OBJECTIVE: Implement Write Back Policy in various OLTP benchmarks and measure the performance changes.

We used Cafe library to implement server side changes required for write back policies.
We implemented write back in four benchmarks

1) Smallbank
2) YCSB
3) Voter
4) SIBench

For each benchmarks various classes that has implemented.
BM_name+CacheStore, **BM_name+WriteBack, BM+Test, BM+warmup(Except in Smallbank which has Warmup)**

1) **CacheStore:** - This class is implement CacheStore interface in cafe library which is mainly responsible for implementing cache for CADS.
2) **WriteBack:**- This class is implement WriteBack interface in cafe library which is mainly responsible for implementing all the methods responsible for write back policy.
3) **Warmup**:- This is used to warmup cache before we start measuring . Here, it cached all the account, checking and saving entries in cache using write through policy.
4) **Test:-** This class is responsible for unit test the code. Here, I run all the procedures. And see if they are working correctly or not. Stats object will show how many cache hits and cache misses, number of commited,aborted sessions and various
   Other information. For eg

```
{
        "read_statements": 3,
        "AREA_CODE_STATE": 2,
        "VOTES": 2,
        "bw_size": 319,
        "CONTESTANTS_size": 44,
        "CONTESTANTS": 2,
        "bw_skv_size": 197,
        "latency_AREA_CODE_STATE": 126,
        "commited_sessions": 1,
        "cache_misses": 3,
        "latency_VOTES": 127,
        "write_statements": 2,
        "VOTES_size": 40,
        "AREA_CODE_STATE_size": 41,
        "latency_CONTESTANTS": 5649
}
```

This shows various informations regarding the operations performed. These are used to check whether implementation is successful and perform as per expected.Also Iam printing key if I encounter cache miss to verify if I am getting cache miss even after warmup.

Along with that modify all the procedures of each benchmarks to support this new architecture.

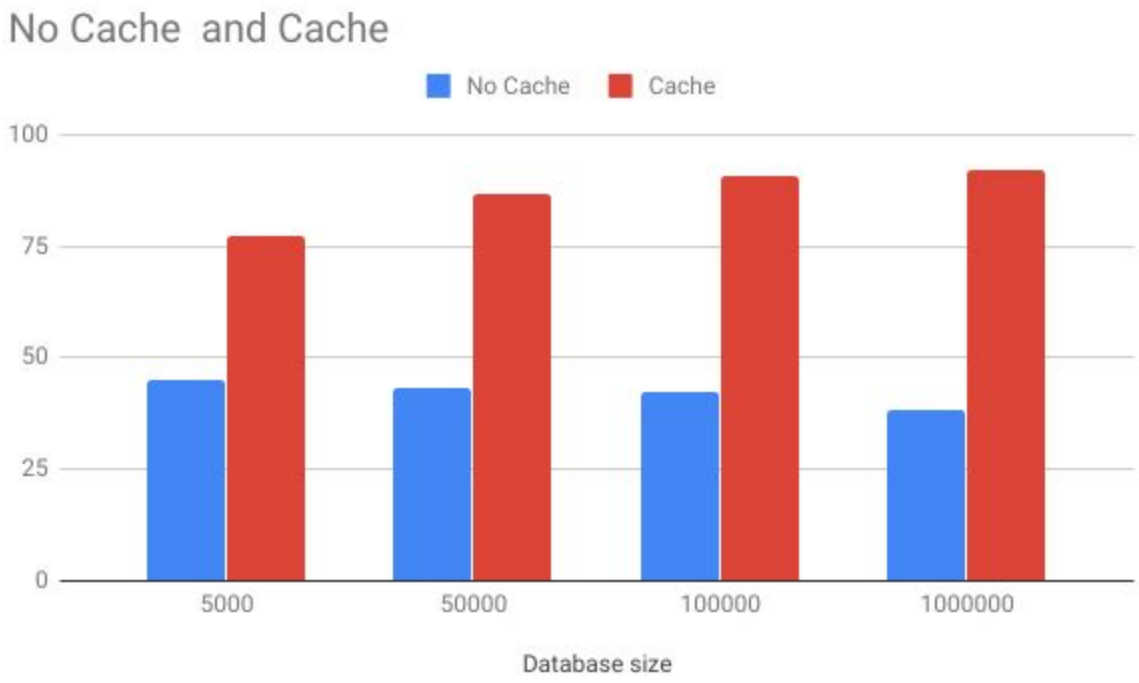
For Measurement purpose I used Azure VM
Specification of VM:
Standard B4ms (4 vcpus, 16 GB memory)
Cache details: memory 10Gb, **max connections 10**, reserved space 10%

