BangDB – *Performance Comparison with LevelDB and BerkeleyDB*

www.iqlect.com

Overview

The aim of the paper is to analyze the Bandb DB performance vis a vis BerkleyDB and LevelDB. Since all these dbs are key value store and comes in similar embedded flavor, hence comparing these three dbs performance in various conditions would throw some light to the user for them to see how BangDB fares against the other dbs in the market.

The comparison has been done on the similar grounds for all the dbs to measure the data as crisply as possible. Since LevelDB supports one access method hence I have used same across all tests. Also the best write performance by LevelDB is achieved using single thread and best read performance by using 4 threads(on 4 CPU machine), hence I have used the best possible numbers for both these dbs. For BangDB and BDB, 4 threads are used for both read and write. I have again used the same configuration of the machine and the db for all the dbs for the comparison analysis.

Here is the various configurations;

Following machine (commodity hardware) used for the test;

- Model : 4 CPU cores, Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz, 64bit
- CPU cache : 6MB
- OS : Linux, 3.0.0-17-generic, Ubuntu, x86_64
- RAM : 8GB
- Disk : 500GB, 7200 RPM, 16MB cache
- File System : ext4

The BangDB configuration;

- Key size : 24 bytes - Random
- Val size : 100 - 400 bytes – Randomly Picked
- Page size : 8KB
- Write ahead log : ON
- Log split check : ON, check every 30 ms
- Buffer pool size : 1 GB
- Background log flush : every 50 ms
- Checkpointing : every ~4 sec
- Buffer pool workers : ON, every 60 ms
- Number of threads : 4

Other parameters;

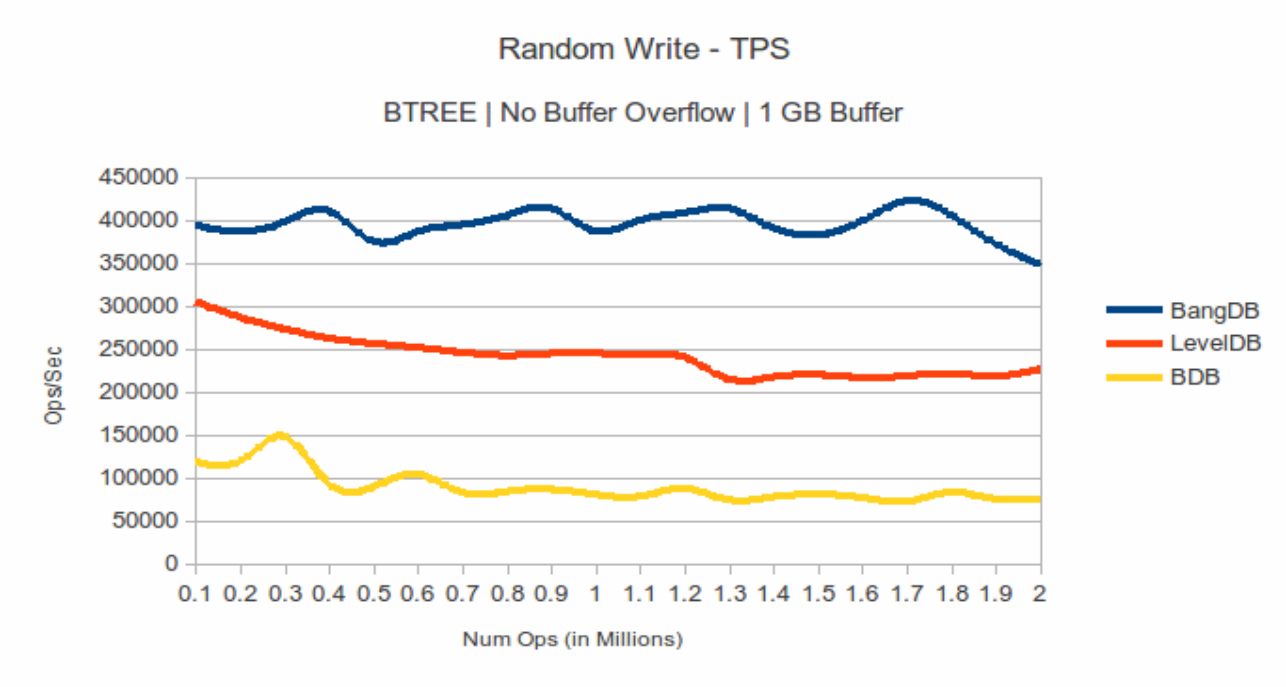
- Write and Read : Random, continuous
- Num of ops : 100K to 10M
- LevelDB Write Buffer : 512 MB
- LevelDB Cache Size : 1 GB
- BerkleyDB Buffer Size : 1 GB

