MPI Java and TAU tutorial

Abstract

The goal of this project was to create a tutorial for installing and using Openmpi Java on the Rocket cluster of the University of Tartu and to use TAU for profiling them.

Introduction

Mpi

MPI is a language-independent communications protocol used to program parallel computers. Both point-to-point and collective communication are supported. MPI "is a message-passing application programmer interface, together with protocol and semantic specifications for how its features must behave in any implementation." MPI's goals are high performance, scalability, and portability.

Open MPI

The Open MPI Project is an open source Message Passing Interface implementation that is developed and maintained by a consortium of academic, research, and industry partners. Open MPI is therefore able to combine the expertise, technologies, and resources from all across the High Performance Computing community in order to build the best MPI library available. Open MPI offers advantages for system and software vendors, application developers and computer science researchers.[1]

Other mpi-java implementations were considered but due to time constraint and configuration limitations open MPI was chosen for this tutorial. Other implementations include mpiJava[2], P2P-MPI[3] and MPJ[4].

Rocket cluster

The main part of the Rocket cluster consists of 135 compute nodes (called stage1 - stage135), 4 compute nodes with Intel Xeon Phi accelerator cards (called booster1 - booster4) and a headnode rocket.hpc.ut.ee. In addition to these nodes there is a large memory machine (2TB of RAM) atlas.hpc.ut.ee, two "fat nodes" (sputnik1 and sputnik2), which can server as special purpose nodes (running virtual machines for example). There are also two GPFS filesystem servers tank1 and tank2 which will provide fast storage for the entire cluster.[5]

All the machines mentioned above are connected to a fast Infiniband 4X QDR fabric, powered by 8 Mellanox switches. In addition to Infiniband, all the aforementioned machines are also connected to a regular ethernet network for easier access.

TAU

TAU Performance System® is a portable profiling and tracing toolkit for performance analysis of

parallel programs written in Fortran, C, C++, UPC, Java, Python.

TAU is a performance evaluation tool. It supports parallel profiling and tracing. Profiling shows you how much (total) time was spent in each routine. Tracing shows you when the events take place in each process along a timeline. Profiling and tracing can measure time as well as hardware performance counters (cache misses, instructions) from your CPU. TAU can automatically instrument your source code using a package called PDT for routines, loops, I/O, memory, phases, etc. TAU runs on most HPC platforms and it is free (BSD style license).TAU has instrumentation, measurement and analysis tools.

Installing Open MPI

I have included an install script for open MPI with java bindings. It is essentially the same script provided to us in lab 3 but with unnecessary options removed.

GCC and jdk modules are required to install open mpi to rocket. I am using gcc-4.8.1 and jdk-1.8.0_25. These modules can be loaded in rocket using *module load* command: *module load gcc-4.8.1; module load jdk-1.8.0_25.*

To give installation directory a name for convenience we export *MPIDIR* as our mpi directory like this: export *MPIDIR=OpenMPI.*

Next step is to make the installation directory using mkdir: *mkdir $MPIDIR.* Then change directory to the MPIDIR by using cd: *cd $MPIDIR.*

Now to download the open mpi we use wget: *wget* [*www.open-mpi.org/software/ompi/v1.8/downloads/openmpi-1.8.4.tar.gz*](www.open-mpi.org/software/ompi/v1.8/downloads/openmpi-1.8.4.tar.gz) To decopress it we use tar: *tar -xvf openmpi-1.8.4.tar.gz.*

Next go to the uncompressed directory: *cd openmpi-1.8.4*. We need to configure the open mpi to use both slurm and java bindings for that we use the configure command: *./configure --prefix=$HOME/$MPIDIR/install --with-slurm --with-jdk-dir=/storage/software/jdk-1.8.0-25 –enable-mpi-java.*

Prefix specifies the installation directory, *--with-slurm means we can use slurm to run our mpi java programs, --with-jdk-dir* specifies the jdk directory and *–enable-mpi-java* enables the use of java bindings in open mpi.

Now to compile it we need to use the make command but since rocket has some problems[6] when compiling Java applications we need to run make as an interactive job. *srun -N 1 -t 0:40:00 --mem=16384 make* and also to install the library we use make install: srun -N 1 -t 0:40:00 --mem=16384 make install. Now we just need to update path so that we can use the libraries: *export PATH=$HOME/$MPIDIR/install/bin:$PATH*
export LD_LIBRARY_PATH=$HOME/$MPIDIR/install/lib:$LD_LIBRARY_PATH

Running the examples

To compile the examples which were provided with the open mpi we can use the open mpi javac wrapper compiler called mpijavac. But since a makefile is provided for the examples we can use it to compile them. First changing the directory to examples: *cd examples*. Next use the makefile to compile the examples: s*run -N 1 -t 0:40:00 --mem=16384 make.*

And finally to run the examples we use interactive jobs again:

srun -N 1 -n 20 -t 0:10:00 --mem=16384 $HOME/$MPIDIR/install/bin/mpirun -np 20 java Hello
srun -N 1 -n 20 -t 0:10:00 --mem=16384 $HOME/$MPIDIR/install/bin/mpirun -np 20 java Ring

Which will run the examples on one node and 20 cores.

