Address Stream Profiling

Charles Drews (csd305@nyu.edu, N11539474)

May 4, 2014

Project Phase 3: User Manual

Virtual Machines: Concepts and Applications, CSCI-GA.3033-015, Spring 2014

Compiling:

My source code is in "profiler.c". It can be compiled simply with "`gcc -o profiler profiler.c`". During development I used "`gcc -Wall -Wextra -o profiler profiler.c`" so I could be sure to resolve all compiler warnings. I also used "`valgrind --leak-check=full ./profiler input.txt`" to find and eliminate all memory leaks. I compiled and tested on both my local machine and on a CIMS machine.

Running the Profiler:

The usage is "`./profiler <path/input_file> [-a]`", where "`-a`" is an optional flag that affects the length of the output. If "`-a`" is omitted, then the output ignores blocks which were executed fewer than five times, and in the edge list for indirect jumps, edges are ignored if they are taken less than 10% of the time. If "`-a`" is included in the call to profiler, then all block and edges will be included in the output. The output files will be created in the same directory which contains the input file.

The following are examples of correct and incorrect usage:

- Correct: `./profiler trace.txt -a`
- Correct: `./profiler inputs/trace.txt`
- Incorrect: `./profiler -a trace.txt`
- Incorrect: `./profiler trace.txt -A`

Input File:

The profiler tool expects the input file to be a memory trace created by the Valgrind Lackey tool. If the program to be profiled is called "program1" and its usage is "`./program1 arg1 arg2`" then please use "`valgrind --log-fd=1 --tool=lackey --trace-mem=yes ./program arg1 arg2 > trace_file.txt`" to generate the necessary memory trace and save it in "trace_file.txt". Then use "trace_file.txt" as input when calling the profiler. The input file for the profiler tool does not need to have any particular file extension; the above use of ".txt" is simply an example; it could just as easily be "trace_file" with no extension. What matters is that the input file for the profile is the output from the Valgrind Lackey tool.

Output Files:

The profiler tool creates two output files, both with names based on the input file name. One output name will be "_SBB_profile.txt" appended to the input file name (after removing any extension from the input file name), and the other will be similar with "_DBB_profile.txt" appended. The former contains the profile based on static basic blocks, and the latter contains the profile based on dynamic basic blocks. Each output also contains a title that specifies whether it is showing static or dynamic basic blocks.

The top of each output file is a title followed by a detailed description of the format of the profile. Then the body of the output file consists of one row for each basic block that was identified by the profiler. The fields of each row provide all the profile statistics for each basic block, and for the outgoing edges from the basic block. What follows is the detailed description which also appears in the output files themselves, which defines the fields present in the body of the output:

```
Profile elements:
ID = unique block identifier number
start = address of block's starting instruction
#I = number of instructions in the block
end = address of block's ending instruction
#E = number of times the block was executed
desc = description of this block's ending instruction:
        - NB = Not a Branch (falls-thru every time)
        - UB = Unconditional Branch
        - CB = Conditional Branch
        - IJ = Indirect Jump (multiple targets)
[TID:#T] = list of this block's outgoing edges
TID = Target ID, the ID of each edge's target block
#T = number of times each edge was taken
(ft) = this edge is a fall-thru to the next block
...  = some IJ edges not shown (use -a to show all)
```

The order of the fields, also provided in the output files themselves, is as follows:

```
ID: start-#I-end #E desc [ TID:#E ]
```

Here is a sample output row, followed by a description of what it indicates:

```
1442:  0x52e789c-10-0x52e78bb 359 IJ [ 2223:38 2148:76 2074:189 ...  ]
```

This row indicates that the profiler identified a block starting with the instruction at address `0x52e789c`, which includes 10 instructions, and ends with the instruction at address `0x52378bb`. This block was given the ID 1142 by the profiler, and was executed 359 times during the creation of the address stream. The block's ending instruction is an indirect jump (IJ). Control flow was passed to the block with ID 2223 on 38 occasions. Control was also passed to block 2148 on 76 occasions and to block 2074 on 189 occasions. The "..." indicates that control was also passed to other blocks, but that the frequency was less than 10% of the time, and therefore those edges were not shown. If the "`-a`" option had been included when calling the profiler, then all edges would be included in the bracketed list, and "..." would not appear.

One more example:

```
2890:  0x535bc70-4-0x535bc7d 87 CB [ (ft)2891:11 2889:76 ]
```

This row indicates that block 2890 includes 4 instructions, from `0x535bc70` to `0x535bc7d` , and was executed 87 times. This block ends with a conditional branch, which was taken 76 times (passing control to block 2889) and not taken 11 times (with control falling through to block 2891).

As described above, if the "`-a`" option is omitted when calling the profiler, then to simplify the output and make it more human-readable, blocks executed fewer than five times are excluded from the output, and the edge list for indirect jumps excludes any edges that were taken less than 10% of the time. If the "`-a`" option is included, then all blocks and edges are included, which can make the output significantly longer.