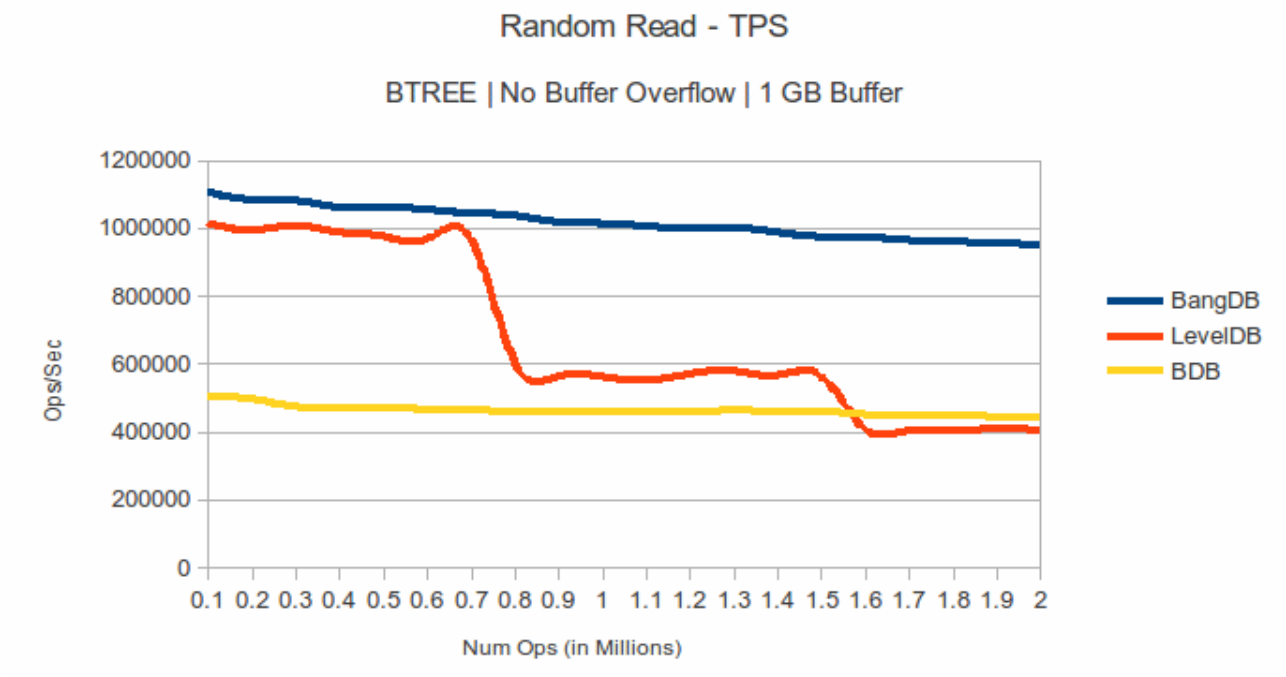
Now in the next section we will go over some of the interesting results;

A) The write and read comparison with BDB and LevelDB. Here the buffer pool size is more

