TAU:591C Report 2

Project Team

Sibin Mohan, Salil Pant, Harini Ramaprasad

{ smohan, smpant, hramapr }@unity.ncsu.edu

Our Project:

- *Installation of Tuning and Analysis Utilities (TAU), a portable profiling and tracing toolkit, on the i32/Linux platform.*
- *Evaluation using ParBenCCh (from the ASCI suite of benchmarks).*

Project Topic – TAU

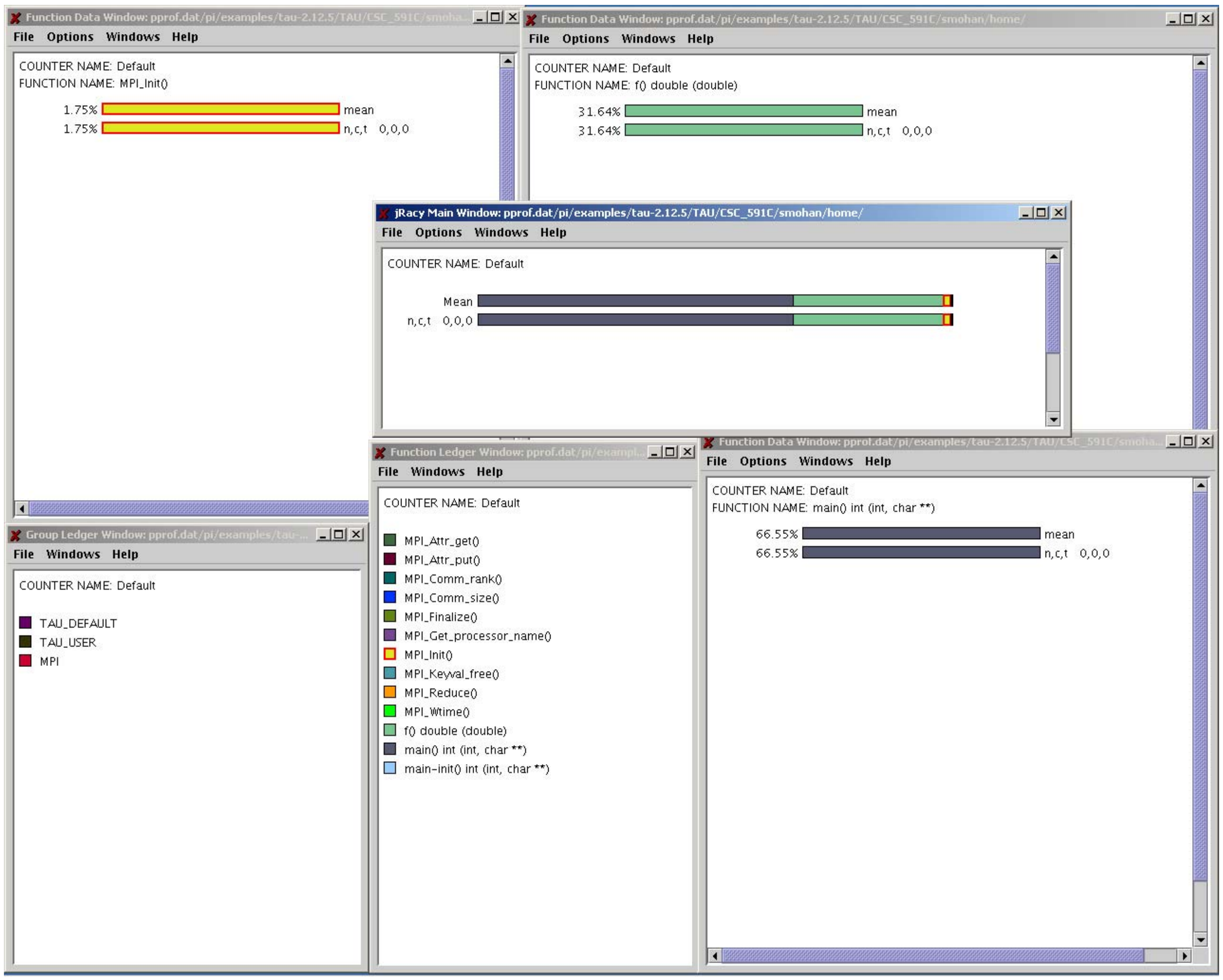
Solved Issues :

