Openmosix e Beowulf: Esempi di programmazione distribuita

Giovanni Perbellini

Agenda

- Openmosix
  - Esercizio “stress”
  - Esercizio “fork”
- Beowulf (PVM)
  - Setup ambiente
  - Compilazione programmi PVM
  - Esercizio Helloworld
  - Esercizio Master-Slave

Openmosix – Esempio 1 (I)

- Script che lancia 10 processi indipendenti

```
#!/bin/bash
#lancia 10 processi awk indipendenti
for n in 0 1 2 3 4 5 6 7 8 9 ; do
awk 'BEGIN { for (i=0;i<10000;i++)
                for (j=0;j<10000;j++);}' &
done
```

- Comando awk
  - Awk suddivide ogni riga dell'input che gli è stato passato in campi
  - BEGIN: inzializzazione del comando awk

Openmosix – Esempio 1 (II)

- Permessi di esecuzione dello script
  - **`chmod +x test-om.sh`**
- Esecuzione dello script
  - **`./test-om.sh &`**
- Verifica del carico di lavoro
  - **`mosmon, mtop`**

Openmosix – Esempio 1 (III)

- Il programma **`mosmon`** permette di vedere i 10 processi migrare tra i nodi del cluster
- Inizialmente il nodo che ha lanciato i jobs si caricherà fino a saturare la cpu
- Dopo un periodo di tempo, i jobs migreranno sugli altri nodi, per essere eseguiti su quei nodi

Openmosix – Esempio 2 (I)

```
…
…
#define PI 3.14159
#define CNT 20000000
int main() {
    int child, i, j;
    float *ret;
    int processes=6;
    printf("Starting %d forkit instances\n", processes);
    for(i=0; i<processes; i++) {
          if ((ret=malloc(CNT*sizeof(float)))==NULL) printf("NO MEM!\n");
          if ((child = fork()) == -1) perror("fork");
          else if (!child) {
                    printf ("\tPROC %i started...\n",i);
                    for(j=0;j<CNT;j++)
                              ret[j]=sin(PI/2)/cos(0)*sin(PI/2);
                    printf ("\t...PROC %i finished!\n",i);
                    exit(0);
          }
          free(ret);
    }
    printf ("...main application finished!\n",i);
    return 0;
}
```

Crea un processo figlio

Openmosix – Esempio 2 (II)

- Programma C che crea 6 processi (utilizzo del costrutto `fork()`)
- Analisi della migrazione dei processi con `mosmon`
- Compilazione
  - `gcc -o openmosix_test openmosix.c`

PVM – Setup ambiente (I)

- --$HOME
  - pvm3
    - bin
      - LINUX
        - *executable files*
    - examples
      - *examples files*
    - ecc.

PVM – Setup ambiente (II)

- Impostare la variabile PVM_ROOT
  - **`export PVM_ROOT = $HOME/pvm3`**
  - **`Impostare la variabile nel file .bashrc`** **(sul cluster ESD)**
- Copiare l'eseguibile (per ogni architettura) nella directory
  - **`$HOME/pvm3/bin/$PVM_ARCH`**
  - (per esempio: **`PVM_ARCH=LINUX`**)

PVM – compilazione programmi

- Modalità con Makefile
  - **`% aimk program`**
    (Makefile.aimk)
- Modalità senza Makefile
- Compilazione utilizzando `gcc` e la libreria PVM
  - **`% gcc -o program program.c -lpvm3`**
  - **`% cp program $PVM_ROOT/bin/$PVM_ARCH`**

Esempio 1 – Helloworld (I)

```
// hello.c
#include <stdio.h>
#include "pvm3.h"

main() {
        int cc, tid;
        char buf[100];
        printf("HELLO: I'm t%x\n", pvm_mytid());
        cc =pvm_spawn("hello_other",(char**)0,0,"",1,&tid);
        if (cc == 1) {
                cc = pvm_recv(-1, -1);
                pvm_bufinfo(cc, (int*)0, (int*)0, &tid);
                pvm_upkstr(buf);
                printf("from t%x : %s\n", tid, buf);
        } else
                printf("can't start hello_other\n");
        printf("HELLO::exit\n");
        pvm_exit();
        exit(0);
}
```

