
百度云
Cockroach DB
CockroachDB
中国社区大会

Gaia, Baidu's Next-Generation Database

Zhang Wanchuan, Baidu
zhangwanchuan@baidu.com

Self intro

Wanchuan Zhang (张皖川)

- 2001, University of Wisconsin, Madison，Ph.D（math) & MS（CS）
- 2001 Oct - 2018 Mar, IBM DB2 (LUW) development, Portland lab & Beijing CDL
- 2018 Mar，Baidu cloud，cloud database development

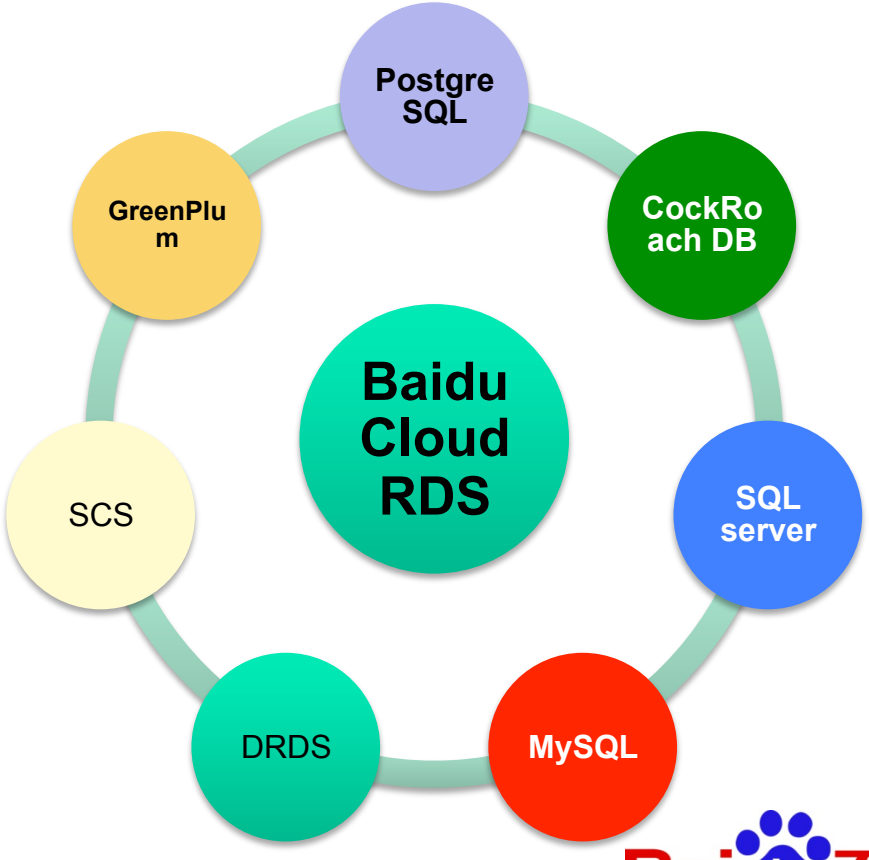
Agenda

- Baidu Cloud RDS (status quo)
- New developments of cloud database service
- Gaia, Baidu’s next-generation database
- Future outlook