- *TAU is a profiling tool used for parallel multithreaded programs. It supports various profiling tools like PAPI, PDT etc and can be used with both OPENMP as well as MPI.*
- *We installed the package on the Linux cluster. Installation was carried out according to the steps mentioned in the README file provided with the package. The steps involved :*
    - *Download and install other libraries/toolkits that TAU uses:*
        - *Program Database Toolkit [http://www.acl.lanl.gov/pdtoolkit] (for automatic source code instrumentation)*
        - *Performance Data Standard and API (PAPI) Library [http://icl.cs.utk.edu/projects/papi/] (for accessing CPU Performance Counters with TAU)*

  - *Opari OpenMP source-to-source instrumentation package [http://www.fz-juelich.de/zam/kojak/] (for instrumenting OpenMP applications)*

- *Configure TAU based on –*
  - *the architecture*
  - *the profiler used*
  - *the type of programs to be tested*

  *This involves running the “**configure**” script with different options. The complete set of options is listed in the “**INSTALL**” file provided with the package.*
- *During installation and configuration of TAU and the other toolkits, various errors/problems were encountered, which had to be resolved before we could proceed. Some of the changes made as a result were :*
  - *The compiler to be used was modified from g++/gcc to mpiCC/mpicc.*
  - *Remove certain compiler flags that do not work with the gnu compilers/mpi compilers.*
  - *Append paths to header files/libraries for MPI/OpenMP.*
- *We also had to install the Java SDK and Java Runtime Environment, for **jRacy** (one of the TAU graphical tools).*
- *The following two tools were run on various sample programs:*
  - *jRacy – a Java based tool which displays results graphically*
  - *pprof - a command line based tool, which displays profiling results in a textual format.*
- *Sample programs and Results :*
  - *Calculation of “**pi**” using MPI. The output of the jRacy tool is as follows :*

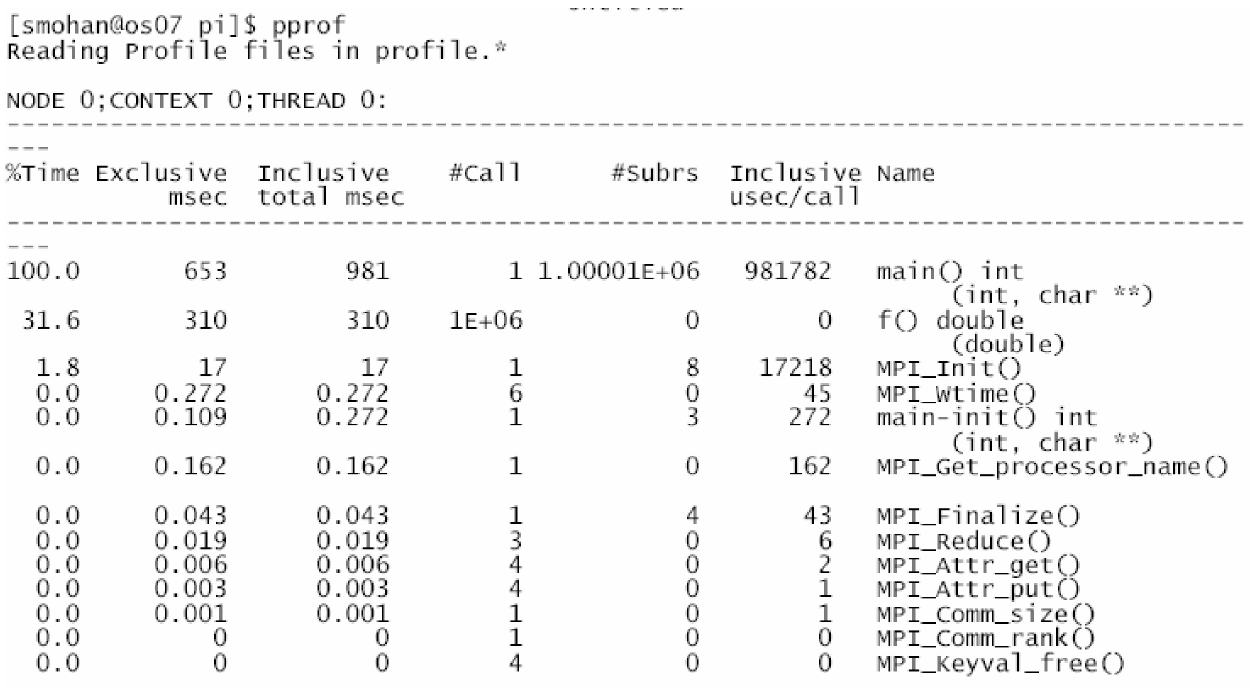


```
Function Data Window: pprof.dat/pi/examples/tau-2.12.5/TAU/CSC_591C/smoha...
COUNTER NAME: Default
FUNCTION NAME: MPI_Init()
1.75%   mean
1.75%   n,c,t   0,0,0

Function Data Window: pprof.dat/pi/examples/tau-2.12.5/TAU/CSC_591C/smohan/home/
COUNTER NAME: Default
FUNCTION NAME: f() double (double)
31.64%  mean
31.64%  n,c,t   0,0,0

jRacy Main Window: pprof.dat/pi/examples/tau-2.12.5/TAU/CSC_591C/smohan/home/
COUNTER NAME: Default
Mean
n,c,t   0,0,0

Group Ledger Window: pprof.dat/pi/examples/tau-...
COUNTER NAME: Default
TAU_DEFAULT
TAU_USER
MPI

Function Ledger Window: pprof.dat/pi/exampl...
COUNTER NAME: Default
MPI_Attr_get()
MPI_Attr_put()
MPI_Comm_rank()
MPI_Comm_size()
MPI_Finalize()
MPI_Get_processor_name()
MPI_Init()
MPI_Keyval_free()
MPI_Reduce()
MPI_Wtime()
f() double (double)
main() int (int, char **)
main-init() int (int, char **)

Function Data Window: pprof.dat/pi/examples/tau-2.12.5/TAU/CSC_591C/smoha...
COUNTER NAME: Default
FUNCTION NAME: main() int (int, char **)
66.55%  mean
66.55%  n,c,t   0,0,0
```