To compile the java code without an makefile we can use mpijavac to do so. I have included a simple Pi calculation and matrix multiplication examples and to compile them we use mpijavac:
srun -N 1 -n 1 -t 0:01:00 --mem=16384 $HOME/$MPIDIR/install/bin/mpijavac Pi.java

srun -N 1 -n 1 -t 0:01:00 --mem=16384 $HOME/$MPIDIR/install/bin/mpijavac Matrixmult.java

And similary with the examples we can run them with mpirun.

Profiling

I decided to try to use TAU[7] for profiling mpi java programs. I ran into several difficulties which are described in the problem section. I have included a script for downloading and installing TAU with mpi and Java support on Rocket.

First we need to load appropriate modules:

module purge
module load gcc-4.8.1
module load jdk-1.8.0_25
module load java-1.8.0_40
module load zlib-1.2.6
module load libpng-1.5.10

Next we create variables for installation directory: *export PROFILINGDIR=profiling*
export TAUDIR=TAU

Then install the Program Database Toolkit (PDT) to take advantage of TAU's automatic instrumentation features: *wget www.cs.uoregon.edu/research/TAU/pdt_releases/pdt-3.20.tar.gz*.

Then untar it: *tar -xvf pdt-3.20.tar.gz.* Change directory: *cd pdtoolkit-3.20/*. Configure it: *./configure -prefix=$HOME/$PROFILINGDIR/$TAUDIR/pdtinstall.* And install: *make; make install.* Change the directory back*: cd* ..

Now it is time to download TAU: *wget http://TAU.uoregon.edu/TAU2.tgz* and unpack it: *tar -xvf TAU2.tgz.* Go to TAU2 directory using: *cd TAU2/*. First download and install binutils: *./configure -bfd=download.* Then configure TAU with java and mpi support: *./configure -useropt='-std=gnu++11' -jdk=/storage/software/jdk-1.8.0-25 -bfd=download -mpiinc=$HOME/$MPIDIR/install/include/ -mpilib=$HOME/$MPIDIR/install/lib/ -prefix=$HOME/*

$PROFILINGDIR/$TAUDIR/TAU2install -pdt=$HOME/$PROFILINGDIR/$TAUDIR/pdtinstall The -bfd option is for using binutils which seem to required for some libraries. -useropt is required for java support and also -jdk option specifies the path to java root directory. -mpiinc nad -mpilib options are required for user defined mpi installation (by default TAU searches an mpi solution from path). The -pdt is for using Program Database Toolkit. To install TAU we then use: *make install*.

After the install we add the TAU bin directory to path and then we are ready to use TAU: *export PATH=$HOME/$PROFILINGDIR/$TAUDIR/TAU2install/x86_64/bin:$PATH.*

Now to verify that it is working with Java we can just go to java examples and try it out: *cd TAU2install/examples/java/pi and* srun -N 1 -n 4 -t 0:10:00 --mem=16384 TAU_java Pi 20000000.

Now it generates a profile file. Using pprof we can see the output of the profile file: *pprof*

Reading Profile files in profile.*

NODE 0;CONTEXT 0;THREAD 0:

%Time	Exclusive msec	Inclusive total msec	#Call	#Subrs	Inclusive usec/call	Name
100.0	19	51	1	1	51563	THREAD=main GROUP=main
62.6	24	32	1	1	32282	Pi main ([Ljava/lang/String;)V
15.9	8	8	1	1	8191	PiCalculateValue getPi (I)D
0.0	0.016	0.016	1	0	16	PiCalculateValue loop (I)V

Here we can see how much time each of the soubroutines took of total time. Paraprof can be used to see it in graphical format which can be easier to understand.

Problems

Mainly I had problems for getting TAU installed in Rocket. I cannot be certain if it is because of bugs in TAU or something in Rocket. After contacting the TAU support, they finally provided a modified TAU tarball which with a little experimenting I was able to install to Rocket. Now the main difficulty is to get TAU to work on mpi and java since ity works with both mpi and java separately but has errors when trying to run on mpi java. I'm still in contact with TAU support and we are trying to make it work with java mpi.

Conclusion

In conclusion I would say that although open mpi provides almost all mpi 3 bindings for Java, there are some limiting factors for usage. First Java memory structure is different from Fortran and C. In C and Fortran we can give an offset for mpi send in case where we send an array and we do not want to send the whole array but from some element to the end. But with open mpi java we cannot do that (other mpi java implementations support this). On Rocket the need to start interactive jobs can also confuse people who would want to compile and run java programs. Secondly there are few examples to learn from and even the official documentation does not give us java binding examples

instead we can only look it up from javadoc which is generated during the installation or we can see the java specification from here[8]. So for easy to implement programs I see no problem in using java especially when the developer is more home with java. The downside is that Java virtual machine is required to run the programs and there can be some errors when several Java versions are installed. I have cut TAU part short since I cannot profile mpi java programs yet but will add more if it is successful.

References

1. http://www.open-mpi.org/
2. http://www.hpjava.org/mpiJava.html
3. http://grid.u-strasbg.fr/p2pmpi/
4. http://mpj-express.org/
5. http://hpc.ut.ee/rocket_cluster
6. http://hpc.ut.ee/user_guides/faq
7. https://www.cs.uoregon.edu/research/TAU/home.php
8. www.open-mpi.org/papers/mpi-java-spec/