Baidu 百度

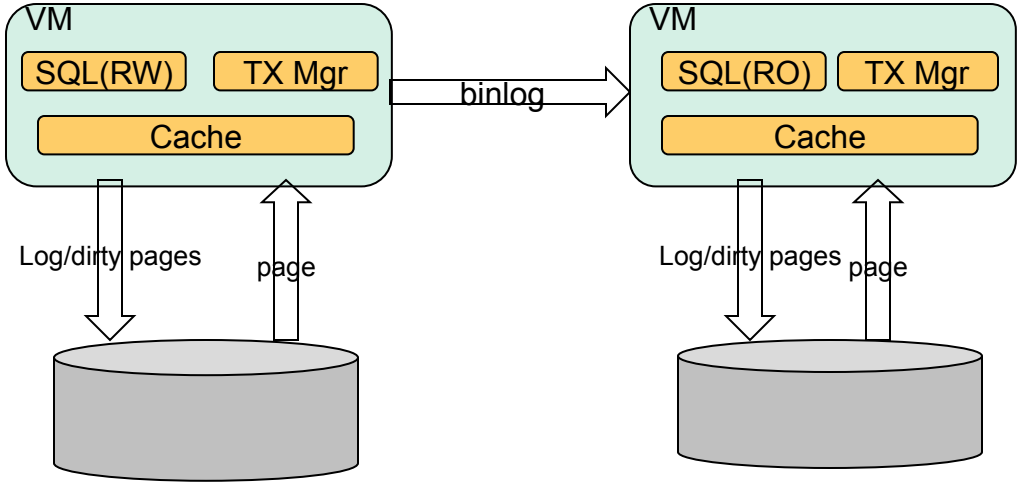
Baidu Cloud RDS

- Database service offering on Baidu cloud
  - MySQL 5.6, 5.7
  - DRDS (distributed tx support)
  - SCS (Redis based cache service)
  - SQL server
  - PostgreSQL
  - GreenPlum (MPP db for analytics)
  - Cockroach DB (newSQL, geo -distributed HTAP database)
- MySQL dominates the deployments



Bai du 百度

Current arch for baidu cloud db



- Monolithic
- Data on local disk
- HA through logical replication
- Offload read to replicas

Baidu 百度

Pain points

- No storage multi-tenancy
  - Low disk utilization (overall usage <= 30%)
- Little elasticity, need to redistribute data to scale
  - Slow replica deployment
  - Capacity change hard
  - Configuration change hard
- HA not sufficient
  - Replication lag
  - Fast recovery from failure?
  - Zero data loss ?
- Far from autonomous

- Performance far from adequate
  - Write amplification
    - Data
    - Logs (binlog, redo log)
    - Double write
    - Repilca
  - Negative correlation btw the foreground and background jobs, e.g. flush dirty pages, checkpoint, purge, backup …
  - Low CPU utilization (<10% useful work)
    - Synchronization cost
    - Scheduling overhead
    - Cache miss
  - IO delay
  - . . . .

Baidu 百度

Cloud RDS service desired characteristics

Ideal characteristics

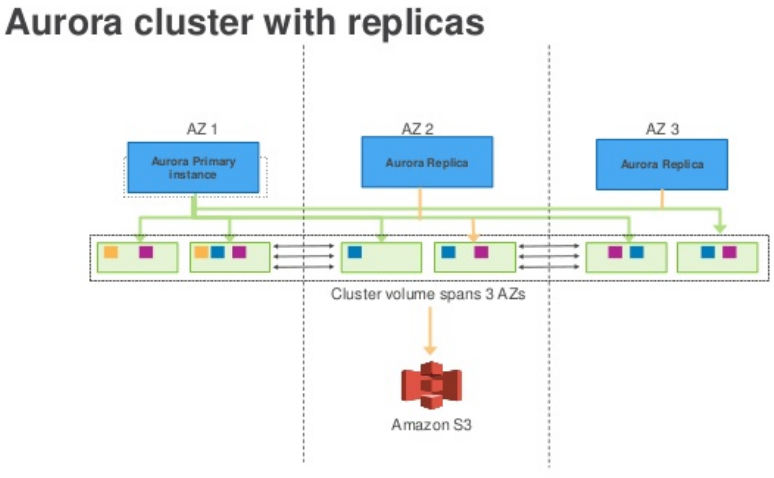
- Multi-tenancy
  - Effiicient resource sharing, e.g. CPU, memory,IO,network
  - Resource consolidation, e.g. Light weight live migration of database tenants
- Elasticity
  - Scale out to adapt to unpredicatable workload characteristics change
- Adequate SLA
  - High availability SLA (24x7, RTO/RPO)
  - Performance SLA(TPS/RT)
- Autonomous & self managed
  - Ease management of large scale deployments
- Pay per use service

Baidu 百度

Emerging new architectures, e.g. aws aurora

AWS aurora highlights

- Decoupled TC(transactional componen and DC(data component)
- Distributed fault tolerant purpose built database optimized cloud storage
- Performance boost
  - Database is all about IO
  - Network attached storage is all about packets per second
  - High throughput processing is all about conte switch



Where did this architecture come from?

