Cornell University
Center for Advanced Computing

Hybrid Computing Lab

John Zollweg

Introduction to Parallel Computing
May 29, 2009

Based on material developed by Kent Milfeld, TACC

What you will learn

- Using numactl in execution of serial, MPI and a 4x# (4 tasks each with # threads) hybrid code

- Communications in Hybrid codes
  - Communication between nodes with threaded MPI calls.
    - MPI calls from serial region
    - MPI calls from master thread in a parallel region
    - MPI calls from all threads in a parallel region

Hybrid Computing – Ranger

- Untar the file numahybrid.tar
  - cd (Start in your home directory.)
  - tar xvf ~train200/numahybrid.tar (extract files)
  - cd numahybrid

numactl_serial – Ranger

The memory intensive daxpy code is run on four different sockets using local, interleave and off-socket-memory policies. The commands below make the daxpy executable and run it with numa control commands. See the job script and the table on the next page for the numa options. Run the job and report the times and relative performance.

- Execute:
  - cd numactl_serial (change directory to numactl_serial)
  - module unload mvapich; module swap pgi intel; module load mvapich
  - make
  - qsub job (submits job)

numactl_serial – Ranger

- From the job output fill in the table.

Command	Time (secs)
numactl –l -C 0	
numactl –l -C 1	
numactl –l -C 2	
numactl –l -C 3	
numactl –i all –C 0	
numactl –i all –C 1	
numactl –i all –C 2	
numactl –i all –C 3	
Numactl –m 3 –C 0	

Rank the performance of local, interleave, and off-Socket-memory policies.

1.)

2.)

3.)

From best to poorest performance.

Numactl_mpi – Ranger

The memory intensive mpi_daxpy code is run on four sockets simultaneously using local, interleave and tacc_affinity policies. The commands below make the mpi_daxpy executable and run it with numa control commands. See the job script and the table on the next page for the numa options. Run the job and report the times and relative performance.

- Execute:
  - cd numactl_mpi (change directory to numactl_mpi)
  - if you have done this already, don't do it again:
    module unload mvapich; module swap pgi intel; module load mvapich
  - make
  - qsub job (submits job)

Numactl_mpi – Ranger

- From the job output fill in the table.

Command	Time (secs)
numactl –l	
numactl –i all	
numactl tacc_affinity	

Rank the performance of local, interleave, and tacc_affinity memory policies.

1.)

2.)

3.)

From best to poorest performance.

Numactl_4x1, Numactl_4x4 – Ranger

The daxpy code is run as 4 tasks in a node (4x1) and 4 tasks with 4 threads in a node (4x4). Cd down to directories numactl_4x1 and numactl_4x4, respectively, and follow the instructions below.

- Execute:
  - cd numactl_4x1 or _4x4(change directory to numactl_mpi)
  - if you have done this already, don't do it again:
    module unload mvapich; module swap pgi intel; module load mvapich
  - make
  - qsub job (submits job)

Numactl_4x1, Numactl_4x4 – Ranger

- From the job output fill in the table.

Command (4x1)	Time (secs)
<no numactl> numactl –l	
numactl –i all	
numactl tacc_affinity	
Command (4x4)	**Time (secs)**
<no numactl> numactl –l	
numactl –i all	
numactl tacc_affinity	

Rank the 4x1 performance of.

1.)

2.)

3.)

From best to poorest performance.

Rank the 4x4 performance of.

1.)

2.)

3.)

From best to poorest performance

Communications in Hybrid codes

- The tmpi (threaded mpi) code, performs various types of communication (point-2-point and broadcast) within a hybrid code. Check to make sure the code performs the operations correctly, compare the cost of sending a single large message in the serial region, and 16 small messages in the parallel region (both mvapich and openmpi MPIs are used.)
- Execute:
  cd threaded_mpi
    if you have done this already, don't do it again:
    module unload mvapich; module swap pgi intel; module load mvapich
  ./build.sh
- This builds tmpi.mvapich1 and tmpi.openmp

Hybrid Job Script

```
#!/bin/tcsh
#                              # use bash shell
#$ -V                          # inherit submission environment
#$ -cwd                        # use submission directory
#$ -N threadedmpi              # jobname (threadedmpi)
#$ -j y                        # stdout/err combined
#$ -o $JOB_NAME.o$JOB_ID       # output name jobname.ojobid
#$ -pe 1way 32                 # 1 task/node, 32 cores total
#$ -q development              # queue name !! can use normal
#$ -l h_rt=00:10:00            # request 10 minutes
## -M <myemail_addr>           # Mail address !! your own mail
## -m be                       # send email at begin/end of job}
#$ -A 20090528HPC              # your account
 set echo                      # echo cmds, use "set -x" in sh

 setenv MY_NSLOTS 2
 setenv OMP_NUM_THREADS 16
 ibrun ./tmpi < input
```

If # of tasks is not a multiple of 16, set value here.

