CSC 548 Project: Memory Trace Compression using Extended PRSDs – Report 2

Milestones

Task	Status	Previous Estimated Completion Date	Estimated Completion Date
Generation of memory trace	Completed	-	-
Generating RSDs	Completed	-	-
Detecting loops and Generating EPRSDs for individual threads	Completed	Nov 1	Nov 2
Change memory allocation scheme, remove compare function, add new template parameters	In Progress	*<new addition>*	Nov 15
Merging EPRSDs across threads using MPI program	Pending	Nov 8	Nov 20
Testing of Memory trace compression tool	Pending	Nov 15	Nov 23
Final Report	Pending	Nov 22	Nov 25

Problems

A template library is being developed to compress memory traces. There are some modifications required in the template library design as follows.

1. EPRSD Compare function was stored as a function pointer. There was a lot of function call overhead each time EPRSD comparison was made.
   Solution: This is being changed to be an inline static method inside the MATCH class. Inlining this function helps to avoid function call overhead.

2. Signature (common part of EPRSD) comparison was done by user whereas it should be handled by the template library.
   Solution: Added function to template library so that signature comparison is done internally and user is allowed to compare only the user-defined part of EPRSD. This comparison is optional and user can simply return TRUE if nothing needs to be compared in user-defined part. Both common part comparison and user-defined part comparison should hold TRUE for a matching EPRSD.

3. EPRSDCompressor class needs to use classes for matching and merging operations.
   Solution: Two additional template parameters are added to EPRSDCompressor class to support match and merge functionality. A class called MATCH implements an EPRSD matching/compare function. MERGE class implements a callback method used to notify user of the merge operations. Both of these functions are static and inline.

4. Memory allocation burden was on the user side. User needs to manage allocation and release of memory which is cumbersome.
   Solution: User passes the memory trace data to template library which in turn allocates memory dynamically and copies the user data. Template library takes care of releasing the memory when EPRSDs are merged or written to file.

Project Web page

http://www4.ncsu.edu/~sbudanu/CSC548_Project/

References

[1] P. Ratn, F. Mueller, Bronis R. de Supinski, Michael Noeth and M.Schulz, ScalaTrace: Scalable Compression and Replay of Communication Traces for High Performance Computing , Journal of Parallel and Distributed Computing, accepted Sep 2008, pages 1-14.

[2] Prasun Ratn, M.S. Thesis, Preserving Time in Large-Scale Communication Traces , North Carolina State University, Aug 2008.

[3] J. Marathe F. Mueller, T. Mohan, S. McKee, B. de Supinski, A.Yoo, METRIC: Memory Tracing via Dynamic Binary Rewriting to Identify Cache Inefficiencies, ACM Transactions on Programming Languages, Vol. 29, No. 2, Apr 2007, pages 1-36.

[4] M. Noeth and F. Mueller and M. Schulz and B. de Supinski Scalable Compression and Replay of Communication Traces in Massively Parallel Environments, P=ac2 Conference, IBM T.J. Watson, Oct 2006.

[5] J. Marathe, F. Mueller, T. Mohan, B. R. de Supinski, S. A. McKee and A. Yoo METRIC: Tracking Down Inefficiencies in the Memory Hierarchy via Binary Rewriting, International Symposium on Code Generation and Optimization, Mar 2003, pages 289-300.

[6] Pin binary instrumentation tool - http://www.pintool.org