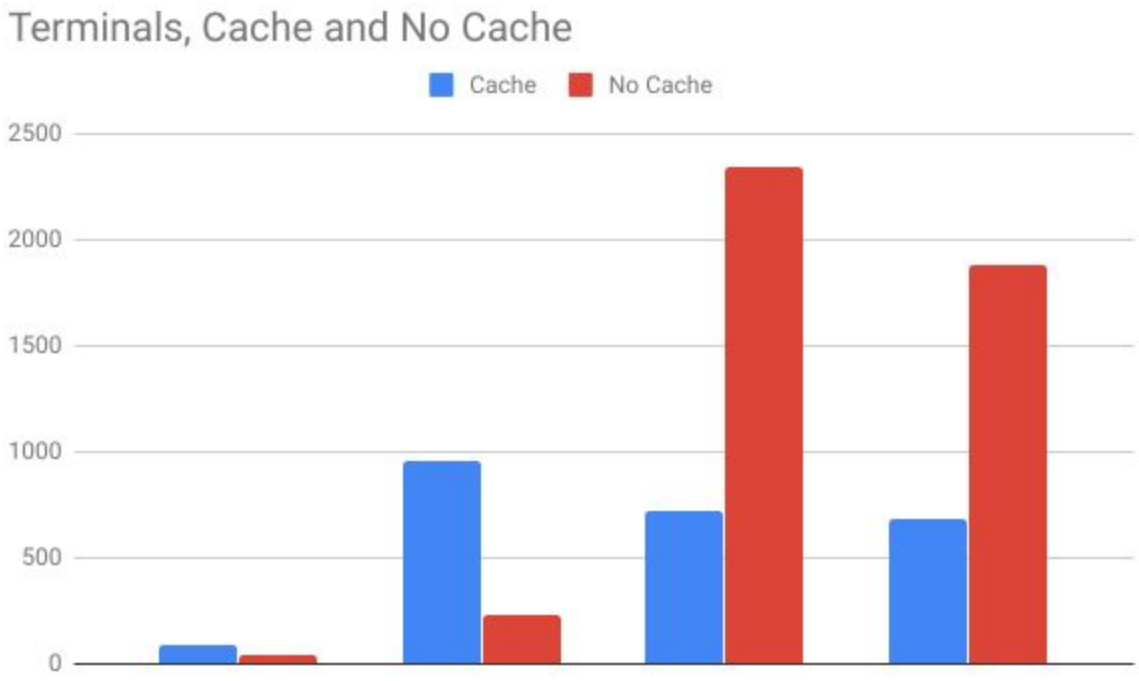
Results:

SmallBank:

Query distribution:

15% Amalgamate
15% Balance
15% Deposit Checking
25% Send Payment
15% Transaction Savings
15% WriteCheck



