PHILIPPS-UNIVERSITÄT MARBURG
Fachbereich Mathematik und Informatik
Prof. Dr. R. Loogen
Dipl.Inf. J.Berthold

D-35032 Marburg
Hans Meerwein Straße

25.Mai 2005

Übungen zu „Parallele Programmierung“, SS 2005

Benutzung von MPI am Fachbereich 12

Konfiguration

Am Fachbereich ist unter Linux die MPI-Implementierung LAM-MPI installiert. Um sie zu benutzen, müssen die folgenden Variablen gesetzt werden, am besten in der Shell-Konfigurationsdatei (normalerweise *.tcshrc*):

TC-Shell
```
setenv LAMHOME /app/lang/parallel/lam-7.0.4
setenv LAMRSH "ssh -x"
setenv PATH ${PATH}:${LAMHOME}/bin
setenv MANPATH $MANPATH:${LAMHOME}/man
```

Das Setzen des Pfads ist laut Anleitung erforderlich, damit LAM korrekt funktioniert. Der MANPATH muss dagegen nicht unbedingt gesetzt werden, falls man keine Anleitung braucht.

Werkzeuge

Werkzeuge zur Bedienung des Systems befinden sich in `${LAMHOME}/bin`. Dies sind insbesondere:

- `lamboot` Zum Starten eines parallelen Rechnerverbunds. In einer Datei (z.B. *lamhosts*) werden die Namen der beteiligten Rechner angegeben.

  ```
  # lamhosts: Rechner im LAM-MPI Universe
  # der lokale Rechner muss drin sein!
  #
  # SR auf D5, alle mit 2 CPUs (Hyper-Threaded):
  namibe cpu=2
  juba   cpu=2
  dodoma cpu=2
  kitwe  cpu=2
  ```

  `#>lamboot lamhosts` startet das parallele System.

- `lamnodes` gibt Informationen über das gestartete System.
- `lamclean` setzt das parallele System zurück.
- `lamhalt` fährt ein gestartetes System herunter.
- `wipe` erzwingt das Herunterfahren, falls Probleme mit `lamhalt` auftreten. Benutzung mit dem gleichen *hostfile* wie beim Starten: `#>wipe lamhosts`

Bei Angabe der Option -v werden (für viele Kommandos) zusätzliche Informationen über die ausgeführten Aktionen angezeigt.

TC-Shell

```
berthold@kitwe:VLparprog05/web> lamboot lamhosts

LAM 7.0.4/MPI 2 C++/ROMIO - Indiana University

berthold@kitwe:VLparprog05/web> lamnodes
n0      namibe:2:
n1      juba:2:
n2      dodoma:2:
n3      kitwe:2:origin,this_node
berthold@kitwe:VLparprog05/web> lamhalt -v

LAM 7.0.4/MPI 2 C++/ROMIO - Indiana University

Shutting down LAM
hreq: waiting for HALT ACKs from remote LAM daemons
hreq: received HALT_ACK from n0 (namibe)
hreq: received HALT_ACK from n1 (juba)
hreq: received HALT_ACK from n2 (dodoma)
hreq: received HALT_ACK from n3 (kitwe)
LAM halted
berthold@kitwe:VLparprog05/web>
```

Übersetzen und Starten von Programmen

Ein einfaches MPI-Programm

```
#include <stdio.h>
#include <mpi.h>

int main(int argc, char *argv[]) {
  int rank, size;
  char message[36], name[12];

  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &size);
  printf("! Knoten %d von %d startet.\n", rank, size);

  if (!rank) { // Hauptknoten: rank == 0
    int recvRank;
    MPI_Status status; /* mpi.h: struct _status{int MPI_SOURCE;int MPI_TAG;
                                              int MPI_ERROR;int st_length} */
    int i;
    for (i = 1; i < size; i++) {
      // empfange von irgendwem eine Nachricht (Tag 4711) mit einem String
      MPI_Recv(message, 35, MPI_CHAR,
               MPI_ANY_SOURCE, 4711, MPI_COMM_WORLD, &status);
      recvRank = status.MPI_SOURCE;
      printf("Knoten %d sagt: %s\n", recvRank, message);
    }
  } else { // andere Knoten:
    int len;
    MPI_Get_processor_name(name,&len);
    sprintf(message,"Knoten %.2d auf Rechner %s.",
```

```
            rank, name);
      // Senden der Nachricht "message" (Länge 35,Datentyp char, Tag 4711) an rank 0.
      MPI_Send(message,35,MPI_CHAR,0,4711,MPI_COMM_WORLD);
      }
    }
    // alle beenden das Programm.
    MPI_Finalize();
    return 0;
  }
```

Übersetzen von Programmen

Programme in C und MPI werden mit einem speziellen Compileraufruf `mpicc` übersetzt. Dieser ruft den eigentlichen Compiler, hier `gcc`, mit passenden Parametern, die man statt dessen auch selbst angeben könnte. Mit `mpicc -showme` wird der Aufruf angezeigt.
Auch wenn nur einzelne Module eines größeren Programms übersetzt werden, sollte `mpicc` verwendet werden.

