Chapter 1. TAU Tutorial

Table of Contents

1.1. Gather information

Before we began installing PDT and TAU you will find it helpful to gather information about your computing environment. TAU and PDT require both a C and C++ compiler. Furthermore this tutorial uses MPICH, if you use a different implementation just find its lib and include directories. Find out the following information about your computing environment before we began:

- The path to your MPI lib directory
- The path to your MPI include directory
- The path of your C compiler (same as the used to compile MPI)
- The path of your C++ compiler (same as the used to compile MPI)

For the remaining of this tutorial we will assume that your C compiler is xlc, your C++ compiler is xlC.

1.2. Installing PDT

To take advantage of TAU's automatic instrumentation features, you will need to install the Program Database Toolkit (PDT). Download the latest version from the PDT pages and put the tar.gz package in the location that you want to install PDT. For this installation, we will assume that you are using IBM's Fortran and C/C++ compilers, with an mpich installation.

Start by uncompressing the PDT package and moving into the PDT directory.

```
%> tar -xvzf pdtoolkit-x.x.tar.gz
%> cd pdtoolkit
```

You can get a sense for what options you can configure PDT with by entering:

```
%> ./configure
Program Database Toolkit (PDT) Configuration
----------------------------------------------
Looks like a Linux machine ...
Looking for C++ compilers .... done
Usage: ./configure [-GNU|-CC|-c++|-cxx|-xlC|-pgCC|-icpc|-ecpc]
                   [-arch=ibm64|ibm64linux|IRIXO32|IRIXN32|IRIX64] [-help]
                   [-compdir=<compdir>>]
                   [-enable-old-headers]
```

```
                    [-useropt=<options>>]
                    [-prefix=<dir>]
                    [-exec-prefix=<dir>>]
```

We will configure PDT for use the the Fortran xlF, xlc, and xlC compilers. To configure PDT, type

```
%> ./configure -xlC

Program Database Toolkit (PDT) Configuration
--------------------------------------------
Looks like a Linux machine ...
Looking for C++ compilers .... done
==> Using /opt/ibmcmp/vacpp/6.0/bin/xlC
Unpacking ppc64/bin ...
==> ARCH is PPCLINUX
==> PLATFORM is ppc64
==> Default compiler options are -O2
==> Makefiles were configured
==> cparse was configured
==> cxxparse was configured
==> f90parse was configured
==> f95parse was configured

Configuration is complete!

Run "make" and "make install"
Add "/home/users/hoge/pdtoolkit-3.4/ppc64//bin" to your path
```

Add the specified directory to your path. In bash, for example you could enter

```
%> export PATH=$PATH:/home/users/hoge/pdtoolkit/ppc64/bin
```

Now, build and install PDT. Unless you specify a different location to install PDT, it will be placed in the current working directory.

```
%> make
...
%> make install
```

Now you're ready to proceed with the TAU installation.

1.3. Installing TAU

Download the latest version of TAU from the TAU home. Place the distribution In the directory that you want to install TAU. Type

```
%> tar -xvzf tau_latest.tar.gz
%> cd tau2
```

We will be installing TAU once again assuming that we are using the IBM compilers (xlf, xlc and xlC), and an MPICH installation. Note where your MPICH installation resides, and configure TAU by entering (replacing the MPICH specifics with those in your local system.

```
%> ./configure -c++=xlC -cc=xlc -fortran=ibm \
  -mpiinc=/opt/osshpc/mpich-1.2.5/32/ch_shmem/include \
  -mpilib=/opt/osshpc/mpich-1.2.5/32/ch_shmem/lib \
```

```
  -pdt=/home/users/hoge/pdtoolkit
```

Add the TAU directory to your path and install.

```
%> export PATH=$PATH:/home/users/hoge/tau2/ppc64/bin
%> make install
```

TAU is installed, and you're ready to start profiling your code.

1.4. Automatic instrumentation using TAU Compiler

For this section of the tutorial we will be using the files found in the `examples/taututorial` directory of the tau distribution. To start, there are two files of note: computePi.cpp and Makefile. computePi.cpp is a C++ program that uses an MPI client-server model to estimate the value of Pi. The server accepts requests for random numbers from the clients, and returns an array of random numbers to the clients. The clients use these values to estimate Pi using a dart-throwing method. When the clients have converged to a satisfactory tolerance, they signal their completion to the server and the program exits.

Build computePi.cpp as you would any c++ mpi application.

```
%> mpicxx -c computePi.cpp -o computePi.o
%> mpicxx computePi.o -o computePi
```

Test the program in your MPI environment. For mpich, the command might be

```
%> mpirun -np 5 ./computePi
Pi is 3.14226
```

to run the program on 5 nodes. Note that this program requires at least two nodes to be running! Once you've confirmed that the program ran successfully, try timing it to get a sense of how long it takes to run.

```
%> time mpirun -np 5 ./computePi
Pi is 3.14226

real    0m2.012s
user    0m1.570s
sys     0m0.330s
```

Now let us rebuild computePi to be instrumented with tau. First we need to tell TAU which instrumentation library to use by setting the environment variable TAU_MAKEFILE to the location of the tau makefile, for example:

```
%> export TAU_MAKEFILE=/home/users/hoge/tau2/ia64/lib/Makefile.tau-mpi-pdt
%> tau_cxx.sh -c computePi.cpp -o computePi.o
%> tau_cxx.sh computePi.o -o computePi
```

Assuming that all goes well, the computePi program will have been automatically built with TAU instrumentation. Run the program as you would any MPI program, i.e.

```
%> time mpirun -np 5 ./computePi
Pi is 3.14226

real    0m2.123s
user    0m1.760s
sys     0m0.270s
```

TAU generates a profile file for every node the program is run on. You can see these files by doing a directory listing.

```
%> ls profile*
profile.0.0.0  profile.1.0.0  profile.2.0.0
profile.3.0.0  profile.4.0.0
```

Now you're ready to view the output of TAU. If you've added the TAU binary directory to your path you can launch the TAU profile viewer, Paraprof.

```
%> paraprof
```

Enjoy exploring the performance data displayed by Paraprof. A complete description of how to use Paraprof is outside the scope of this document. Please see the Paraprof Manual [http://www.cs.uoregon.edu/research/tau/docs/paraprof/index.html] for more information.