Starts new PVM processes (l'eseguibile `hello_other` viene cercato in $HOME/pvm3/bin/$PVM_ARCH/)

Receive a message (tid=-1: will accept a message from any process)

Returns information about a message buffer

Unpack the active message buffer into arrays of prescribed data type

Esempio 1 – Helloworld (II)

- `int numt = pvm_spawn(char *task, char **argv, int flag, char *where, int ntask, int *tids)`
  - **`Task`**: nome del file eseguibile
  - **`Argv`**: puntatore all'array degli argomenti dell'eseguibile
  - **`Flag`**: opzioni di spawn (0=PVM può scegliere ogni macchina per far partire il task)
  - **`Where`**: specifica dove iniziare i processi PVM
  - **`Ntask`**: numero di processi figli
  - **`Tids`**: array con l'ID dei processi figli generati
  - **`numt`**
    - numero di processi figli generati
- `int bufid = pvm_recv( int tid, int msgtag )`
  - **`Tid`**: ricezione messaggi dal processo mittente `tid`
  - **`Msgtag`**: ricezione messaggi dal processo mittente `msgtag`
  - **`Bufid`**
    - Identificatore del messahggi oricevuto

Esempio 1 – Helloworld (III)

```
// hello_other.c
#include "pvm3.h"

main() {
        int ptid;
        char buf[100];

        ptid = pvm_parent();

        strcpy(buf, "Hello world from ");
        gethostname(buf + strlen(buf), 64);

        pvm_initsend(PvmDataDefault);
        pvm_pkstr(buf);
        pvm_send(ptid, 1);

        printf("HELLO_OTHER::exit\n");
        pvm_exit();
        exit(0);
}
```

Returns the tid of the process that spawned the calling process

Clear default send buffer and specify message encoding

pvm_pack - Pack the active message buffer with arrays of prescribed data type

Immediately sends the data in the active message buffer

Esempio 1 – Helloworld (IV)

- Creazione di un nuovo task e passaggio dei messaggi tra task
- Compilazione 1:
  - `% aimk hello hello_other`
- Compilazione 2:
  - `% gcc -o hello hello.c -lpvm3`
  - `% mv hello $PVM_ROOT/bin/$PVM_ARCH`
  - `% gcc -o hello_other hello_other.c -lpvm3`
  - `% mv hello_other $PVM_ROOT/bin/$PVM_ARCH`
- Esecuzione 1:
  - `% pvm`
  - `pvm> spawn -> hello`
- Esecuzione 2:
  - `% pvm`
  - `pvm> quit`
  - `% ./hello`

Esempio 2 – Master - Slave (I)

Returns information about the present virtual machine configuration

Set number of slaves to start

start up slave tasks

```
#include <stdio.h>
#include "pvm3.h"
#define SLAVENAME "slave1"
main(){
        int mytid;                      /* my task id */
        int tids[32];                   /* slave task ids */
        int n, nproc, numt, i, who, msgtype, nhost, narch;
        float data[100], result[32];
        struct pvmhostinfo *hostp;
        mytid = pvm_mytid();            /* enroll in pvm */
        pvm_config( &nhost, &narch, &hostp );
        nproc = nhost * 3;
        if( nproc > 32 ) nproc = 32 ;
        printf("Spawning %d worker tasks ... " ,nproc);
        numt=pvm_spawn(SLAVENAME, (char**)0, 0,"",nproc,tids);
        if( numt < nproc ){
            printf("\n Trouble spawning slaves.Error codes:\n");
            for( i=numt ; i<nproc ; i++ ) {
                  printf("TID %d %d\n",i,tids[i]);
            }
            for( i=0 ; i<numt ; i++ ){
                  pvm_kill( tids[i] );
            }
            pvm_exit();
            exit(1);
        }
        printf("SUCCESSFUL\n");
…
```