Fusion innovation of

- Decoupled TC & DC, e.g. MS deuteronomy system
- Distributed cloud storage
- Open source RDS, e.g. MySQL

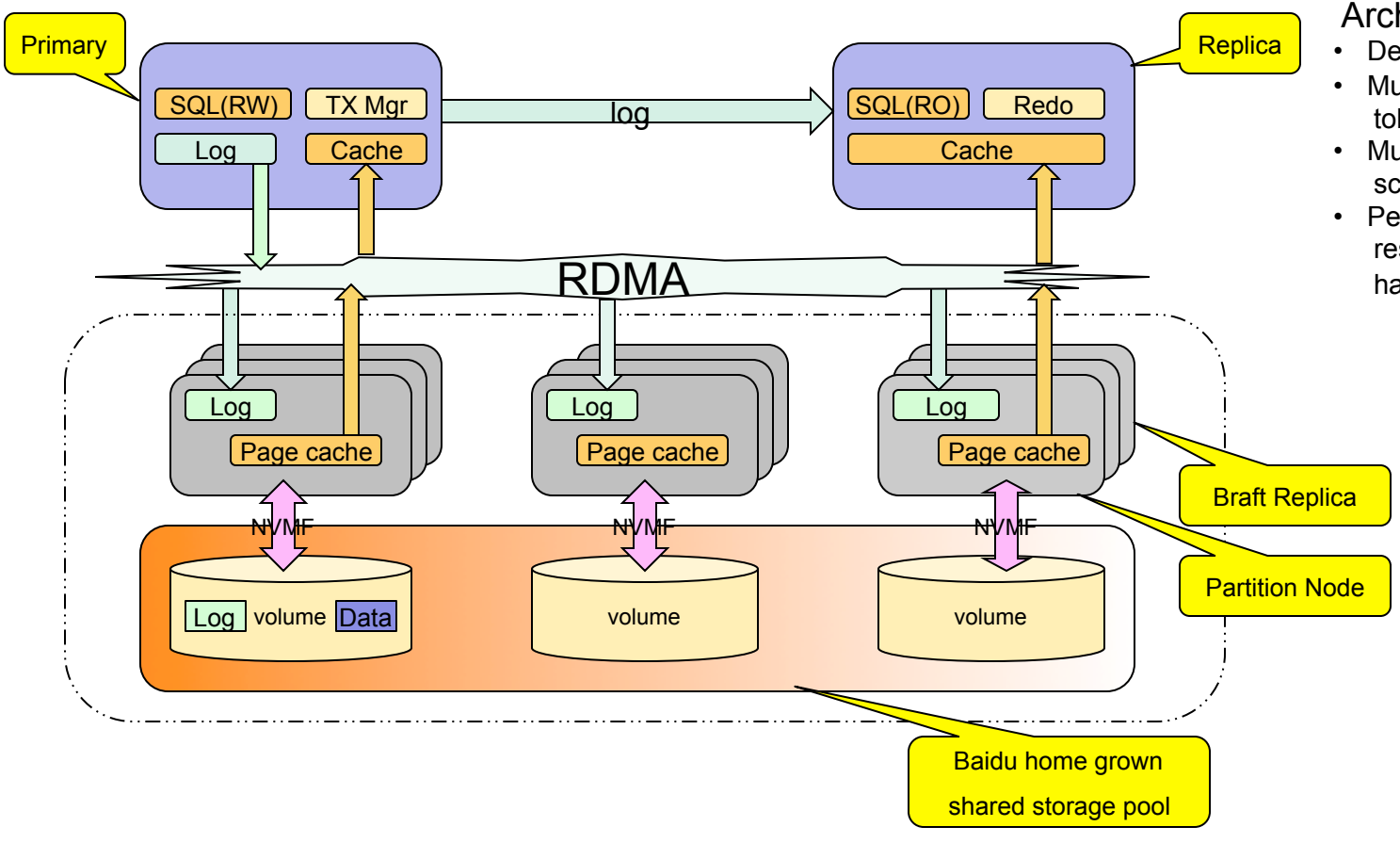
Baidu 百度

Gaia Design principles

- Compatible with MySQL
  - Explore the mature ecosystem
- Decoupling TC & DC for better resource utilization, elasticity & HA
  - Offload log & data persistence to storage node
  - Reduce network traffic
  - Remove background data jobs
  - Each layer can scale independently
  - Improved RTO/RPO
- Multi-tenant distributed fault tolerant storage
  - Distributed Storage to localize failure
  - Leverage mature replication technology for per partition fault tolerance
- Database performance optimization
  - Facilitate better processing pipeline for performance scalability
  - Batching to reduce per item cost
  - Adapt software design to HW change to harvest HW acceleration
    - Lock free data structure & algorithms
    - RDMA
    - NVMF
    - NUMA optimization