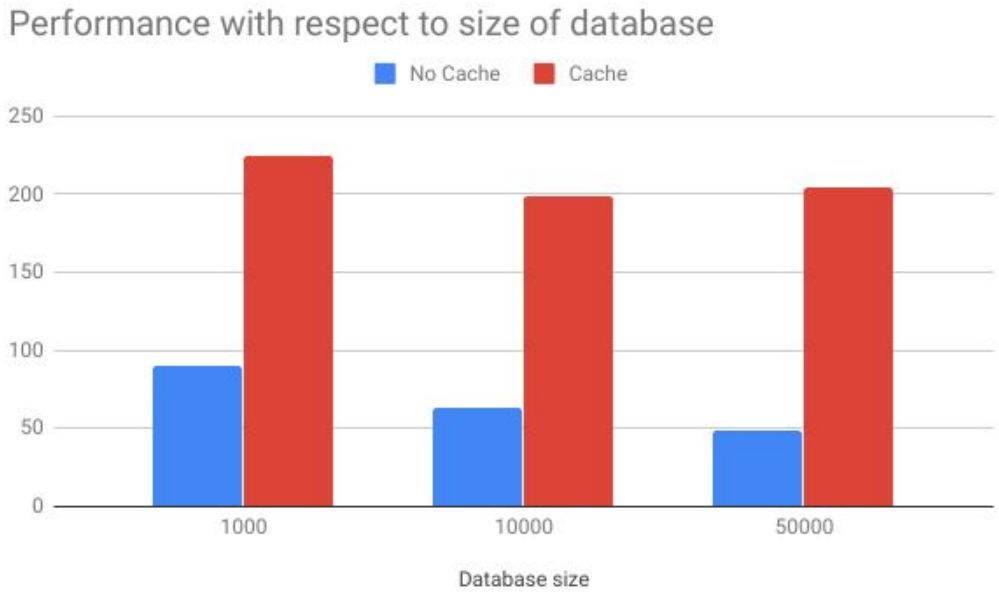


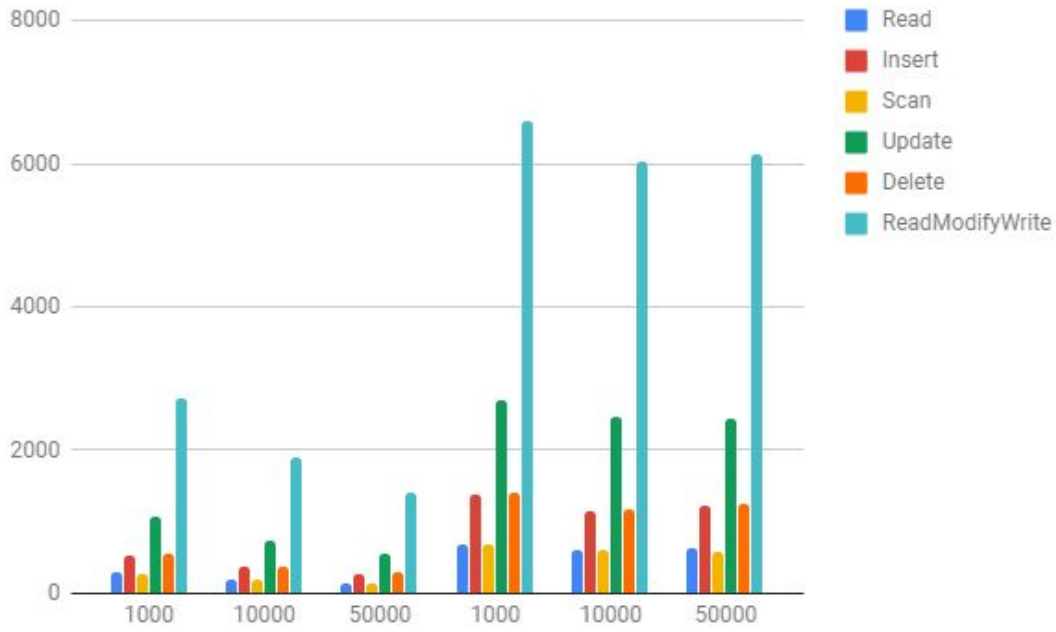
In fig below first four are entries for non cache and last four for where caching was there.

