BangDB – Architecture

www.iqlect.com

Architecture White Paper

BangDB – Embedded Version 1.5 Architecture Overview

BangDB Embedded version is key value store which becomes part of the application process. The BangDB Embedded is part of the BangDB family which will have nosql dbs on various flavors namely, embedded db, network server (client/server) and distributed elastic cluster (IMDG, elastic cache, Data fabric). This white paper is about the architecture of the very first product of the family. The key objectives while designing the BangDB were;

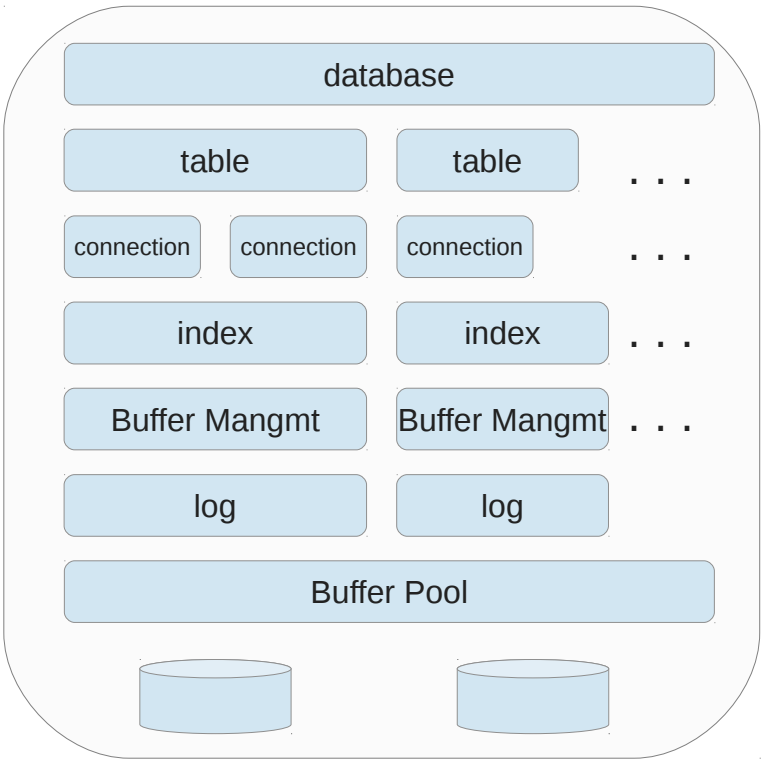
- The Flexibility - key value store in various forms
- The performance and scalability
- The robustness and reliability

The reason for top three objectives was to provide a db which meets various application requirements, scale as needed and remain robust and stable as traditional databases

System Architecture

The Embedded version of the BangDB would look like following from high level.
Basically the three main important components of the db are;

- The Access Methods or indexes
- The Buffer Pool and management, and
- The Write Ahead Log



The Access Methods or Indexes

Btree and hash methods are currently supported in the BangDB as access methods. The Btree implementation is basically the variant one as B+link Tree, where the pages at the same level are referenced by the predecessors. The hash implementation is again a variant one as Extendabale hash with the concept of directory of indexes for the data. The reasons for the variations are basically performance and the high concurrency. The BangDB access methods are highly concurrent, which means on a machine with more CPUs or Cores, the db will perform better. For example, the maximum pages that can be locked at a time for Btree are two that too in some special scenarios which are not so frequent. Similarly the read for Ext hash is completely lock free giving very high read throughput on multi-core machine.

The user can pick Btree or Hash based on their need. For example, when data should be stored in order for later range or scan query, Btree is the option and when order is not important hash can be used. Note that Btree gives balanced read and write efficiency whereas for Ehash the read throughput is much more than to that of write, though Ehash write performance is very close to Btree write performance

The Buffer Pool

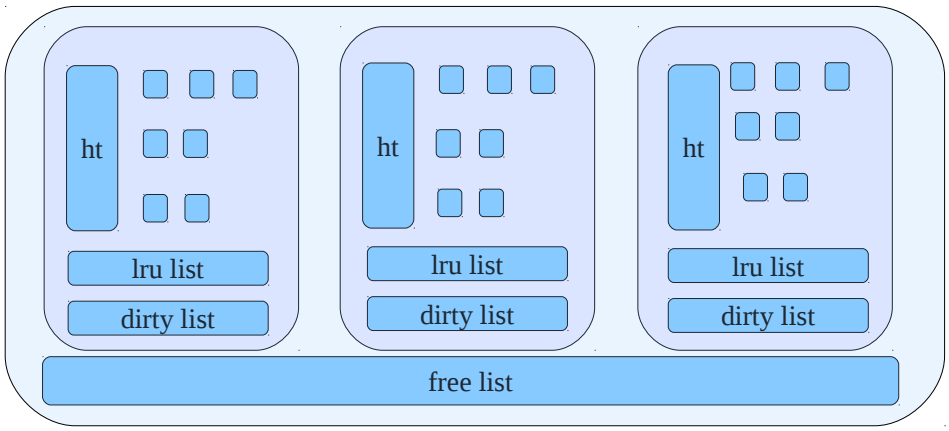
The BangDB, when enabled, reads and writes data from the buffer pool only. The buffer pool provides the flexibility of avoiding the disk completely when not required resulting in higher performance. The buffer pool is allocated according to the hint supplied by the user hence user should provide enough buffer space to the db for better performance. The buffer pool also allows one to control the memory budget on a machine, for ex; one can set the buffer pool size as 256MB and always works out of this much memory even though the data being handled is much much larger than this.