Baidu 百度

Gaia Architecture Overview



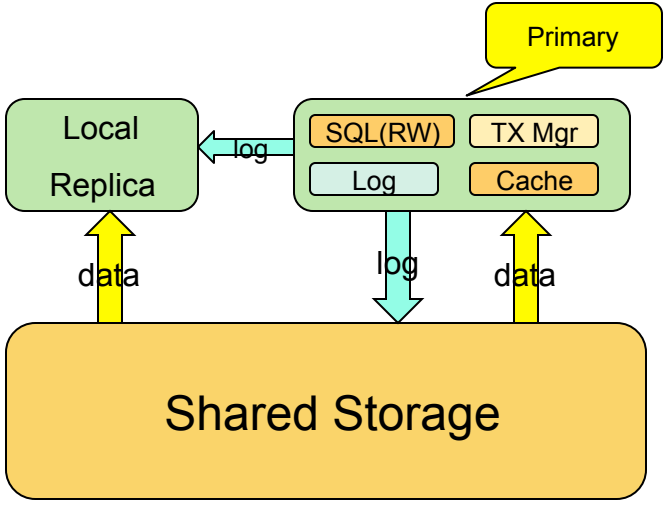
Architecture highlights

- Decoupled TC & DC
- Multi-tenant distributed fault tolerant storage
- Multiple replicas for read scaling & failover
- Performance boost by better resource utilization & harvesting HW acceleration

Gaia computing layer: primary

Primary

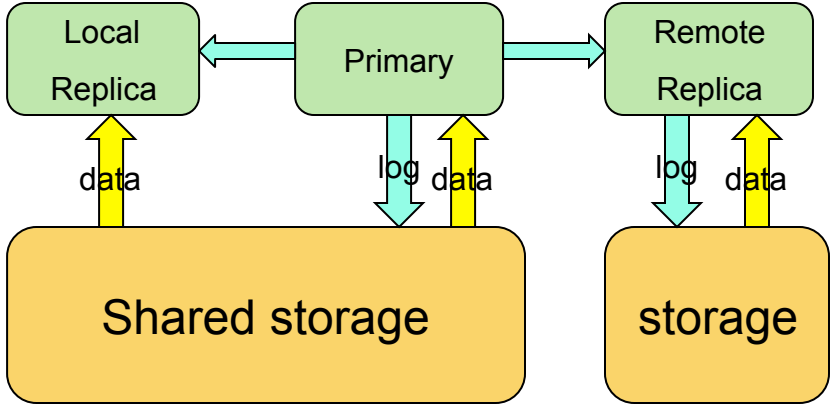
- Handle RW workload
  - Focus on SQL execution, transaction & cache management
  - Update cache (no disk write)
  - Generate log and ship log to storage (RDMA)
  - Move forward log durable point (VDL) upon Ack by storage
- Cache policy
  - cache the latest version of the data page
  - can't evict page with page Lsn <= VDL
- Asynchronously send log to replicas
- Performance optimization (discussed later)



