Enhancing the Performance of the DynamoRIO Memtrace Client for Memory-Intensive Workloads and Exploring Last-Level Branch Prediction

Gandholi Sarat - 23008

Guide: *Dr. R. Raghunatha Sarma*

Mentors: *Dr. Shubhankar Suman Singh , Mr. M. Naveen*

April 14, 2025

Introduction

Dynamic binary instrumentation (DBI) tools such as DynamoRIO is essential for analyzing software, providing valuable insights into how programs behave while they're running.

Aim

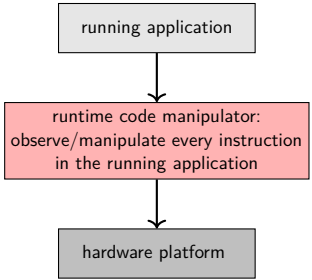
- To enhance the performance of the DynamoRIO Memtrace client for memory-intensive workloads.
- To explore the concept of Last-Level Branch Prediction (LLBP) and its potential benefits in improving the Branch Prediction.

Methodology

- Understanding Current tracing client
- Selecting and understanding the Benchmark
- Look for some optimizations
- Exploring LLBP

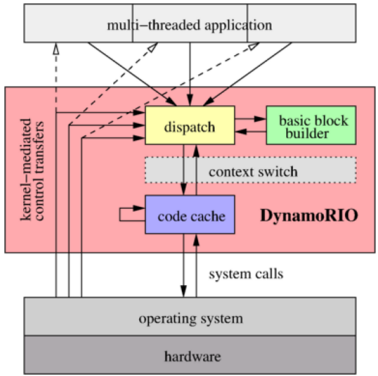
DynamoRIO's Architecture

- **DynamoRIO:** A runtime code manipulation system
- **Core features:** Efficiency, transparency, and comprehensive control
- **Functionality:** Intercepting and modifying every executed instruction



DynamoRIO's Architecture

- **Core operation:** Shifting execution to a specialized code cache
- **Dynamic basic block management:** Incremental copying of application code
- **Trace formation:** Amalgamation of frequently executed basic block sequences
- **Indirect branch resolution:** Techniques like inlined table lookups or comparisons



Understanding Memtrace

- Working of the code
- Trace format
- Buffering techniques used
- Scope for improvement

Improvements Made

- Change to `dr_fprintf`
- Using binary format rather than text
- Changing buffer size
- Using offset in the address
- Compressing the output data before writing to file

Change of Bandwidth with Tracing

S. No	Trace Enabled	Trace File Format	Copy (MB/s)	Scale (MB/s)	Add (MB/s)	Triad (MB/s)	Avg Rate (MB/s)	Trace File Size (GB)	Write BW (MB/s)	Time (Sec)	Total (MB/s)
1	No	-	20007.7	19543.4	23354.9	23444.5	21587.63	—	0	1	21587.63
2	Yes	Text	7.2	3.8	5.6	5.7	5.57	53.4	24.37	2244	29.9
3	Yes	Binary	658.4	425.6	649.9	676.1	602.5	50.3	2452.7	21	3055.22

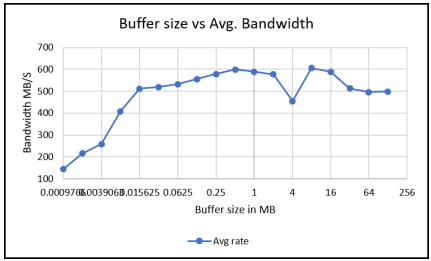
Change of Bandwidth with Tracing

S. No	Trace Enabled	Trace File Format	Copy (MB/s)	Scale (MB/s)	Add (MB/s)	Triad (MB/s)	Avg Rate (MB/s)	Trace File Size (GB)	Write BW (MB/s)	Time (Sec)	Total (MB/s)
1	No	-	20007.7	19543.4	23354.9	23444.5	21587.63	—	0	1	21587.63
2	Yes	Text	7.2	3.8	5.6	5.7	5.57	53.4	24.37	2244	29.9
3	Yes	Binary	658.4	425.6	649.9	676.1	602.5	50.3	2452.7	21	3055.22

85.85% Performance Loss from No Trace to Binary Trace

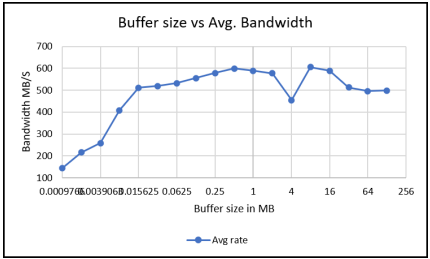
Buffer Performance Table

Buffer Size (MB)	Copy	Scale	Add	Triad	Avg rate (MB/Sec)	Time (Sec)
0.000976563	160.7	107.8	162.9	158.7	143.8	80
0.001953125	244.1	160.5	243.5	237.8	216.03	54
0.00390625	373.8	230.9	169.4	174.7	258.03	114
0.0078125	454.4	305.6	464.5	461.7	408.17	30
0.015625	566.9	385.2	579.4	582.8	510.5	25
0.03125	590.9	390.1	576.4	642.5	519.13	23
0.0625	586.8	407.7	604.1	690.5	532.03	28
0.125	621.1	429.4	614.2	596.1	554.9	27
0.25	658.4	425.6	649.9	676.1	577.97	21
0.5	654.1	466.9	675.4	655.6	598.8	19
1	642.4	451.5	671	681.5	588.3	21
2	668	420.7	643	647	577.23	32
4	515.2	339.4	508.8	493.2	454.47	29
8	689.7	448.6	676.5	678.7	604.93	28
16	676.7	438.2	652.3	641.2	589.07	28
32	577.4	380.4	578.9	559.1	512.23	26
64	560.7	356.2	562.8	543.4	496.23	30
128	563.3	359.8	569.6	554.4	497.57	25