Esempio 2 – Master - Slave (II)

Broadcast initial data to slave tasks

Multicasts the data in the active message buffer to a set of tasks

Wait for results from slaves

```
…
n = 100;
/* initialize_data( data, n ); */
for( i=0 ; i<n ; i++ ){
  data[i] = 1.0;
}
pvm_initsend(PvmDataDefault);
pvm_pkint(&nproc, 1, 1);
pvm_pkint(tids, nproc, 1);
pvm_pkint(&n, 1, 1);
pvm_pkfloat(data, n, 1);
pvm_mcast(tids, nproc, 0);
msgtype = 5;
for( i=0 ; i<nproc ; i++ ){
  pvm_recv( -1, msgtype );
  pvm_upkint( &who, 1, 1 );
  pvm_upkfloat( &result[who], 1, 1 );
  printf("I got %f from %d; ",result[who],who);
  if (who == 0)
     printf("(expecting %f)\n",(nproc - 1) * 100.0);
  else
     printf("(expecting %f)\n",(2 * who-1) * 100.0);
}
/* Program Finished exit PVM before stopping */
pvm_exit();
}
```

Esempio 2 – Master – Slave (III)

Receive data from master

Determine which slave I am (0 -- nproc-1)

Do calculations with data

Send result to master

```
#include <stdio.h>
#include "pvm3.h"
main(){
   int mytid;       /* my task id */
   int tids[32];    /* task ids   */
   int n, me, i, nproc, master, msgtype;
   float data[100], result;
   float work();    /* enroll in pvm */
   mytid = pvm_mytid();
   msgtype = 0;
   pvm_recv( -1, msgtype );
   pvm_upkint(&nproc, 1, 1);
   pvm_upkint(tids, nproc, 1);
   pvm_upkint(&n, 1, 1);
   pvm_upkfloat(data, n, 1);
   for( i=0; i<nproc ; i++ )
       if( mytid == tids[i] ){ me = i; break; }
   result = work( me, n, data, tids, nproc );
   pvm_initsend( PvmDataDefault );
   pvm_pkint( &me, 1, 1 );
   pvm_pkfloat( &result, 1, 1 );
   msgtype = 5;
   master = pvm_parent();
   pvm_send( master, msgtype );
   pvm_exit(); /* Program finished.Exit PVM */
}
```

Esempio 2 – Master – Slave (IV)

```
/* Simple example: slaves exchange data with left neighbor (wrapping) */
float work(me, n, data, tids, nproc)
int me, n, *tids, nproc; float *data;
{
   int i, dest;
   float psum = 0.0;
   float sum = 0.0;
   for( i=0 ; i<n ; i++ ){
      sum += me * data[i];
   }    /* illustrate node-to-node communication */
   pvm_initsend( PvmDataDefault );
   pvm_pkfloat( &sum, 1, 1 );
   dest = me+1;
   if( dest == nproc ) dest = 0;
   pvm_send( tids[dest], 22 );
   pvm_recv( -1, 22 );
   pvm_upkfloat( &psum, 1, 1 );
   return( sum+psum );
}
```

Esempio 2 – Master – Slave (V)

- Creazione di un nuovo task e passaggio dei messaggi tra task
- Compilazione 1:
  - `% aimk master1 slave1`
- Compilazione 2:
  - `% gcc -o master1 master1.c -lpvm3`
  - `% mv master1 $PVM_ROOT/bin/$PVM_ARCH`
  - `% gcc -o slave1 slave1.c -lpvm3`
  - `% mv slave1 $PVM_ROOT/bin/$PVM_ARCH`
- Esecuzione 1:
  - `% pvm`
  - `pvm> spawn -> master1`
- Esecuzione 2:
  - `% pvm`
  - `pvm> quit`
  - `% ./master1`