15-740 Project Milestone: Virtual Application Profiler

Sven Stork
svens@cs.cmu.edu

Anthony Gitter
agitter@cs.cmu.edu

November 15, 2009

1 Status

We are pleased to report that our project has been proceeding according to schedule and that we are on track to achieve our proposed 100% goals. We have not had problems working with the external software resources required for our project (Pin and SQLite) and have all resources needed to complete our project.

Our initial 100% goal was intentionaly open-ended in terms of what specific analysis we intended to implement. After meeting with Dr. Mowry and discussing what types of analysis would be most interesting, we have refined our goals. Specifically, we will no longer emphasize the types of analysis that can be performed with pure SQL queries because this feature is largely architecture-independent. Instead, we will focus our replay and analysis on thread interactions in a multicore system. This will ensure that our profiler and evaluation are centered on issues that arise in parallel programs on multicore machines as opposed to profiling application behavior in a manner that is not tied to the underlying architecture.

The only major surprise we have encountered thus far has been the size of our SQLite database logs. We believe this problem to be manageable because we modified our profiler to support dynamic trace enabling/disabling (details below). As a new 125% goal, we could explore more sophisticated solutions that have been employed in related work, such as Netzer's transitive reduction [1].

2 Accomplishments

We implemented a Pin tool that can collect information such as memory accesses, memory allocation, function calls, and virtual time and log it in a SQLite database. Furthermore, basic profiling and replay functionality is in place and we are able to use SQL queries to play back the logged data. As a proof of concept, we have adapted our cache simulator from the first assignment to work with our profiler's replay feature.

To address the aforementioned issue of log size, we modified the profiler so that the programmer can selectively decide what information should logged and at which points in the application it should be logged. Tracing is enabled and disabled by a simple function call to our library function.

The work we have completed achieves the 75% goal we set in our project proposal, which was also the goal we had set for our milestone progress.

3 Revised Schedule

For the next week, Sven will focus on modifying our profiler to log and replay additional information needed to analyze multicore thread interactions, such as locking. Anthony will primarily concentrate on developing the algorithms used for the analysis of this data. In the final week, we will both evaluate our system on test applications, prepare the poster, and write the final report.

References

[1] R.H.B. Netzer. Optimal tracing and replay for debugging shared-memory parallel programs. In *Proceedings of the 1993 ACM/ONR workshop on Parallel and distributed debugging*, pages 1–11. ACM New York, NY, USA, 1993.