Week#4 Monitoring Buffer Miss Scenario in MySQL/InnoDB

Siyeol Choi
2019315530

1. INTRODUCTION

In this lab experiment, we target to measure the ratio of each step of 1) searching the free list 2) scan LRU tail for clean page 3) Flush Single Dirty page to disk. Then investigate which step would be fatal to DBMS performance and take research on its reasons as well.

2. METHODS

This week we modified the source code of the mysql itself so that we could trace the portion of the time consumed during each of the phases. There are three big phases, including searching free list, scanning LRU tail for clean page, and flushing single dirty page to disk.

Inside buf0lru.cc file, there's buf_LRU_get_free_block function which returns a free block from buf_pool by iterating over to conduct all three steps mentioned above. In between each of steps, there are if statements which breaks the loop and restarts the loop.

Three main functions that conducts the utilization of our desired goal are "buf_LRU_get_free_only", "buf_LRU_scan_and_free_block", and "buf_flush_single_page_from_LRU". After each of the function, there's an if statement which assures that the returned value of the functions is not null. The fprintf stderr should be located inside the if statement so that it logs only the cases where it is non-null. In this case, non-null means that the function functioned as desired, so that cases should only be counted.

3. Performance Evaluation

3.1 Experimental Setup

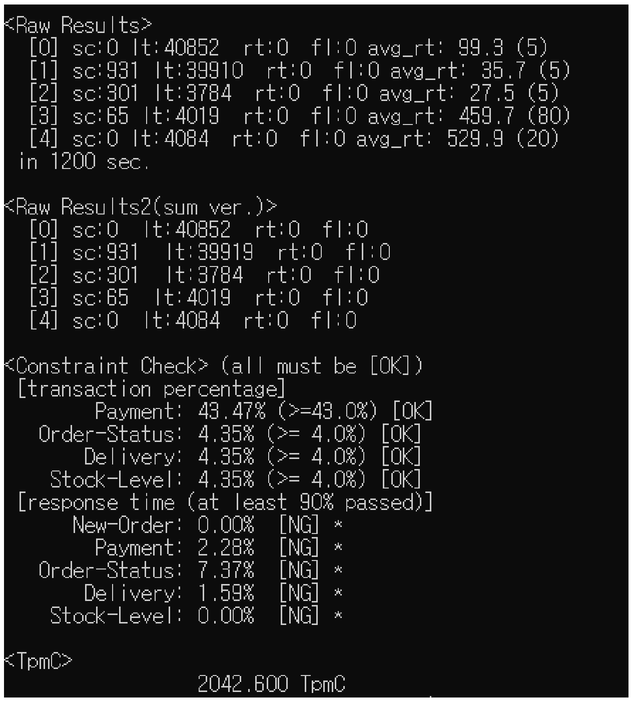
[System Setup]

MySQL version 5.7.33	
OS	Ubuntu 20.04.1 LTS (GNU/Linux 4.19.104-microsoft-standard
CPU model name	Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
cpu cores	4
Vendor:	Msft
Product:	Virtual Disk
Revision:	1.0
Compliance:	SPC-3
User Capacity:	274,877,906,944 bytes [274 GB]
Logical block size:	512 bytes
Physical block size:	4096 bytes

[Benchmark setup]

Type	Configuration
DB size	2GB (20 warehouse)
Buffer Pool	
Size	100MB (5% of DB size)
Benchmark	
Tool	tpcc-mysql
Runtime	1200s
Connections	8

3.2 Experimental Results



```
<Raw Results>
  [0] sc:0 lt:40852  rt:0  fl:0 avg_rt: 99.3 (5)
  [1] sc:931 lt:39910  rt:0  fl:0 avg_rt: 35.7 (5)
  [2] sc:301 lt:3784  rt:0  fl:0 avg_rt: 27.5 (5)
  [3] sc:65 lt:4019  rt:0  fl:0 avg_rt: 459.7 (80)
  [4] sc:0 lt:4084  rt:0  fl:0 avg_rt: 529.9 (20)
 in 1200 sec.

<Raw Results2(sum ver.)>
  [0] sc:0  lt:40852  rt:0  fl:0
  [1] sc:931  lt:39919  rt:0  fl:0
  [2] sc:301  lt:3784  rt:0  fl:0
  [3] sc:65  lt:4019  rt:0  fl:0
  [4] sc:0  lt:4084  rt:0  fl:0

<Constraint Check> (all must be [OK])
 [transaction percentage]
        Payment: 43.47% (>=43.0%) [OK]
   Order-Status: 4.35% (>= 4.0%) [OK]
       Delivery: 4.35% (>= 4.0%) [OK]
    Stock-Level: 4.35% (>= 4.0%) [OK]
 [response time (at least 90% passed)]
      New-Order: 0.00%  [NG] *
        Payment: 2.28%  [NG] *
   Order-Status: 7.37%  [NG] *
       Delivery: 1.59%  [NG] *
    Stock-Level: 0.00%  [NG] *

<TpmC>
                 2042.600 TpmC
```

[Result of grep -c]

Step 1) 4,817,136

Step 2) 552,109

Step 3) 11,299

Total) 5,423,556

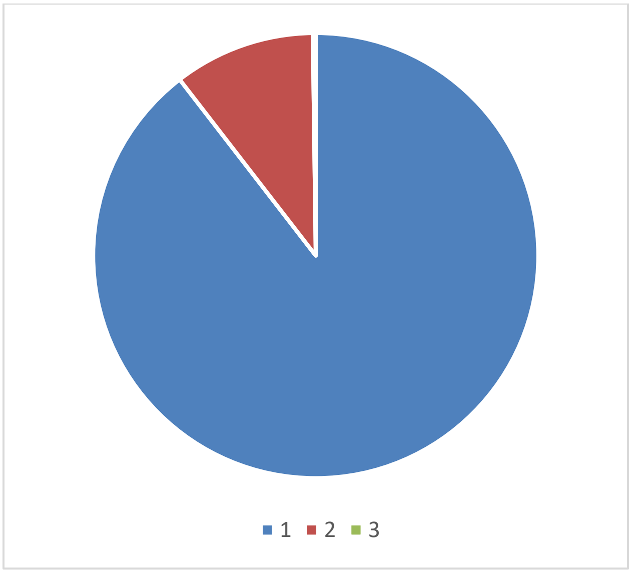
[Percentage]

Step 1) 88.8188 %

Step 2) 10.1798 %

Step 3) 0.2083 %

[Pie Graph]



Step 1, which is getting a free buffer frame in the free list took up the largest portion of the total process. It took 88.82% of the total process, and it means enhancing the performance of getting a free buffer frame in the free list would highly affect the overall benchmark of the total Buffer miss scenario.

The reason why the step 1 consumes up the biggest portion is as follows: Step 2 and Step 3 are counted on the assumption that the no block was in the free list. However, during the initial phases of iterations, it is obvious that there are free blocks in the free list.

Since the case when step2 is called is only when the clean page is found by LRU scan and victim page is founded. Thus, when the scan is successful, so the step3 is the case when the step2 did not found an appropriate victim page. Accordingly the portion of step 3 is comparably very low.

4. Conclusion

Getting a free buffer frame in the free list consists of the largest portion of the whole process of getting a free block of buf_LRU. The portion of getting free buffer frame is ~88.82%, so enhancement of performance during that phase would largely improve the overall performance.

5. REFERENCES

[1] Measure steps https://github.com/LeeBohyun/mysql-tpcc/blob/master/buffer_manager/buffer_miss_scenario_monitoring.md

[2] Report format https://github.com/LeeBohyun/SWE3033-F2022/blob/main/report-guide.md