Buffer Performance Table

Buffer Size (MB)	Copy	Scale	Add	Triad	Avg rate (MB/Sec)	Time (Sec)
0.000976563	160.7	107.8	162.9	158.7	143.8	80
0.001953125	244.1	160.5	243.5	237.8	216.03	54
0.00390625	373.8	230.9	169.4	174.7	258.03	114
0.0078125	454.4	305.6	464.5	461.7	408.17	30
0.015625	566.9	385.2	579.4	582.8	510.5	25
0.03125	590.9	390.1	576.4	642.5	519.13	23
0.0625	586.8	407.7	604.1	690.5	532.03	28
0.125	621.1	429.4	614.2	596.1	554.9	27
0.25	658.4	425.6	649.9	676.1	577.97	21
0.5	654.1	466.9	675.4	655.6	598.8	19
1	642.4	451.5	671	681.5	588.3	21
2	668	420.7	643	647	577.23	32
4	515.2	339.4	508.8	493.2	454.47	29
8	689.7	448.6	676.5	678.7	604.93	28
16	676.7	438.2	652.3	641.2	589.07	28
32	577.4	380.4	578.9	559.1	512.23	26
64	560.7	356.2	562.8	543.4	496.23	30
128	563.3	359.8	569.6	554.4	497.57	25



New Bandwidth:

$$604.93 + \left(\frac{50.3\,\text{GB}}{28\,\text{sec}}\right) = \textbf{2444.47 MB/Sec}$$

88.67% Performance Loss with new Buffer Size

Change of Bandwidth with offset

Trace File Format	Copy (MB/s)	Scale (MB/s)	Add (MB/s)	Triad (MB/s)	Avg Rate (MB/s)	Trace File Size (GB)	Write BW (MB/s)	Time (Sec)	Total (MB/s)
Binary	727.0	479.9	715.1	714.7	659.18	50.3	2575.36	20	3234.54

Change of Bandwidth with offset

Trace File Format	Copy (MB/s)	Scale (MB/s)	Add (MB/s)	Triad (MB/s)	Avg Rate (MB/s)	Trace File Size (GB)	Write BW (MB/s)	Time (Sec)	Total (MB/s)
Binary	727.0	479.9	715.1	714.7	659.18	50.3	2575.36	20	3234.54

85.01% Performance Loss from No Trace to Binary Trace

Change of Bandwidth with offset

Trace File Format	Copy (MB/s)	Scale (MB/s)	Add (MB/s)	Triad (MB/s)	Avg Rate (MB/s)	Trace File Size (GB)	Write BW (MB/s)	Time (Sec)	Total (MB/s)
Binary	727.0	479.9	715.1	714.7	659.18	50.3	2575.36	20	3234.54

5.54% improvement from default memtrace client

Compression the data with LZ4

No. of times Buffer Size	Copy	Scale	Add	Triad	Avg rate (MB/Sec)	Time (Sec)
1	496.9	444.2	494.7	490.3	481.525	24
2	365.5	322.2	371.2	363.7	355.65	34
3	423.3	368.2	417.9	415.8	406.3	29
4	498.5	443.7	494.9	491.4	482.125	24
5	495.2	441.8	494.9	488.7	480.15	24
6	487.8	434.7	484.8	479.9	471.8	25
7	488.0	434.9	485.0	480.1	472	24
8	493.9	439.8	493.3	486.8	478.45	24
9	490.8	430.1	492.0	485.7	474.65	24
10	490.4	436.9	490.0	483.3	475.15	24
11	489.5	434.4	488.7	482.6	473.8	25
12	488.3	435.8	489.6	483.0	474.175	24
13	490.8	435.9	487.6	483.1	474.35	25
14	480.3	424.9	478.0	472.4	463.9	25
15	488.2	431.9	486.2	480.2	471.625	24
16	466.7	411.7	464.6	460.7	450.925	26
32	478.0	421.9	477.4	471.5	462.2	25
64	457.1	398.0	455.9	450.7	440.425	26

New Bandwidth:

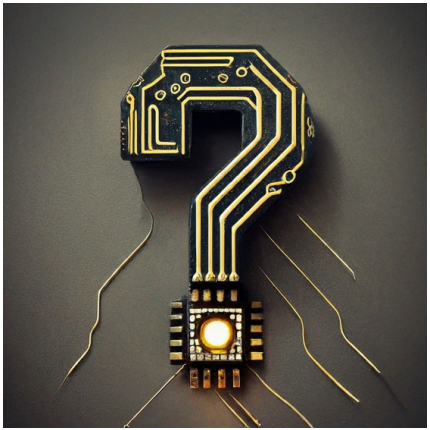
$$481.5 + \left(\frac{7.1\,\text{GB}}{24\,\text{sec}}\right)$$
$$= \mathbf{784.43\,MB/Sec}$$

- **Improvement through offset**:
  - **5.54% improvement from defalut memtrace client**
- **Compression**:
  - Compression is becoming overhead and leading to a lower bandwidth in the application
  - But there is significant reduction in the file size 7.1GB from 50.3GB
  - So there is a trade off here

Future Scope

- Check the current client with other real world benchmarks
- Dynamic Scheduling Strategies

Questions



Thank You

References

- https://dynamorio.org/
- https://groups.csail.mit.edu/cag/rio/derek-phd-thesis.pdf
- https://github.com/lz4/lz4
- https://www.cs.virginia.edu/stream/