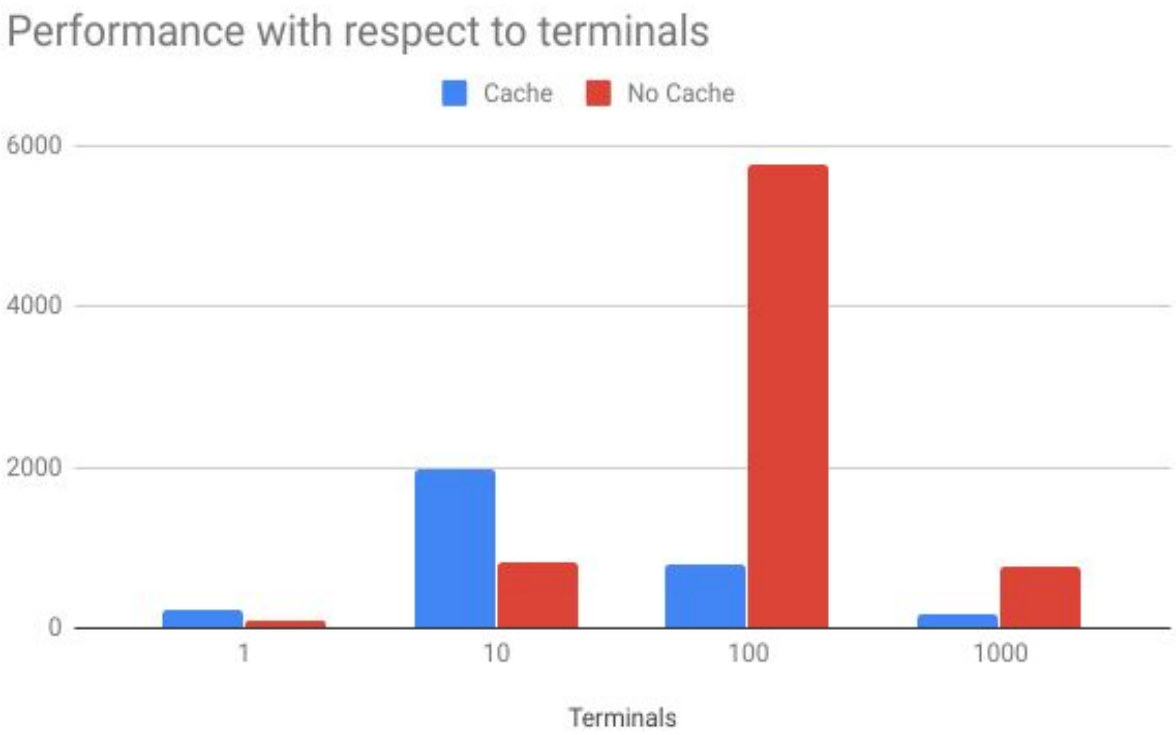
YCSB:

Query distribution:

5% ReadRecord
20% Update
5% Scan
10% Delete
50% RMW(Read Modify write)
10% Insert