TC-Shell

```
berthold@kitwe:VLparprog05/C> mpicc -O2 -o helloMPI HelloMPI.c -showme
gcc -I/app/lang/parallel/lam-7.0.4/include -pthread -O2 -o helloMPI HelloMPI.c
-L/app/lang/parallel/lam-7.0.4/lib -llammpio -llamf77mpi -lmpi -llam -lutil
berthold@kitwe:VLparprog05/C>
```

Starten

`mpirun` startet ein MPI-Programm. Als wichtigster Parameter wird angegeben, auf wie vielen "Knoten" (Rechnern im System) und wo das Programm laufen soll. Die Knoten (Nodes) im LAM-System werden dabei mit `n0` bis `n<zahl>` bezeichnet. Ein Knoten kann mehrere CPUs enthalten, die mit `c0` bis `c<zahl>` in Reihenfolge der Knoten nummeriert sind. Wichtige Optionen für `mpirun` sind:

Option	**Bedeutung**	**Beispiele**	
`N/C`	Programm auf allen verfügbaren Knoten () bzw. CPUs starten	`mpirun C myApp`	startet eine Instanz des Programms `myApp` pro CPU. Auf Knoten mit zwei Prozessoren werden also zwei Instanzen gestartet. Maßgeblich sind die im Hostfile angegebenen Knoten.
`-np <zahl>`	Programm auf `zahl` verfügbaren CPUs starten	`mpirun -np 12 myApp`	startet `myApp` auf 12 CPUs im System. Die Zuteilung erfolgt Round-Robin, evtl. werden mehrere Instanzen pro CPU gestartet.
`-n<liste>`	Programm auf in `liste` angegebenen Knoten starten	`mpirun -n0-2,2,2,2-3 myApp`	startet je eine Kopie von `myApp` auf den Knoten `n0,n1,n3` sowie insgesamt 4 Instanzen auf `n2`.
`-c<liste>`	Programm auf in `liste` angegebenen CPUs starten	`mpirun -c0-7 myApp`	startet `myApp` auf den CPUs `c0` bis `c7`. Falls im Hostfile je Knoten 2 CPUs angegeben waren, werden je zwei Kopien auf `n0,n1,n2` und eine auf `n3` gestartet.

Noch einfacher ist die Verwendung von `mpiexec`, die gleichzeitig das Starten und Beenden des LAM-Systems erledigen kann (passendes Hostfile im Aufruf).

TC-Shell
```
berthold@kitwe:VLparprog05/C> mpirun n0-3 helloMPI
! Knoten 3 von 4 startet.
! Knoten 0 von 4 startet.
! Knoten 1 von 4 startet.
! Knoten 2 von 4 startet.
Knoten 1 sagt: Knoten 01 auf Rechner juba.
Knoten 2 sagt: Knoten 02 auf Rechner dodoma.
Knoten 3 sagt: Knoten 03 auf Rechner kitwe.
berthold@kitwe:VLparprog05/C> mpirun c0-3 helloMPI
! Knoten 0 von 4 startet.
Knoten 1 sagt: Knoten 01 auf Rechner namibe.
! Knoten 1 von 4 startet.
! Knoten 2 von 4 startet.
! Knoten 3 von 4 startet.
Knoten 3 sagt: Knoten 03 auf Rechner juba.
Knoten 2 sagt: Knoten 02 auf Rechner juba.
berthold@kitwe:VLparprog05/C>
berthold@kitwe:VLparprog05/C> mpiexec -v -boot -machinefile ~/.www/parprog05/lamhosts c0-2 helloMPI
mpiexec: Booting lam..

LAM 7.0.4/MPI 2 C++/ROMIO - Indiana University

n0<23533> ssi:boot:base:linear: booting n0 (namibe)
n0<23533> ssi:boot:base:linear: booting n1 (juba)
n0<23533> ssi:boot:base:linear: booting n2 (dodoma)
n0<23533> ssi:boot:base:linear: booting n3 (kitwe)
n0<23533> ssi:boot:base:linear: finished
mpiexec: Lamboot Complete
mpiexec: Launching MPI programs
16325 helloMPI running on n0
16326 helloMPI running on n0
24624 helloMPI running on n1
! Knoten 0 von 3 startet.
! Knoten 1 von 3 startet.
! Knoten 2 von 3 startet.
Knoten 1 sagt: Knoten 01 auf Rechner namibe.
Knoten 2 sagt: Knoten 02 auf Rechner juba.
mpiexec: MPI program execution over..
mpiexec: Performing Lamhalt

LAM 7.0.4/MPI 2 C++/ROMIO - Indiana University

Shutting down LAM
hreq: waiting for HALT ACKs from remote LAM daemons
hreq: received HALT_ACK from n1 (juba)
hreq: received HALT_ACK from n0 (namibe)
hreq: received HALT_ACK from n2 (dodoma)
hreq: received HALT_ACK from n3 (kitwe)
LAM halted
mpiexec: Lamhalt complete
berthold@kitwe:VLparprog05/C>
```

Weitere Informationen

Anleitung für LAM-MPI: *http://www.lam-mpi.org/download/files/7.1.1-user.pdf*
Tutorials: *http://www.lam-mpi.org/tutorials/*