Gaia computing layer: replicas

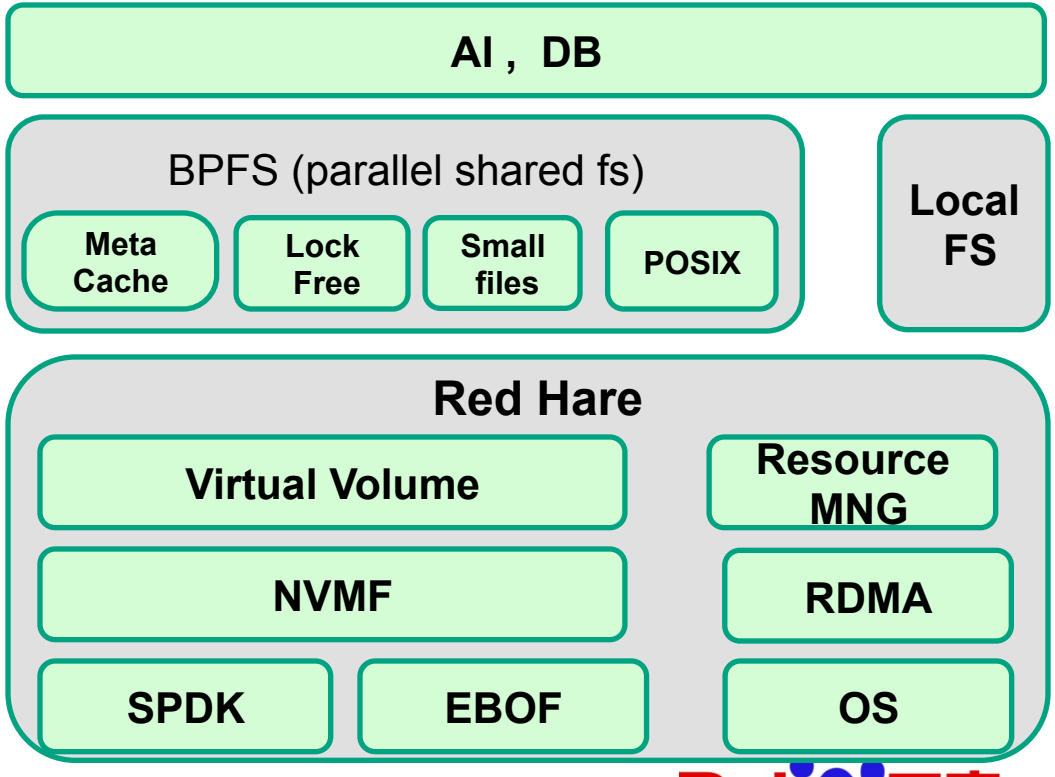
Replicas

- Support RO workload
- Support both local & remote replicas
  - Local replica for read scaling & fast takeover
  - Remote replica for DR & local read
- Enable planned & unplanned takeover/failover
- Update cache by applying redo log
  - Fully parallel apply & apply to cache only
  - MTR (mini-transaction, e.g. BTree split & merge) atomic redo
  - Light weighted mechanisms to order redo of dependent log records
- Support MVCC read (RR isolation)
  - Obtain active transaction list from primary upon start up
  - Maintain active transaction list while redo marches forward (transaction information embedded in redo log records)
  - Readers create “read view” based on the active transaction list



Gaia Storage

- Database is partitioned across a fleet of storage nodes
- Data volume allocated from cloud storage resource pool (Baidu home grown unified storage "Red Hare")
  - Better multi-tenancy
  - Unified storage management
  - HW acceleration to boost performance (user mode IO stack built with SPDK, NVMF, RDMA etc)
  - Autonomous features
    - Auto growth
    - Self healing



Gaia storage

Storage services

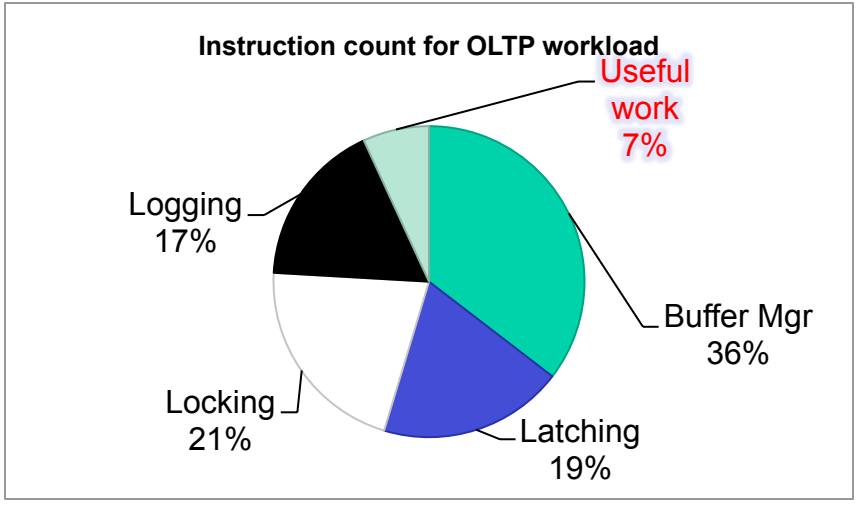
- Persist log & materialize data (D within ‘ACID)
  - Replicate log received from primary
  - Replicate log through parallel raft (braft) and send Ack back to primary
  - Apply committed log to materialize data pages
- Provide shared data access by primary & replicas
  - Light weighted mechanism to maintain multiple page versions for MVCC
- Offload background jobs to storage
  - Periodic back up data & log
  - Coordinate with primary & replicas to purge data & log

