Einführung in PVM (Teil 2)

Übungen zur Vorlesung
„Parallele und verteilte Algorithmen“

Philipps-Universität Marburg
Sommersemester 2002

Prozessgruppen

- dynamisches Zusammenfassen von Prozessen zu Gruppen
  - inum pvm_joingroup(„group_name“)
    - erster Aufruf erzeugt neue Gruppe
    - Rückgabewert ist eine eindeutige Gruppenid.
    - Zugehörigkeit zu mehreren Gruppen ist möglich
  - info pvm_lvgroup(„group_name“)
    - Verlassen einer Gruppe

Achtung:
Um diese Routinen verwenden zu können, muss eine separate Bibliothek libgpvm3.a zum Programm gebunden werden.

- Abfrageroutinen
  - inum pvm_getinst(*group,tid) bestimmt Gruppenid zu Gruppe und Taskid
  - tid pvm_gettid(*group,inum) bestimmt Taskid zu Gruppe und Gruppenid
  - size pvm_gsize(*group) bestimmt Anzahl der Prozesse in einer Gruppe

Beispiel

```
{       int mytid, mygid1, mygid2, mygid3;

        mytid = pvm_mytid();

        mygid1 = pvm_joingroup("prognosis");
        mygid2 = pvm_joingroup("negative");

        printf("joined prognosis %d, negative %d\n", mygid1, mygid2);

        mygid1 = pvm_lvgroup("prognosis");
        mygid2 = pvm_lvgroup("negative");

        printf("left prognosis %d, negative %d\n", mygid1, mygid2);
        pvm_exit();
}
```

Gruppenoperationen

- Broadcast
  - info pvm_bcast(*group,msgtag)
- Synchronisation
  - int pvm_barrier(*group,count)
- Reduktion
  - info pvm_reduce(*op(),*data,nitem,datatype,msgtag,group, root)
    - vordefinierte Reduktionsoperationen:
      PvmMax, PvmMin, PvmSum, PvmProduct
- Verteilen und Sammeln von Arrays
  - info pvm_scatter(*result,*data,count,datatype,msgtag,group,root)
  - info pvm_gather(*result,*data,count,datatype,msgtag,group,root)

Beispiel: Auszüge aus ge.c

```
{   int mytid, mygid, ctid[32];
    int nproc, i, indx;
    int cc;

    mytid = pvm_mytid();
    nproc = atoi(argv[1]))
    /* join a group */
    mygid = pvm_joingroup("ge");
    ...
    /* if I'm the first to join then start the others */
    if (mygid == 0) {
        /* start a bunch of children */
        pvm_spawn(argv[0], (char**) 0, 0, "",
                                    nproc-1, ctid));
        ...
        /* tell them how many sibs */
        pvm_initsend(PvmDataDefault);
        pvm_pkint(&nproc, 1, 1);
        pvm_mcast(ctid, nproc-1, 15);
        }
```

```
else {
        /* find out the number of sibs */
        pvm_recv(pvm_parent(), 15);
        pvm_upkint(&nproc, 1, 1);
        fprintf(stderr,"nproc %d\n",nproc);
        }

/* sync on a barrier */
pvm_barrier("ge", nproc)
fprintf(stderr,
        "group %s size %d gid %d: sync\n",
         "ge", pvm_gsize("ge"), mygid);

/* everyone broadcast their gids and tids */
pvm_initsend(PvmDataDefault);
pvm_pkint(&mygid, 1, 1);
pvm_pkint(&mytid, 1, 1);
pvm_bcast("ge", 63);
```

Beispiel: ge.c (Fortsetzung)

```
    /* recv all the gids and
                tids (except from myself) */
    for (i = 0; i < nproc-1; i++) {
                pvm_recv(-1, 63);
                pvm_upkint(&indx, 1, 1);
                pvm_upkint(ctid+indx, 1, 1);
                }
    /* set my tid too */
    ctid[mygid] = mytid;
    /* check to make sure the gids and tids are correct */
    for (i = 0; i < nproc; i++) {
                if (i != pvm_getinst("ge", ctid[i]))
                    { fprintf(stderr, "gid %d doesn't
                            match up!\n", i); }
                if (ctid[i] != pvm_gettid("ge", i))
                    {fprintf(stderr, "gid %d doesn't
                            match up!\n", i); }
                }
```

```
    /* leave the group */
    pvm_barrier("ge",nproc);
    pvm_lvgroup("ge")
    pvm_exit();
    printf("done\n");
    return 0;
}
```

Ohne diese Barriere ist nicht gewährleistet, dass alle Prozesse noch in der Gruppe sind, wenn die Tids und Gids abgeglichen werden.

Beispiel: Teste reduce – scatter – gather: trsg.c

```
main()
{
 int myginst, i, j, gsize, count, nprocs, msgtag, datatype, info_product, info_user;
 int  tids[MAXNPROCS], myrow[MAXNDATA], matrix[MAXNDATA*MAXNPROCS];
 ...
 char *gname = "group_rsg";

 /* join the group */
 myginst = pvm_joingroup(gname);
 ...
 /* I am the first group member, get input, start up copies of myself */
 if ( myginst == 0 )
 {   ...
     tids[0] = pvm_mytid();
     if (nprocs > 1)
              pvm_spawn(„trsg“, (char**)0, 0, "", nprocs-1, &tids[1]);

     /* wait until they have all started and joined, then send input values */
     while (gsize = pvm_gsize(gname) < nprocs)
              pvmsleep(1);
```

Beispiel-Fortsetzung: trsg.c

```
     /* send input values */
     pvm_initsend(PvmDataDefault);  pvm_pkint(&nprocs, 1, 1);  pvm_pkint(&count,  1, 1);
     pvm_bcast(gname, msgtag=17);  }
 else
 {   /* receive the input values */
     pvm_recv(-1, msgtag=17);      pvm_upkint(&nprocs, 1, 1);    pvm_upkint(&count,  1, 1); }

 rootginst = 0;  /* determine the group root */

 /* init the matrix values on the root processor */
 if (myginst == rootginst)  ...

 /* scatter rows of matrix to each processor */
 pvm_scatter(myrow, matrix, count, PVM_INT, msgtag=19, gname, rootginst);

 /* local computations  */   PSum = ...

 /* gather partial sums to the rootginst */
 pvm_gather(PartSums, &PSum, 1, PVM_INT, msgtag=21, gname, rootginst);

 /* do a global sum over myrow, the result goes to rootginst */
 pvm_reduce(PvmSum, myrow, count, PVM_INT, msgtag=23, gname, rootginst);
 ...
```