This will give you 10 exclusive minutes of 2 nodes (=32/16) 1 task per node (1-way) and a total of 2 tasks in the development queue. 16 threads (OMP_NUM_THREADS 16) are launched on each node.

MPI/OpenMP Ranger

- Submit the batch job:

% qsub job ...

Welcome to TACC's Ranger System, an NSF Teragrid Resource
--> Submitting 2 tasks...
--> Submitting 1 tasks/host...
--> Submitting exclusive job to 2 hosts...
...
Your job 18073 ("threadedmpi") has been submitted

% qstat

```
job-ID  prior   name       user     state submit/start at     queue         slots
-----------------------------------------------------------------------------
  18075 0.00001 threadedmp milfeld  r     01/17/2008 22:48:54 normal@i104-408 32
```

% showq

...

Call from Serial Region – Ranger

```
include "mpif.h"
…
call MPI_Init_thread(MPI_THREAD_MULTIPLE, iprovided,ierr)
call MPI_Comm_size(MPI_COMM_WORLD,nranks, ierr)
call MPI_Comm_rank(MPI_COMM_WORLD,irank,ierr)

if(irank == 0) then
   call mpi_send(as,n,MPI_REAL8, 1,9,MPI_COMM_WORLD, ierr)
   call mpi_recv(as,n,MPI_REAL8, 1,1,MPI_COMM_WORLD, istatus,ierr)
else if (irank == 1) then
   call mpi_recv(as,n,MPI_REAL8, 0,9,MPI_COMM_WORLD, istatus,ierr)
   call mpi_send(as,n,MPI_REAL8, 0,1,MPI_COMM_WORLD, ierr)
endif

if(irank .eq. 0) read(*,'(i5)') iread1
call MPI_Bcast(iread1,1,MPI_INTEGER, 0,iwcomm, ierr)
```

"Serial Code"

Broadcast in // Region – Ranger

```
!$OMP PARALLEL private(i,ithread,nthreads, icp1, icp2, icpd)


ithread =OMP_GET_THREAD_NUM()

if(ithread == 0) then
   if(irank .eq. 0) read(*,'(i5)') iread2
   call MPI_Bcast(iread2,1,MPI_INTEGER, 0,iwcomm, ierr)
end if
```

Parallel Region

Multi-threaded communication

```
!$OMP DO ordered
  do i = 1,nthreads
!$OMP ordered
   if(irank == 0) then
      call mpi_send(as,ns,MPI_REAL8, 1,ithread,MPI_COMM_WORLD, ierr)
      call mpi_recv(as,ns,MPI_REAL8, 1,ithread,MPI_COMM_WORLD, istatus,ierr)
   else if (irank == 1) then
      call mpi_recv(as,ns,MPI_REAL8, 0,ithread,MPI_COMM_WORLD, istatus,ierr)
      call mpi_send(as,ns,MPI_REAL8, 0,ithread,MPI_COMM_WORLD,ierr)
   endif
!$OMP end ordered
  end do

   if(irank == 0 .and. ithread == 0) then
        call mpi_send(as,n,MPI_REAL8, 1,ithread,MPI_COMM_WORLD, ierr)
        call mpi_recv(ar,n,MPI_REAL8, 1,ithread,MPI_COMM_WORLD, istatus,ierr)
   else if (irank == 1 .and. ithread == 0) then
        call mpi_recv(ar,n,MPI_REAL8, 0,ithread,MPI_COMM_WORLD, istatus,ierr)
        call mpi_send(as,n,MPI_REAL8, 0,ithread,MPI_COMM_WORLD, ierr)
   endif
   !$OMP barrier

   !$OMP END PARALLEL
   call mpi_finalize(ierr)
```

Parallel Region

Each Thread Sends (block size = ns).

Not needed with mvapich2

Only Thread 0 Sends (block size =n =16 x ns).

End of Parallel

End of MPI

MPI/OpenMP – Ranger

Hybrid Communication Cost (Output from tmpi):

```
Mvapich1
 Serial Region Ping Pong   (words:secs) 400000:      0.00509
 Serial Region Broadcast          (sec)             0.00002
 Parallel Region Broadcast        (sec)             0.00001
 Parallel region messages:
 One Large message   size:secs 400000 tot time:     0.00555
 16  Small messages  size:secs  25000 tot time:     0.00548

individual times:  0.00042  0.00033  0.00038  0.00033  0.00033  0.00033
0.00033  0.00033  0.00033  0.00034  0.00033  0.00033  0.00033  0.00034  0.00033  0.00033

OpenMPI
 Serial Region Ping Pong   (words:secs) 400000:      0.00501
 Serial Region Broadcast          (sec)             0.00005
 Parallel Region Broadcast        (sec)             0.00001
 Parallel region messages:
 One Large message   size:secs 400000 tot time:     0.00553
 16  Small messages  size:secs  25000 tot time:     0.13446

individual times:  0.12864  0.00038  0.00038  0.00039  0.00038  0.00065  0.00036
0.00036  0.00038  0.00034  0.00038  0.00037  0.00036  0.00037  0.00036  0.00036
```