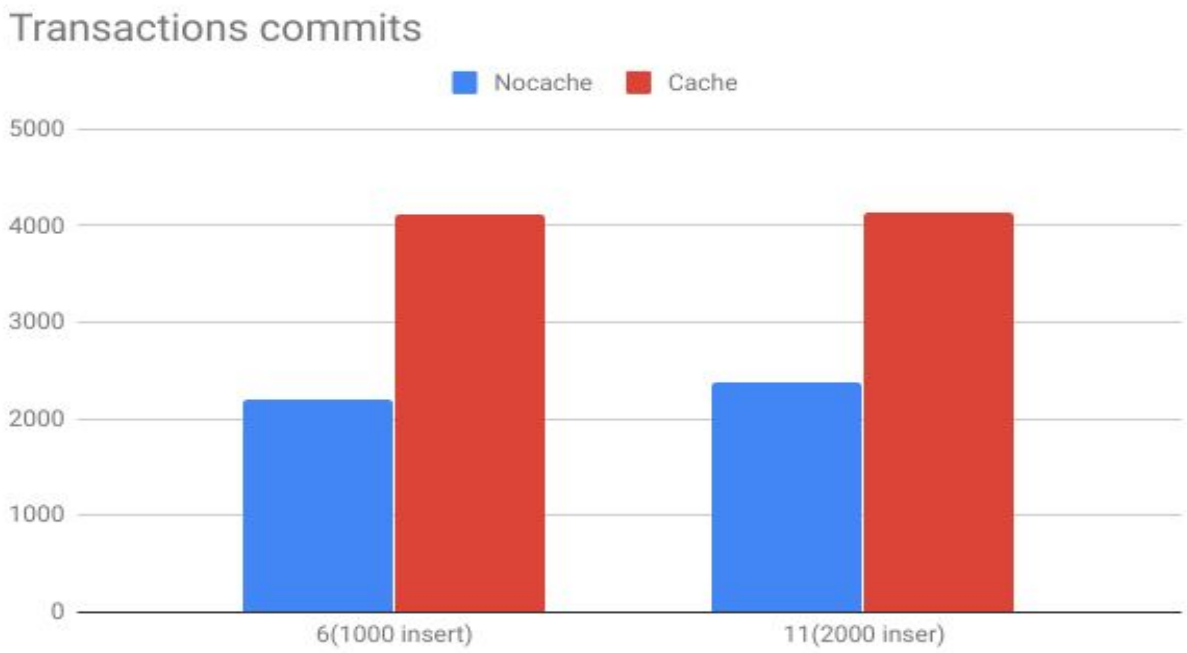




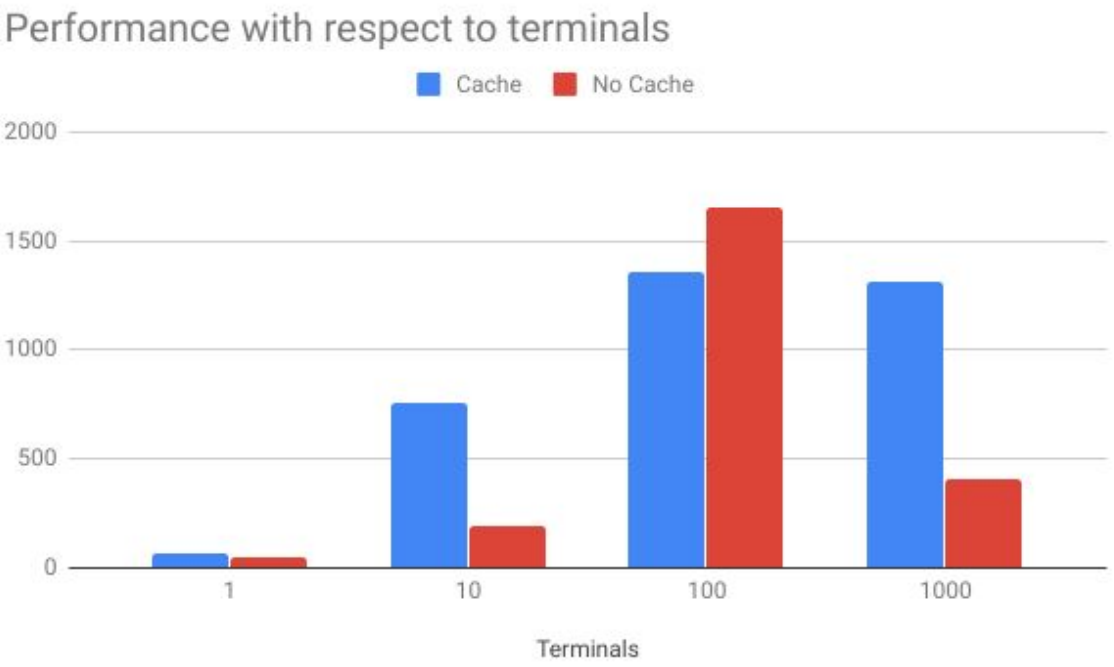

Performance with respect to terminals
Cache
No Cache
6000
4000
2000
0
1
10
100
1000
Terminals

Voter:

The Voter workload is derived from the software system used to record votes for a Japanese and Canadian television talent show. As users call in to vote on their favorite contestant during the show, the application invokes transactions that update the total number of votes for each contestant. The DBMS records the number of votes made by each user up to a fixed limit. A separate transaction is periodically invoked to compute vote totals during the show. This benchmark is designed to saturate the DBMS with many short-lived transactions that all update a small number of records.




Performance with respect to database size
No Cache
Cache
80
60
40
20
0
6(1000 insert)
11(2000 inser)
Database size


Performance with respect to terminals
Cache
No Cache
2000
1500
1000
500
0
1
10
100
1000
Terminals

SIBENCH:

Query distribution:

50% Minoperation 50% UpdateOperation