Baidu 百度

Gaia database performance optimization

Optimized database performance

- SEDA like threading model
  - Decoupled user thread from connection
  - Dedicated log IO thread
  - Asynchronous commit
- Scalable logging subsystem
  - Lock Free WAL
  - Flushing pipeline
  - NUMA optimization
- Optimized memory management
- More to come …



"OLTP through the looking glass" sigmod08

Gaia (primary server) thread model

MySQL thread model is evolving, as of 5.6 it features

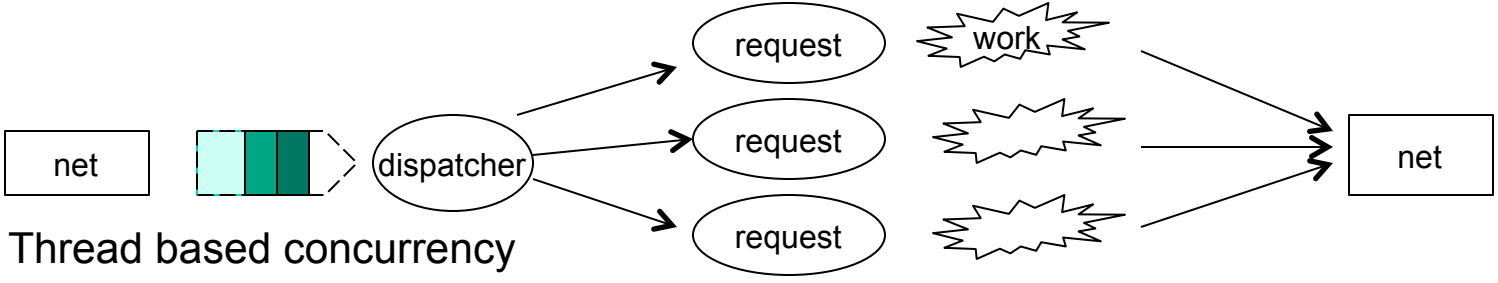
- Threaded server, 1 thread per connection
- 1 page cleaner
- User thread does log IO
- User thread wait until log IO complete before commit could proceed, e.g. unlock, fix transaction state, respond to client …

Pain points

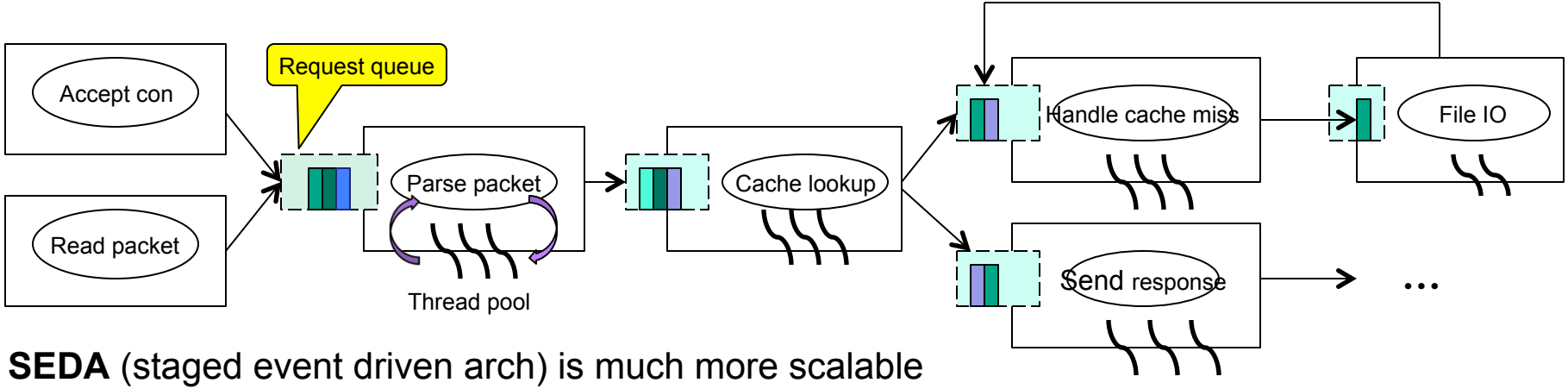
- IO wait
- scheduling overhead, e.g. context switch
- Overcommitting resource, e.g. performance degrades severely when large number of threads dispatched (> 1000 connections kills performance)
- …