The buffer pool maintains lots of data structure and background workers. For example, buffer pool maintains a hash table for all the page headers for faster access. The lru list and dirty page list are maintained for implementing temporal locality concept and flushing right set of pages at any given time. Also a common free page list is available to all the buffer pools to be utilized as needed.

The Background workers ensure that the buffer pool health is checked frequently and pages are flushed or reclaimed accordingly. Note that the BangDB implements quasi adaptive algorithm for page flushing and reclaiming to ensure that the db pause is not too high when it faces critical shortage of free pages and disk IO does not increase abruptly in high pressure scenario. The workers also try to read and write in bunches and try to avoid separate page writes as much as possible.

The page pre-fetching scheme helps in bringing the pages in memory when it senses that those pages might be accessed in future. The semi adaptive approach of page pre-fetching lets the db bring more than required number of pages by piggy backing the page that intends to bring based on a request

The high level view of the buffer pool is as follows;



The Write Ahead Log

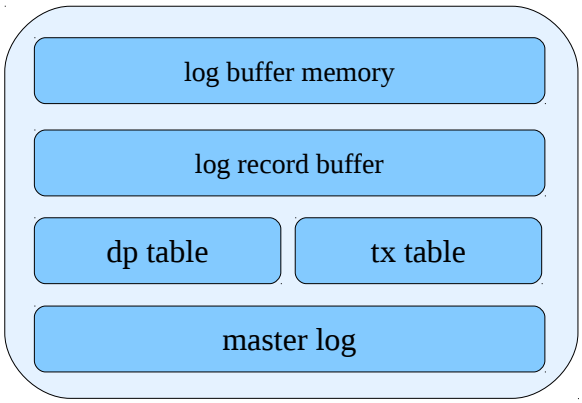
The BangDB implements write ahead log, the ARIES algorithm, for data durability and atomicity. The db always writes a log for every write operation even though it doesn't persist the actual data to the disk. Since db performs all its operations from the buffer pool and doesn't go to the disk at all if not needed that means all the data written in the buffer pool can vanish if the process is killed and data can be lost. Writing log for all data modification operation ensures that the all operation log is maintained and frequent flushing of the log to disk ensures that the data can be recovered from the log if required. The write ahead log provides the data recovering capability when required. For example in the event of process or machine crash etc..., BangDB recovers the data when restarted and brings the db to the state where it was when it crashed.

The log is sequential and it's flushed to the disk by the background workers frequently. The user can set the frequency based on the need, higher frequency means data loss would be minimal in case of any eventuality. BangDB provides the flush frequency knob with impressive Milli sec (theoretically can be in micro sec) as the least count. The default frequency is 50ms and db performs very good even at this frequency.

When enabled, write ahead log keeps on writing the individual operations in the log file and keeps on rotating the buffer as and when it gets filled. The write ahead log provides the log check-pointing, analyze and replay functionality which helps in recovering data when db was not closed properly or in case db/machine crashed. There are multiple workers for wal which keeps on checking to do various housekeeping jobs. One of the workers wakes up regularly and flushes log if required. The log is flushed in bulk and being sequential write it happens relatively quickly. All log flushes are synchronous hence guarantee the persistence of log data which is critical.

The wal also comes handy in implementing the transaction. For data redundancy, the log files can be saved at multiple locations as database can be recreated with the help of the log files. Also for server, log play important role in replication.

The write ahead log(wal) contains log shared memory, record headers, log records, master log various handles etc. The high level view of the wal is as following;



Transaction

The BangDB runs in transactional or non-transactional mode based on the user setting. If enabled, the db provides the full ACID support for the user transactions. BangDB implements optimistic concurrency control(occ) to ensure the ACID. It provides the parallel serialization to ensure highest level of isolation and at the same time high performance by leveraging the multiple cores on the machine. The same db, if opened in non transactional mode works without the explicit begin and commit transaction but still for single op provides the ACID guarantee.

Client Access

Since the embeddable version becomes part of the process hence there is no separate layer for the db to be used by the client. The client loads the db into the process and access through the set of APIs. But for other versions or flavors of BangDB, a client layer would be provided apart from other required layers.

Please see the API doc for detail info.

Conclusion

The BangDB is driven from three core goals, flexibility, robustness and performance. The db comes in various flavors to suit different needs of may be a single application as in most cases. The db ensures data atomicity, durability and recoverability. And db remains high performer even in very conservative settings. The BangDB gives the highest IOPS in several scenarios as displayed in our compete analysis with Oracles BerkleyDB and Google's LevelDB

This is very high level description of the db. The detail discussion would be in separate documents for individual components. Please check out the documents available on the site at www.iqlect.com