*The above snapshots (from **jRacy**) show the function-call behaviour of the MPI-based pi program. It shows the time spent in each function, as well as the number of calls made to each function.*

Using the above diagram, and the statistics it provides, we may be able to obtain information about performance bottlenecks, and see how improvements can be made to the program.

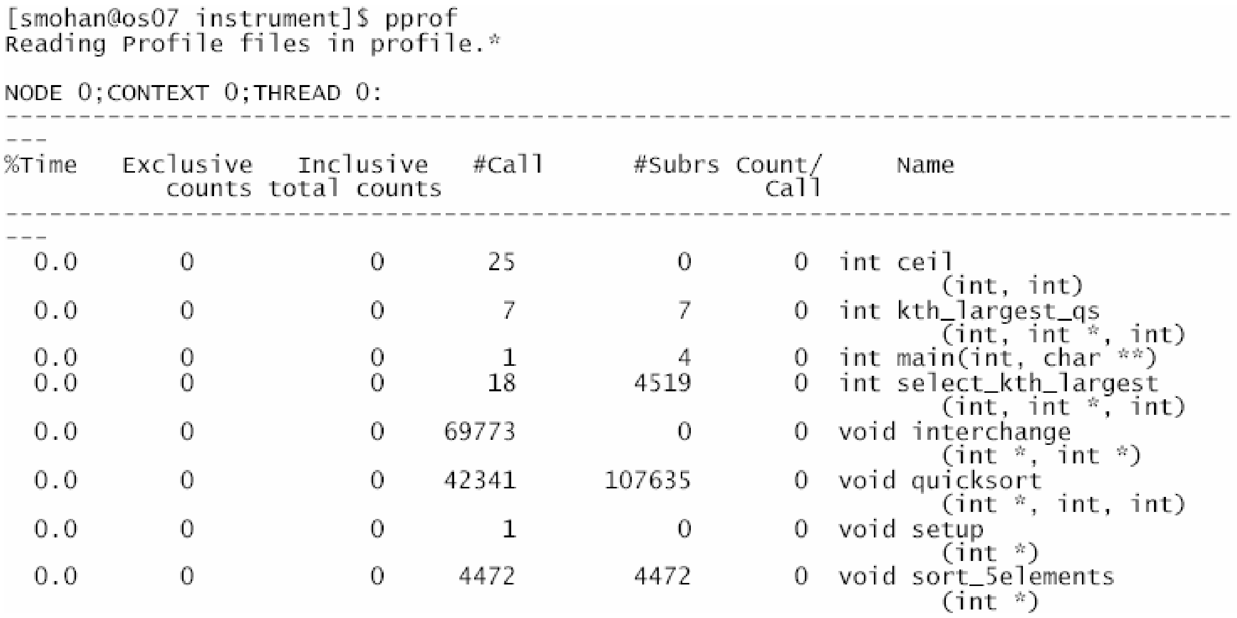
*The profiling results for the above program using **pprof** are :*



```
[smohan@os07 pi]$ pprof
Reading Profile files in profile.*

NODE 0;CONTEXT 0;THREAD 0:
---------------------------------------------------------------------------------
---
%Time Exclusive  Inclusive     #Call      #Subrs  Inclusive Name
           msec  total msec                       usec/call
---------------------------------------------------------------------------------
---
100.0       653        981         1 1.00001E+06   981782   main() int
                                                                 (int, char **)
 31.6       310        310     1E+06           0        0   f() double
                                                                 (double)
  1.8        17         17         1           8    17218   MPI_Init()
  0.0     0.272      0.272         6           0       45   MPI_Wtime()
  0.0     0.109      0.272         1           3      272   main-init() int
                                                                 (int, char **)
  0.0     0.162      0.162         1           0      162   MPI_Get_processor_name()

  0.0     0.043      0.043         1           4       43   MPI_Finalize()
  0.0     0.019      0.019         3           0        6   MPI_Reduce()
  0.0     0.006      0.006         4           0        2   MPI_Attr_get()
  0.0     0.003      0.003         4           0        1   MPI_Attr_put()
  0.0     0.001      0.001         1           0        1   MPI_Comm_size()
  0.0         0          0         1           0        0   MPI_Comm_rank()
  0.0         0          0         4           0        0   MPI_Keyval_free()
```

- *Finding the **kth-largest** element in an array of integers in two ways.*

  We show only the results from pprof for this program.



```
[smohan@os07 instrument]$ pprof
Reading Profile files in profile.*

NODE 0;CONTEXT 0;THREAD 0:
---------------------------------------------------------------------------------
---
