Performance Analysis and Tuning with TAU

Kevin James Edwards

kjedward@ncsu.edu

CSC 548 – Parallel Systems

North Carolina State University

Dr. Frank Mueller

1. Progress

My initial area of focus for Tuning and Performance Analysis with TAU was to install the tool suite on our cluster and try to use it to gather some statistics about some MPI and OpenMP programs. I have been able to install TAU on our cluster but have not been able to use it to gather any statistics as of yet. I have tried to use it for a number of programs but have not exactly figured out why I cannot use it to gather statistics. My understanding is that once everything is setup correctly I can simply execute any MPI program and TAU will create profile trace files for each node that can be used later to analyze. This however is not yet happening and I am trying to determine the reason why.

Some of the tutorials I have used to determine how to install TAU have included information about Program Database Toolkit (PDT). My understanding is that this toolkit is used to help analyze a program and helps TAU perform its runtime analyze. At first I did not believe this toolkit was necessary but after continued reading I now believe that it is required. Unfortunately PDT is a fairly large application and I did have some space limitations that were holding me back. These space problems have been resolved and I am currently working towards building PDT and figuring out how it is the missing piece.

In the meantime I have also done some more research into potential benchmarks I can focus on for improving performance. I have seen many papers on the SWEEP3D[4] benchmark that purpose inefficiencies in the code.[2] This was my first focus since research has already been done with this benchmark. However, I am having problems building the benchmark on our cluster. At first I had problems because it is written in FORTRAN but I was able to get past that issue. Currently I am getting an error about a missing file however it does not exist in the code I have downloaded.

I have also looked into other potential benchmarks to work with. Since TAU needs extensions to be able to analyze MPI programs, I would like to also look at some OpenMP based benchmarks. The NAS Parallel Benchmarks are also available in an OpenMP version and I think these will be a good place to start. I am currently working on building these benchmarks but have put more focus on other items previously mentioned.

Since I still am unable to appreciate what TAU can do as far as analyzing parallel programs I am unsure as to which benchmarks I will ultimately focus on. My planned approach is to once I get TAU working correctly I will try it with a few simple benchmarks that I wrote earlier in the semester. These include implementations of matrix multiplication and the calculation of pi. Once I get a better understanding of TAU's capabilities I will then try it with larger benchmarks I have previously mentioned. I would like to work with the SWEEP3D benchmark simply because of the previous research done on some inefficiencies but I am not sure that I will be able to see these with TAU without using Vampir. Since this extension, Vampir, is not free and only available as a trial version I am not sure that I will pursue this area. I may try improving efficiency on some of the small benchmarks I have written earlier in the semester and then move on to larger benchmarks based upon my findings using TAU. Unfortunately is it too premature in my research to determine exactly which benchmark will be my focus.

2. Future Work

In the coming weeks my main focus will be on getting TAU functional for multiple benchmarks. This will require installing PDT correctly and then proceeding from there. This step is crucial and will serve as a very imminent goal for myself. I must get TAU working on some benchmarks before I can proceed any further.

Once I have TAU installed and functional I will begin using it to analyze some benchmarks. I will first focus on simple programs such as the OpenMP pi calculation I wrote earlier in the semester. This work will help me determine all of the ways TAU is capable of helping tune a parallel program. I would like to find a way to improve my code from earlier in the year based upon the TAU analyze of it. This experience with a simpler program will be very important when I tackle something on a larger scale. Additionally, if I am unable to improve performance with a larger benchmark, this work may need to become the focus of my research.

Once I am able to use TAU on simple benchmarks I will turn to larger ones. I will first focus on those which use OpenMP because my understanding is that they will work better with TAU as is. Therefore my first larger benchmarks to work with will be the OpenMP versions of the NAS Parallel Benchmarks. There are many individual benchmarks within this suite therefore it will be hard to determine which is most suitable if any.

If I am unable to find a benchmark that looks like a candidate for improvement I will then turn to the SWEEP3D benchmark. This is a prime candidate for improving efficiency because previous research has been done by others on this particular benchmark. My only worry is that I will be unable to see these areas for improvement without using Vampir, which is not free. Therefore I would like to work with OpenMP programs first before turning to those which use message passing via MPI.

Ultimately I would like to find a benchmark that may have inefficiencies due to either poor usage of cache space or inefficient message passing. We have learned in class of many ways to improve cache usage via modifying loops within code, blocking and other techniques. However this may not be possible to implement such that performance improves for any cluster the benchmark is run on. Therefore I may have to become very specific towards our cluster such that I will customize a particular benchmark for the cache size on our nodes. I will need to find a benchmark that performs poorly on our cluster due to many cache misses. I will then need to determine for this candidate which area of the worker node code could be enhanced via some improved cache usage technique. This may end up being the best approach to this project given the amount of time left in this semester.

Ultimately I feel the hardest part of this project will be determining a suitable benchmark for improving performance. With so many benchmarks available it will be difficult to determine which actually has a minor flaw that may be approved upon. This is why I might have to either work with a simple benchmark such as matrix multiplication or focus on increasing performance for our cluster only. I also feel like implementing Stassen's matrix multiplication algorithm may be suitable for this project. Until I determine a main benchmark for the focus of my research I will not be able to progress further in the project.

3. Project Timeline

The following is a timeline for the remaining weeks of this project. Obviously my first goal is to get TAU working on our cluster and see some reports on some benchmarks. Then my focus will turn to determining a path to follow in my research including which benchmark I chose to work with. These two items are extremely important because until I have achieved these goals I cannot progress further in the project.

Week	Goal
1	Have TAU installed and have run it on a few benchmarks. Begin thinking about which benchmarks would be appropriate for focus.
2	Work on improving a small benchmark more efficient using some techniques we have learned in class and use this information towards determining a larger benchmark for the main focus.
3	With larger benchmark known, work on making more efficient.
4	Formulate results into final report.

4. References

[1] Tuning and Analysis Utilities (TAU), http://www.cs.uoregon.edu/research/tau/home.php, 2006

[2] Samit Jain ,Matthew Crocker, Nasim Mahmood and James C. Browne. Productivity and Performance Through Components: The ASCI Sweep3D Application

[3] Jaydeep Marathe, Anita Nagarajan, Frank Mueller. Detailed Cache Coherence Characterization for OpenMP Benchmarks. ACM. 2004.

[4] ASCI Purple Benchmarks. http://www.llnl.gov/asci/purple/benchmarks/limited/code_list.html, 2006