lRVM: light weight Recoverable Virtual Memory

Machiry Aravind Kumar Prabhavathy Viswanathan

Georgia Institute of Technology
{amachiry,pviswanathan6}@gatech.edu

1. How to compile your library?
   (a) make clean.
   (b) make all.
   (c) gcc <your_test_file> librvm.a (make sure that you include "rvm.h" in the test file or copy rvm.h to /usr/include)
2. Any thoughts you have on the project, including things that work especially well or which don't work.

   There has to be bakeoff kind of competetion at the end of the project. This will motivate students to write good and performent code. As of now, most of the students just write code which works for base cases. Problems like : memory leak, corner cases are not taken in to consideration, but these need to be fixed for any system code.
3. How you use logfiles to accomplish persistency plus transaction semantics?
   - Ensuring transaction semantics: When a region of the memory is about to be modified, we copy the region in to in memory backup area. When a transaction is aborted we copy the contents from the backup area to the corresponding memory region which ensures that all the changes are reverted back. On the other hand, if the transaction is committed, we copy the contents of the memory region to the log file in the following format:

     ```
     STARTTRANSACT:<transactionID>
     <for each region>
     SEGNAME:<segmentName>
     OFFSET:<offset>
     DIFFS:<sizeofthemodifiedregion>
     <data in raw bytes of DIFFS size>
     ENDTRANSACT
     ```
   - Persistency is achieved by recording all the modifications to the on disk logfile (lrvmlog), which will be stored in the same directory as the one given for rvm_init.
4. How many files do you have? What goes in them?

   We maintain a file for each segment, which stores the persistent contents of the corresponding segment and 2 metadata files:
   - lrvmlog : This is the log file that contains all the persisted modifications of various transactions.
   - lrvmSEGINFO : This file contains information of various segments along with their size.
5. How do the files get cleaned up, so that they do not expand indefinitely?

   As mentioned before, we maintain only 2 files for our metadata: lrvmlog and lrvmSEGINFO.

- lrvmlog gets cleaned up in 2 ways: First, when a segment is mapped, we consume all the modifications for the segment from the log file and second from explicit call from user to truncate the log: rvm_truncate_log.
- lrvmSEGINFO gets cleaned up when an existing segment is deleted by using : rvm_destroy

6. How to run tests?
   (a) Compile the sources to get librvm.a
   (b) Copy librvm.a to Tests folder.
   (c) cd to Tests folder
   (d) run python tester.py