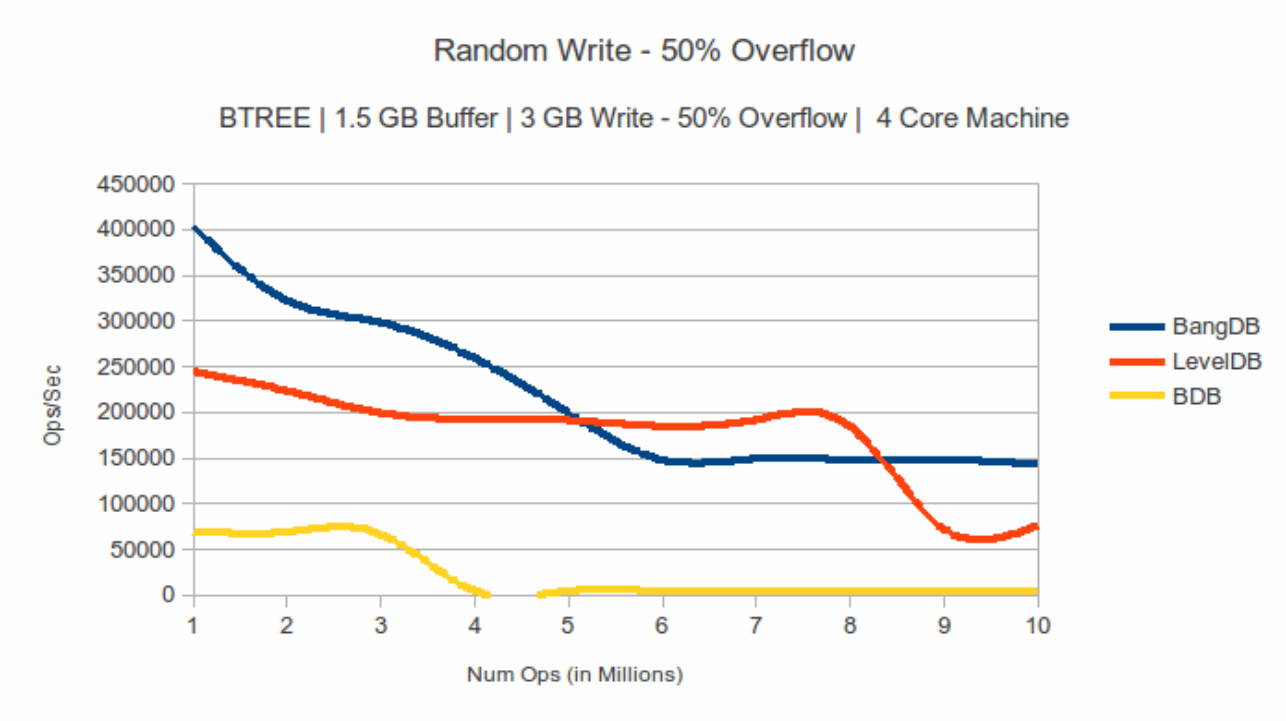
than to fit all the data that we write. Hence no overflow will occur. The log is enabled for dbs. We allocated 1GB as buffer pool and write within the limit.

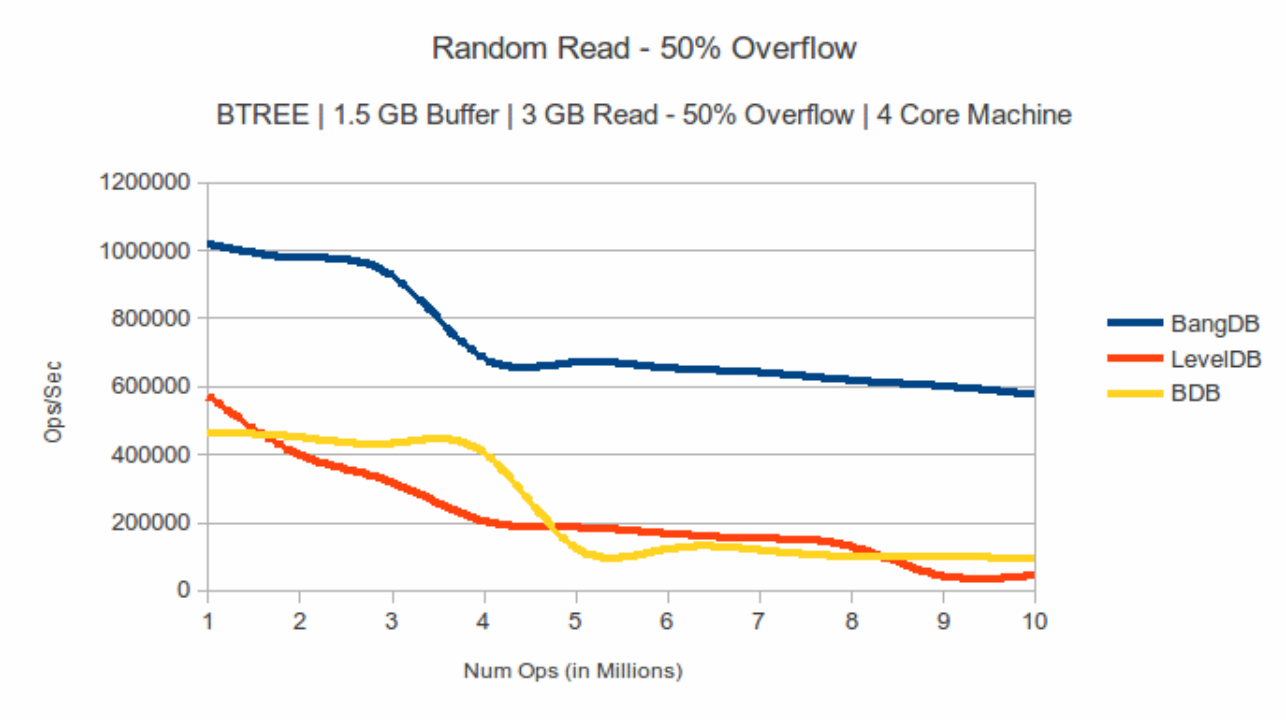
Following stat is for the btree as access method; The results for ehash is also similar but since LevelDB as only tree as access method hence we restrict to that for the comparison