%Time    Exclusive   Inclusive    #Call      #Subrs Count/      Name
            counts total counts                      Call
---------------------------------------------------------------------------------
---
  0.0        0            0        25           0       0  int ceil
                                                                 (int, int)
  0.0        0            0         7           7       0  int kth_largest_qs
                                                                 (int, int *, int)
  0.0        0            0         1           4       0  int main(int, char **)
  0.0        0            0        18        4519       0  int select_kth_largest
                                                                 (int, int *, int)
  0.0        0            0     69773           0       0  void interchange
                                                                 (int *, int *)
  0.0        0            0     42341      107635       0  void quicksort
                                                                 (int *, int, int)
  0.0        0            0         1           0       0  void setup
                                                                 (int *)
  0.0        0            0      4472        4472       0  void sort_5elements
                                                                 (int *)
```

Future Steps :

- *To configure TAU to work with OpenMP programs and hybrid MPI-OpenMP programs. The former still has a few minor problems to be ironed out, whereas the latter hasn't been attempted yet.*

Project Topic – ParBenCCh

Solved Issues :

- *Installation and configuration of ParBenCCH.*
- *Minor issues in installation of ParBenCCh still remain.*

Future Steps:

- *Complete Installation of ParBenCCH.*
- *Run various benchmark tests provided along with package.*
- *Configure TAU to evaluate this benchmark.*
- *Suggest improvements to benchmark to remove performance bottlenecks.*
- *Suggest improvements/new features for TAU.*

Work Distribution:

All three of us were involved in all parts of the project. We split up only the following:

- *Configuration & Installation specifics of TAU – Sibin.*
- *Configuration & Installation specifics of ParBenCCH – Salil.*
- *Configuration, Execution & result evaluation of Test programs – Harini.*
- *Documentation and reports – Salil & Harini*
- *Web-page maintenance - Sibin*