

Optimization and Scalability Lab

Carlos Rosales
carlos@tacc.utexas.edu

September 13th, 2011
Introduction to Parallel Computing



THE UNIVERSITY OF TEXAS AT AUSTIN
TEXAS ADVANCED COMPUTING CENTER

Setup

- Login to Ranger:
 - `ssh username@ranger.tacc.utexas.edu`
- Make sure you are using the intel compiler
 - `module swap pgi intel`
 - `module list`
- Untar the lab files:
 - `cd`
 - `tar xvf ~train00/opt_lab.tar`
- Change directories and ls to see the files:
 - `cd opt_lab`
 - `ls`

Serial Optimization (Optional)

- Compile serial.c or serial.f90 with increasing levels of automatic optimization:
 - `icc -O0 serial.c -o serial_00 / ifort -O0 serial.f90 -o serial_00`
 - `icc -O1 serial.c -o serial_01 / ifort -O1 serial.f90 -o serial_01`
 - `icc -O2 serial.c -o serial_02 / ifort -O2 serial.f90 -o serial_02`
 - `icc -O3 serial.c -o serial_03 / ifort -O3 serial.f90 -o serial_03`
- Notice the optimization messages printed by the Intel compiler.
 - At the O2 level three loops are vectorized
 - At the O3 level three loops are fused into one, and then the larger loop is vectorized
- Compile using **-opt-report 2** to get additional information on the optimizations performed
- Submit the job through the SGE queue system:
 - `qsub ./serial_job.sge`
- Construct a table with the total timing for the different executables (get the timings from the output file **serial.\$JOBID.out**)

Parallel Scalability

- In this example you will use **mpiP** to evaluate the scalability of a matrix multiplication code.
- Load the mpiP module:
 - `module load mpiP`
 - `module list`
- Compile the matmult.c or matmult.f90 source with the flags required to link in the mpiP library:
 - `mpicc -g -L$TACC_MPIP_LIB -lmpiP -lbfd -liberty ./matmult.c`
 - `mpif90 -g -L$TACC_MPIP_LIB -lmpiP -lbfd -liberty ./matmult.f90`
- Set the environmental variables that control mpiP data collection behavior:
 - `setenv MPIP '-t 10 -k 2'`
- Submit the job through the SGE queue system:
 - `qsub ./parallel_job.sge`

Parallel Scalability II

- The initial submission using 2 processing cores only (-pe 2way 16). Check execution and MPI times in the .mpiP file created.
- Change the submission script to use 4 cores (-pe 4way 16), 8 (-pe 8way 16) and 16 (-pe 16way 16), and build a table with the execution times.
- Does the execution time decrease linearly with the number of cores? Why?
- Open the matmult.c or matmult.f90 file and change the parameter SIZE_OF_MATRIX from the default 2000 to 1000.
- Repeat runs for 2, 4, 8, and 16 cores using this size and fill in the table below.
- Is there a difference in the scaling when you compare the results for the 2000 and the 1000 size problems? Why?

SIZE	2 cores	4 cores	8 cores	16 cores
1000 x 1000				
2000 x 2000				