B) The write and read comparison where the db will overflow to disk as well. This time buffer pool allocated was 1.5GB and we wrote and read around 3 GB. The log is enabled for BangDB and other dbs

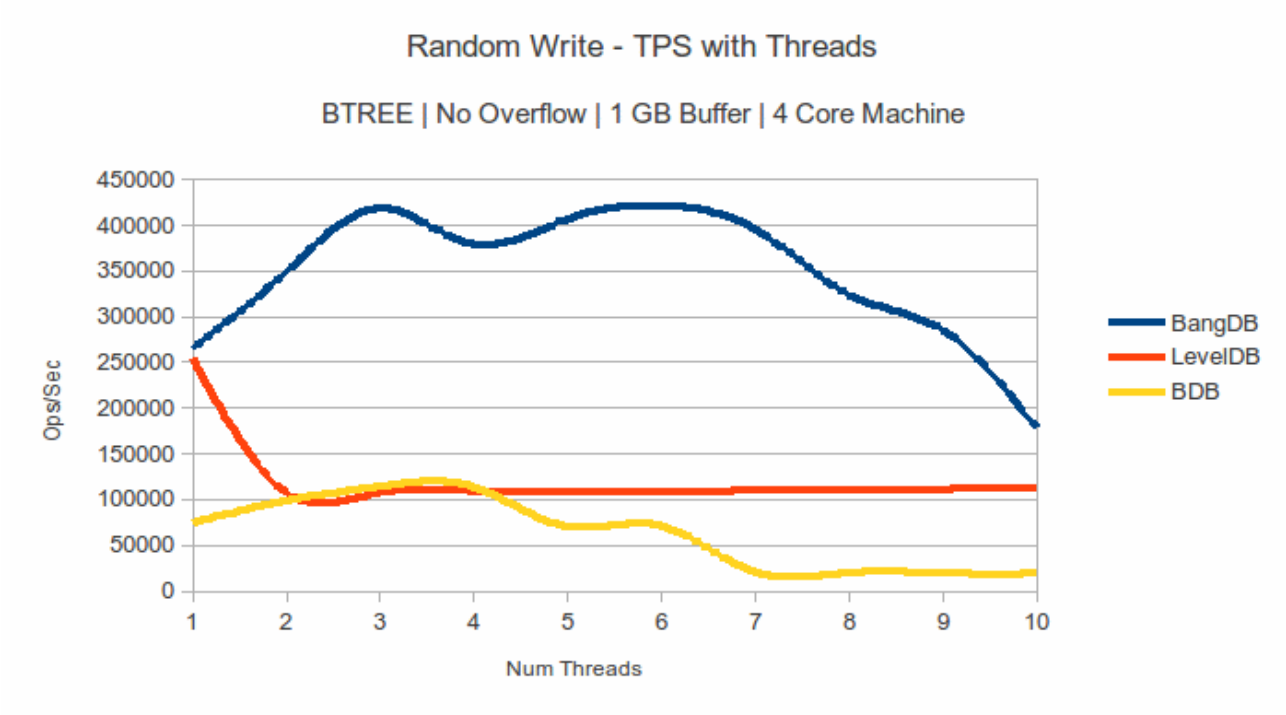




Note that at around 4-5 million the buffer gets full and db starts writing to disk as well. Similarly for read around 4-5 million lots of pages are getting swapped to read from disk. The BangDB currently doesn't compress data hence tend to write more than compared to LevelDB for

the same number of operations. Once we enable compression the above figure for BangDB would further improve.
Important point to note above is that based on our analysis, the write for BDB deteriorates once it starts writing beyond the buffer pool limit. Whereas the read for LevelDB goes down when the limit is breached. But for BangDB read and write both remain consistent and degrade gracefully with the stress

