM-demonia käynnistettäessä salasana (`pw=` parametri; esimerkiksi `$HOME/.rhosts`-tiedoston puuttuessa).**

Erikoismerkityksensä on seuraavilla merkeillä `hostfilessa`:

- **#-merkillä alkavat rivit ovat kommenttirivejä.**
- **Tähdellä (*) alkavat rivit muuttavat `hostfile`-parametrien oletusasetuksia.**
- **&-merkillä alkavat rivit kertovat PVM:lle myöhemmin laskentaan osallistuvien tietokoneiden nimet (passiiviset koneet).**

PVM-hostfile esimerkki:

```
# Tämä on vain kommenttirivi

#
# Pari CSC:n konetta
#
* dx=/usr/local/bin/pvmd ep=/tmp/sbs
finsun
convex.csc.fi

#
# TKK:n DEC AXP gamma.hut.fi sotkemaan kuvioita...
#
gamma.hut.fi ep=/tmp/sbs

#
# Pari Tampereen TKK:n konetta soppaan mukaan
#
* dx=$HOME/bin/ALPHA/pvmd3 ep=bin/ALPHA:bin:bin/misc
alpha.cc.tut.fi lo=kankaala pw
beta.cc.tut.fi lo=sbs dx=/alpha/bin/pvmd3

#
# Passiivinen kone: myöhemmin (ehkä) mukaan ?
#
* dx=/usr/local/bin/pvmd
& azalea.csc.fi ep=/tmp/sbs
```

PVM-aliohjelmakirjasto

PVM:n aliohjelmakirjastokutsuilla hoidetaan PVM-prosessien välistä tiedonsiirtoa ja synkronointia sekä hallitaan PVM-istuntoa omasta ohjelmasta käsin.

Aliohjelmakirjasto on kirjoitettu C-kielellä, mutta käyttäjäliittymät on tehty C- ja Fortran-kieleen.
Myös C++:n käyttö on mahdollista `pvm3.h`:ssa olevien C-prototyyppien ansiosta, mutta `streams`-tyyppisten operaatioiden kohdistaminen tietoalkioihin ei ole (vielä) mahdollista.

C- ja Fortran-kieliset aliohjelmatkutsut ovat PVM:n versiossa 3.x seuraavanlaisia:

- `info = pvm_XXX(...)`**-funktiot C-ohjelmille.**
- `call pvmfXXX(...,info)`**-subroutinet Fortran-ohjelmille.**

Käytäntö on sellainen, että C-ohjelmat palauttavat kutsun onnistumisen paluukoodina (`info`) suoraan, mutta Fortran-ohjelmissa paluukoodi on aina aliohjelman viimeisenä parametrinä.

Esimerkki PVM-ohjelmasta (Fortran)

```
      program helloworld
      implicit none
c-- Seuraava löytyy CSC:ssä hakemistosta: /usr/local/include
      include 'fpvm3.h'

      integer NPROC
      parameter (NPROC = 4)
      integer taskid(0:NPROC-1)
      integer my_taskid
      integer my_parent
      integer info,i,tid

      call pvmfmytid(my_taskid)
      if (my_taskid .lt. 0) then
         call pvmfperror('pvmfmytid()@main()')
         stop 'ERROR'
      endif

      call pvmfparent(my_parent)
      if (my_parent .lt. 0) then

         call pvmfspawn(
     $        'helloworld',
     $        PVMDEFAULT,
     $        '*',
     $       NPROC-1,
     $        taskid(1),
     $        info)
```

```
        taskid(0) = my_taskid
        print *,
     $       'Following subprocesses have been created by tid=',
     $       taskid(0)
        do 100 i=1,NPROC-1
           print *, i,':',taskid(i)
100     continue
     endif

     if (my_parent .lt. 0) then
        do 200 i=1,NPROC-1
           call pvmfrecv(-1,999,info)
           call pvmfunpack(INTEGER4,tid,1,1,info)
           print *,'Hello world from tid=',tid
200     continue
     else
        call pvmfinitsend(PVMDEFAULT,info)
        call pvmfpack(INTEGER4,my_taskid,1,1,info)
        call pvmfsend(my_parent,999,info)
     endif

     call pvmfexit(info)
     stop 'SUCCESS'
     end
```

Esimerkkiajo:

```
%  pvmd3 hostfile &
%  xterm -e pvm &
%  helloworld
 Following subprocesses have been created by tid=      262156
            1:      262157
            2:      524293
            3:      786435
 Hello world from tid=      262157
 Hello world from tid=      786435
 Hello world from tid=      524293
STOP SUCCESS statement executed
%  pvm
pvmd already running.
pvm>  halt
Killed
```

