

Instrumenting your Fortran Code to Check Process/Thread Placement

Category: Process Pinning

Summary: Pinning, the binding of a process or thread to a specific core, can improve the performance of your code. To check whether your Fortran code has been successfully pinned, use the C code, `mycpu.c`, found below.

The MPI function `mpi_get_processor_name` and the Linux C function `sched_getcpu` can be inserted into your source code to check process and/or thread placement.

The MPI function `mpi_get_processor_name` returns the hostname an MPI process is running on (to be used for MPI and/or MPI+OpenMP codes only). The Linux C function `sched_getcpu` returns the processor number the process/thread is running on.

If your source code is written in Fortran, you can use the C code, `mycpu.c`, below, which allows your Fortran code to call `sched_getcpu`.

C Program mycpu.c

```
#include <utmpx.h>
int sched_getcpu();

int findmycpu_ ()
{
    int cpu;
    cpu = sched_getcpu();
    return cpu;
}
```

Compile `mycpu.c` as follows to produce the object file `mycpu.o`:

```
pfe20% module load comp-intel/2011.2
pfe20% icc -c mycpu.c
```

The example below demonstrates how to instrument an MPI+OpenMP source code with the above functions. The added lines are shown in red.

```
program your_program
  use omp_lib
  ...
  integer :: resultlen, tn, cpu
```

```

integer, external :: findmycpu
character (len=8) :: name

call mpi_init( ierr )
call mpi_comm_rank( mpi_comm_world, rank, ierr )
call mpi_comm_size( mpi_comm_world, numprocs, ierr )
call mpi_get_processor_name(name, resultlen, ierr)
!$omp parallel

    tn = omp_get_thread_num()
    cpu = findmycpu()
    write (6,*) 'rank ', rank, ' thread ', tn,
& ' hostname ', name, ' cpu ', cpu
.....
!$omp end parallel
call mpi_finalize(ierr)
end

```

Compile your instrumented code as follows:

```

pfe20% module load comp-intel/2011.2
pfe20% module load mpi-sgi/mpt.2.06a67
pfe20% ifort -o a.out -openmp mycpu.o your_program.f -lmpi

```

Sample PBS script

The PBS script below shows an example for running the hybrid MPI+OPenMP code across two nodes, with 2 MPI processes per node and 4 OpenMP threads per process, and using [mbind](#) to pin the processes and threads.

```

#PBS -lselect=2:ncpus=12:mpiprocs=2:model=wes
#PBS -lwalltime=0:10:00

cd $PBS_O_WORKDIR

module load comp-intel/2011.2
module load mpi-sgi/mpt.2.06a67

mpiexec -np 4 mbind.x -cs -n2 -t4 -v ./a.out

```

Here is a sample output:

```

These 4 lines are generated by mbind only if you have included the -v option:
host: r212i1n8, ncpus 24, process-rank: 0, nthreads: 4, bound to cpus: {0-3}
host: r212i1n8, ncpus 24, process-rank: 1, nthreads: 4, bound to cpus: {6-9}
host: r212i1n9, ncpus 24, process-rank: 2, nthreads: 4, bound to cpus: {0-3}
host: r212i1n9, ncpus 24, process-rank: 3, nthreads: 4, bound to cpus: {6-9}

```

```

These lines are generated by your instrumented code:
rank    0 thread    0 hostname r212i1n8 cpu    0
rank    0 thread    1 hostname r212i1n8 cpu    1
rank    0 thread    2 hostname r212i1n8 cpu    2

```

```
rank    0 thread    3 hostname r212i1n8 cpu    3
rank    1 thread    0 hostname r212i1n8 cpu    6
rank    1 thread    1 hostname r212i1n8 cpu    7
rank    1 thread    2 hostname r212i1n8 cpu    8
rank    1 thread    3 hostname r212i1n8 cpu    9
rank    2 thread    0 hostname r212i1n9 cpu    0
rank    2 thread    1 hostname r212i1n9 cpu    1
rank    2 thread    2 hostname r212i1n9 cpu    2
rank    2 thread    3 hostname r212i1n9 cpu    3
rank    3 thread    0 hostname r212i1n9 cpu    6
rank    3 thread    1 hostname r212i1n9 cpu    7
rank    3 thread    2 hostname r212i1n9 cpu    8
rank    3 thread    3 hostname r212i1n9 cpu    9
```

Note that these lines in your output may be listed in a different order.

This approach was suggested by NAS SGI analyst Ken Taylor.

Article ID: 309

Last updated: 18 Dec, 2012

Computing at NAS -> Best Practices -> Process Pinning -> Instrumenting your Fortran Code to Check Process/Thread Placement

<http://www.nas.nasa.gov/hecc/support/kb/entry/309/?ajax=1>