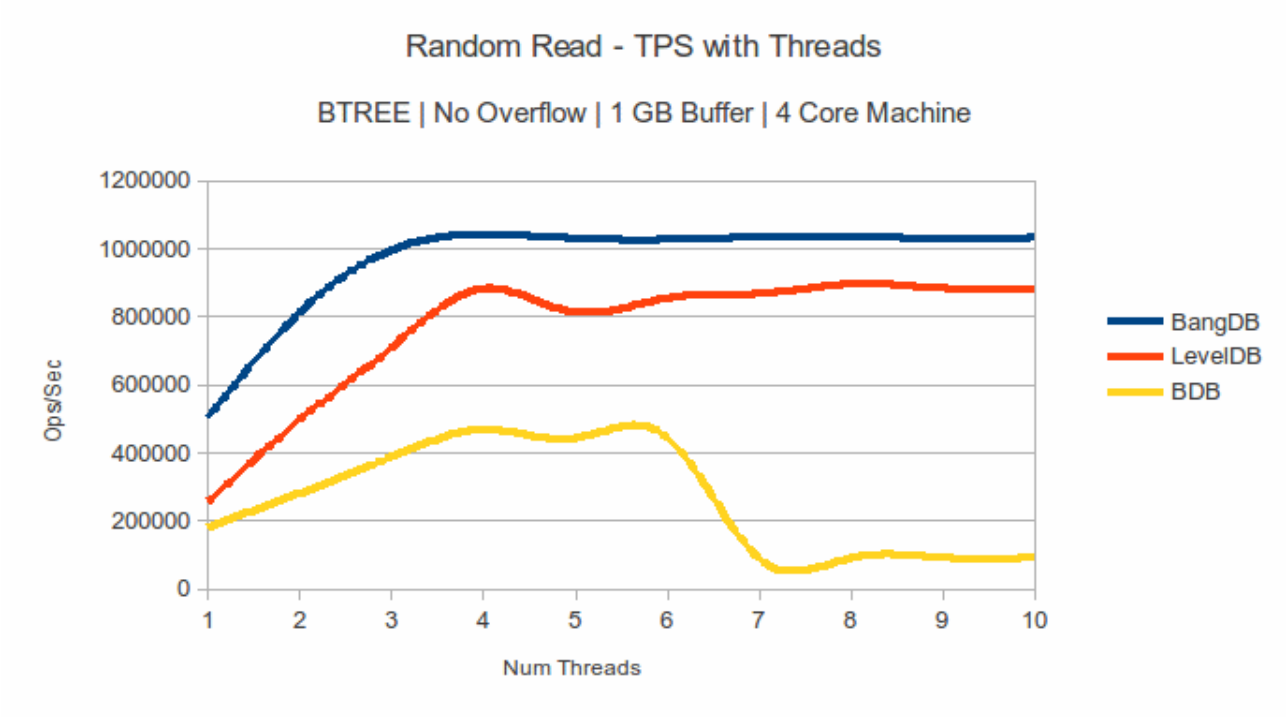
C) Finally test using multiple threads. The test kept on increasing the number of thread from 1 to 10 on 4 CPU machine against 100K to 1M writes and reads. This is how individual dbs performed.



You may see more performance comparison done at

http://www.gridquorum.com/2013/10/bangdb-vs-leveldb-performance-comparison.html

http://www.gridquorum.com/2013/08/redis-vs-bangdb-performance-comparison_16.html



As evident, BangDB is in totally concurrent and performs better with more number of threads on multi core machine. In fact it performs best when number of concurrent threads are close to the number of cores, in this case 4

This concludes our high level performance comparison for BangDB with BerkleyDB and LevelDB.