Esimerkki PVM-ohjelmasta (C)

```
#include <stdio.h>
#include <string.h>
#include "pvm3.h"

#define NPROC 4
main()
{
  register int i;
  int my_taskid;
  int my_parent;

  my_taskid = pvm_mytid();
  if (my_taskid < 0) {
    pvm_perror("pvm_mytid()@main()");
    exit(1);
  }

  my_parent  = pvm_parent();
  if (my_parent < 0) { /* MASTER process */
    int taskid[NPROC];

    pvm_spawn("helloworld",
      NULL,
      PvmTaskDefault,
      "",
      NPROC-1,
      &taskid[1]);
```

```
    taskid[0] = my_taskid;
    printf("Following subprocesses have been created by tid=%d:\n",
   taskid[0]);
    for (i=1; i<NPROC; i++) printf("%4d:[%d]\n",i,taskid[i]);
  }

  if (my_parent < 0) { /* MASTER process */
    for (i=1; i<NPROC; i++) {
      int tid;
      pvm_recv(-1,999);
      pvm_upkint(&tid,1,1);
      printf("Hello world from tid=%d\n",tid);
    }
  }
  else { /* SLAVE processes */
    pvm_initsend(PvmDataDefault);
    pvm_pkint(&my_taskid,1,1);
    pvm_send(my_parent,999);
  }

  pvm_exit();
  exit(0);
}
```

pvm_mytid()

- Ilmoittaudutaan PVM-prosessiksi.
- Yleensä pääohjelman alkuun.
- C: `my_taskid = pvm_mytid();`
- Fortran: `call pvmfmytid(my_taskid)`
- Negatiivinen paluukoodi ilmoittaa virheen.

pvm_parent()

- Selvitetään isäntäprosessin nimi.
- Voidaan hyödyntää mm. SPMD-ohjelmissa, joissa on yksi ohjelma, mutta eri data-alueet. (Isäntä ja lapsiprosessit ajavat samaa ohjelmakoodia).
- C: `my_parent = pvm_parent();`
- Fortran: `call pvmfparent(my_parent)`
- Negatiivinen paluukoodi kertoo, että kyseessä on PVM-demonin aloittama (isäntä)prosessi.

pvm_spawn()

- Luo yhden tai useampia PVM-prosesseja yhdellä kertaa.
- Prosesseja voidaan luoda määrätylle koneelle:

```
C: pvm_spawn("a.out",NULL,PvmTaskHost,"alpha.hut.fi",1,&tid[0]);
Fortran:
   call pvmfspawn('a.out',PVMHOST,'alpha.hut.fi',1,itid,info)
```

  tai määrätylle arkkitehtuurille:

```
C: pvm_spawn("node",NULL,PvmTaskArch,"SGI",3,&tid[0]);
Fortran:
   call pvmfspawn('node',PVMARCH,'SGI',3,itid(1),info)
```

  tai PVM-demonin vapaasti valitsemille koneille (siinä järjestyksessä, kun `hostfile` määrää):

```
C: pvm_spawn("slave",NULL,PvmTaskDefault,"",16,&tid[0]);
Fortran:
   call pvmfspawn('slave',PVMDEFAULT,'*',16,itid(1),info)
```

- Paluukoodi kertoo luotujen prosessien todellisen lukumäärän.
- C-kielisissä ohjelmissa voidaan välittää myös parametrilista.

pvm_exit()

- Poistutaan PVM-istunnosta.
- *Ei lopeta* ohjelman suoritusta.
- C: `info = pvm_exit();`
- Fortran: `call pvmfexit(info)`

Message Passing PVM:llä

- Käytetään tiedon siirtämiseksi prosessien (tai prosessiryhmien) välillä.
- Useita eri tietotyyppejä kerrallaan.
- Automaattinen tiedonkonversio (XDR), mikäli käytetään erilaisia lukujen esitystapoja PVM-koneessa.
- Lähettäminen:

C	Fortran
`pvm_initsend(PvmDataDefault)`	`call pvmfinitsend(PVMDEFAULT)`
`pvm_pk*(...);`	`call pvmfpack(...)`
`pvm_send(tid,msgtag);`	`call pvmfsend(itid,msgtag,info)`

- Vastaanotto:

C	Fortran
`pvm_recv(tid,msgtag)`	`call pvmfrecv(itid,msgtag,info)`
`pvm_upk*(...);`	`call pvmfunpack(...)`

