9th Advanced School on PARALLEL COMPUTING

Advanced MPI *hands-on*

Giusy Muscianisi – g.muscianisi@cineca.it
SuperComputing Applications and Innovation Department

CINECA

February 11 - 15, 2013

Blue Gene/Q Personality

Include file

#include <spi/include/kernel/location.h>

Structure

Personality_t pers

Query function

- Kernel_GetPersonality(&pers, sizeof(pers));

Properties

- pers.Network_Config.[A-E]nodes : Nb nodes in each torus dimension
- pers.Network_Config.[A-E]coord : Coordinates of the nodes in the torus
- pers.Network_Config.[A-E]bridge : Coordinates of the IO bridges in the torus

Other routines

Kernel_ProcessorID() : Processor ID (0-63)

Kernel_ProcessorCoreID() : Processor core ID (0-15)

Kernel_ProcessorThreadID() : Processor thread ID (0-3)

Blue Gene/Q Personality

```
#include <stdio.h>
#include <mpi.h>
#include <spi/include/kernel/location.h>

int main(int argc, char * argv[]){
        uint64_t Nflags;
        char procname[128];
        Personality_t pers;
        int rank, procid, core, hwthread, namelen;
        int Anodes, Bnodes, Cnodes, Dnodes, Enodes;
        int Acoord, Bcoord, Ccoord, Dcoord, Ecoord;
        int Atorus, Btorus, Ctorus, Dtorus, Etorus;

        MPI_Init(&argc, &argv);
        MPI_Comm_rank(MPI_COMM_WORLD, &rank);
        MPI_Get_processor_name(procname, &namelen);

        procid = Kernel_ProcessorID(); // 0-63
        core = Kernel_ProcessorCoreID(); // 0-15
        hwthread = Kernel_ProcessorThreadID(); // 0-3
```

Blue Gene/Q Personality

```
        Kernel_GetPersonality(&pers, sizeof(pers));
        Anodes = pers.Network_Config.Anodes; Acoord = pers.Network_Config.Acoord;
        Bnodes = pers.Network_Config.Bnodes; Bcoord = pers.Network_Config.Bcoord;
        Cnodes = pers.Network_Config.Cnodes; Ccoord = pers.Network_Config.Ccoord;
        Dnodes = pers.Network_Config.Dnodes; Dcoord = pers.Network_Config.Dcoord;
        Enodes = pers.Network_Config.Enodes; Ecoord = pers.Network_Config.Ecoord;
        Nflags = pers.Network_Config.NetFlags;
        if (Nflags & ND_ENABLE_TORUS_DIM_A) Atorus = 1; else Atorus = 0;
        if (Nflags & ND_ENABLE_TORUS_DIM_B) Btorus = 1; else Btorus = 0;
        if (Nflags & ND_ENABLE_TORUS_DIM_C) Ctorus = 1; else Ctorus = 0;
        if (Nflags & ND_ENABLE_TORUS_DIM_D) Dtorus = 1; else Dtorus = 0;
        If (Nflags & ND_ENABLE_TORUS_DIM_E) Etorus = 1; else Etorus = 0;

        if (rank == 0) {
        printf("block shape : <%d,%d,%d,%d,%d>\n", Anodes,Bnodes,Cnodes,Dnodes,Enodes);
        printf("torus links enabled : <%d,%d,%d,%d,%d>\n", Atorus,Btorus,Ctorus,Dtorus,Etorus);

        }

        printf("rank %d has processor name %s\n", rank, procname);
        printf("rank %d location <%d,%d,%d,%d,%d> core %d hwthread %d procid = %d\n",
                        rank,Acoord,Bcoord,Ccoord,Dcoord,Ecoord,core,hwthread,procid);

MPI_Finalize();

return 0;

}
```

MPI_Get_processor_name

```
int MPI_Get_processor_name( char *name,
                            int *resultlen );
```

Gets the name of the processor

Parameters:

name [out]

A unique specifier for the actual (as opposed to virtual) node. This must be an array of size at least MPI_MAX_PROCESSOR_NAME.

resultlen [out]

Length (in characters) of the name

MPI_Get_processor_name

```
int MPI_Get_processor_name(char *name, int
   *resultlen );
```

The name returned should identify a particular piece of hardware; the exact format is implementation defined. This name may or may not be the same as might be returned by gethostname, uname, or sysinfo.

This routine returns the name of the processor on which it was called at the moment of the call. The name is a character string for maximum flexibility. From this value it must be possible to identify a specific piece of hardware; possible values include "processor 9 in rack 4 of mpp.cs.org" and "231" (where 231 is the actual processor number in the running homogeneous system). The argument name must represent storage that is at least MPI_MAX_PROCESSOR_NAME characters long. MPI_GET_PROCESSOR_NAME may write up to this many characters into name.

The number of characters actually written is returned in the output argument, resultlen.

Identification of Compute Cards in a Blue Gene/Q

Each Compute Cards is identified as:

R.. - M. - N.. - J..

R = coordinates of the rack in the BG/Q system (row-colunm)
M = midplane inside the rack (0-1)
N = nodeboard number inside the midplane(00-15)
J = compute card number inside the nodeboard (00-31)

Blue Gene/Q Personality

```
if (rank == 0) {
        printf("block shape : <%d,%d,%d,%d,%d>\n", Anodes,Bnodes,Cnodes,Dnodes,Enodes);
        printf("torus links enabled : <%d,%d,%d,%d,%d>\n", Atorus,Btorus,Ctorus,Dtorus,Etorus);
}
printf("rank %d has processor name %s\n", rank, procname);
printf("rank %d location <%d,%d,%d,%d,%d> core %d hwthread %d procid = %d\n",
        rank,Acoord,Bcoord,Ccoord,Dcoord,Ecoord,core,hwthread,procid);
```

****

128-node block, 2 ranks per node, 256 total MPI tasks, mapping ABCDET

```
block shape : <2,2,4,4,2>
torus links enabled : <0,0,1,1,1>

rank 0 has processor name Task 0 of 256 (0,0,0,0,0,0) R02-M1-N00-J00
rank 0 location <0,0,0,0,0> core 0 hwthread 0 procid = 0

rank 1 has processor name Task 1 of 256 (0,0,0,0,0,1) R02-M1-N00-J00
rank 1 location <0,0,0,0,0> core 8 hwthread 0 procid = 32

rank 2 has processor name Task 2 of 256 (0,0,0,0,1,0) R02-M1-N00-J07
rank 2 location <0,0,0,0,1> core 0 hwthread 0 procid = 0
```

Blue Gene/Q Personality – Exercises

1) Build Personality binary:
   - Using IBM XL C compiler for Blue Gene
2) Build job submission file and submit job
3) Experiment various execution configurations in terms of:
   - Number of nodes
   - Number of tasks per node
   - Number of threads per task
4) Experiment various task placement policies