- Monta yhdellä kertaa: `pvm_mcast()` / `pvm_recv(-1,tag)`
- non-Blocking Receive: `pvm_nrecv()`
- Prosessiryhmille lisäksi muita aliohjelmia.

pvm_pk*() / pvm_upk*()

- Tiedonpakkaus/-purkurutiinit.
- Seuraavat tietotyypit tuettuja (tiedonpakkaus):

C	Fortran
`pvm_pkbyte(char *cp,int cnt,int strd)`	`call pvmfpack(BYTE1,cp,cnt,strd,info)`
`pvm_pkcplx(float *zp,int cnt,int strd)`	`call pvmfpack(COMPLEX8,zp,cnt,strd,info)`
`pvm_pkdcplx(double *zp,int cnt,int strd)`	`call pvmfpack(COMPLEX16,zp,cnt,strd,info)`
`pvm_pkdouble(double *dp,int cnt,int strd)`	`call pvmfpack(REAL8,dp,cnt,strd,info)`
`pvm_pkfloat(float *fp,int cnt, int strd)`	`call pvmfpack(REAL4,fp,cnt,strd,info)`
`pvm_pkint(int *ip,int cnt, int strd)`	`call pvmfpack(INTEGER4,ip,cnt,strd,info)`
`pvm_pklong(long *ip,int cnt, int strd)`	`call pvmfpack(INTEGER4,ip,cnt,strd,info)`
`pvm_pkshort(short *ip,int cnt, int strd)`	`call pvmfpack(INTEGER2,ip,cnt,strd,info)`
`pvm_pkstr(char *sp)`	`call pvmfpack(STRING,sp,cnt,strd,info)`

- Tiedonpurku: `pvm_pk*() => pvm_upk*()` C:ssä ja `pvmfpack() => pvmfunpack()` Fortranilla.
- Stride-parametrilla (`strd`) voidaan valita joka N:s alkio.
- Struktuurit täytyy koodata erikseen.

Jos tulee ongelmia ...

Seuraavat tilanteet ovat "tavanomaisia", kun aloittelee PVM:llä:

- **Lähetettävän ja vastaanotettavan viestin tietomäärä, viestitunnukset (message id) tai kohteet (task id) eivät vastaa toisiaan lähettävässä ja vastaanottavassa päässä.**
  Ratkaisu: **Tarkista lähetys- ja vastaanottokomentojen vastaavuus ohjelmassasi.**

- **Tiedostoa `fpvm3.h` (`pvm3.h`) ei löydy, kun käännetään PVM-sovellusta.**
  Ratkaisu: **Tee seuraava symbolinen linkki hakemistoon, jossa ohjelmasi majailevat:**

  ```
  % ln -s /usr/local/include/fpvm3.h .
  ```

- **Jos PVM-demoni (`pvmd3`) ei käynnisty ja outo virheilmoitus tulee näytölle, niin voi syitä olla monia. Kokeile seuraavia:**

  1. **Anna komento `pvm` ja sen alla komento `halt`, jolloin aktiiviset PVM-demonit kuolevat.**
  2. **Yritä tuhota tiedosto `/tmp/pvml.<uid>`, missä `<uid>` on käyttäjätunnuksesi numero (kts. UNIX-komento `id`). Tiedosto on voinut jäädä edellisestä PVM-demonin (epäonnistuneesta) ajamisesta.**

3. **Tarkista, että kaikkien PVM-laskentaan osallistuvien koneiden tiedostoissa** `$HOME/.cshrc` **ei tulosteta mitään. Siirrä kaikki tulostaminen tiedostoon** `$HOME/.login` **tai "eristä" tulostaminen** `$HOME/.cshrc`**:ssä seuraavasti:**

```
if ( $?prompt ) then
   echo "Terminal type is $TERM"
   stty erase ^? intr ^C
   uptime
endif
```

Lisäinformaatiota saatavilla

- **Lähdekielinen koodi:**
  Postiosoite: `netlib@nac.no`
  Subjektikenttään: `send index from pvm3`
- **Tai suoraan FTP:llä:** `research.att.com (/netlib/pvm3)`.
  Käytä `binary transfer`**-moodia!**
- **CSC:ssä kaikilta koneilta PVM:n User Guide:**
  `/usr/local/bin/pvm/doc/ug_v3.1.ps.Z`
- **Usenet News:** `comp.parallel.pvm`
- **PVM-ryhmän postiosoite:** `pvm@msr.epm.ornl.gov`