Baidu 百度

Digression into SEDA (for scalable web service)



Thread based concurrency



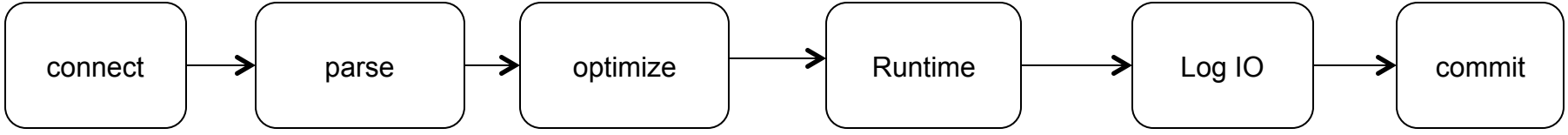
SEDA (staged event driven arch) is much more scalable

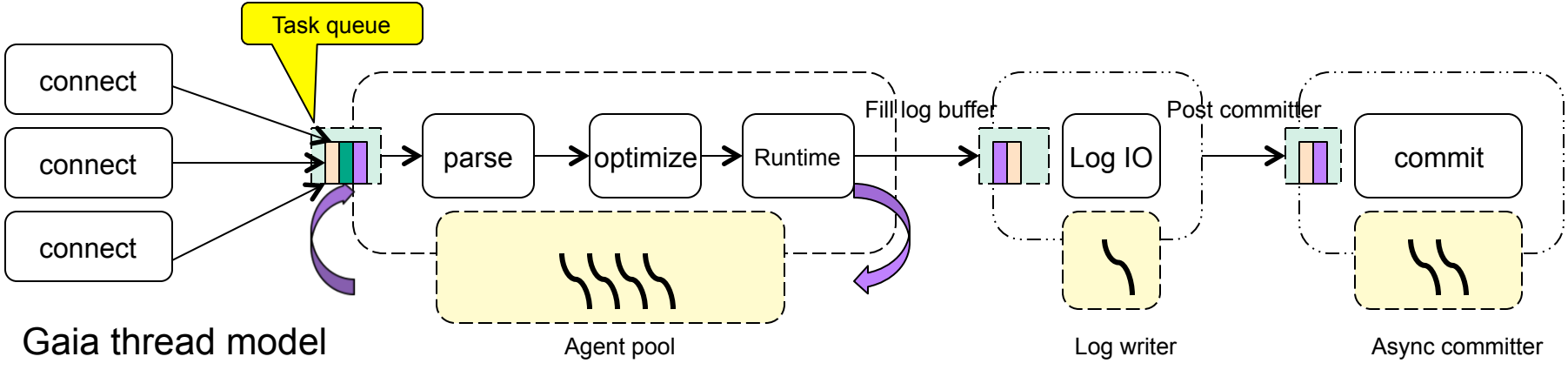
- Jobs divides into stages
- Dynamic & bounded thread pool per stage
- Stage connected through task queues

Baidu 百度

Query processing pipeline

Query processing stages





Gaia thread model

Q: can we break it into more fine grained stages?

Starvros, H " A case for staged databases"

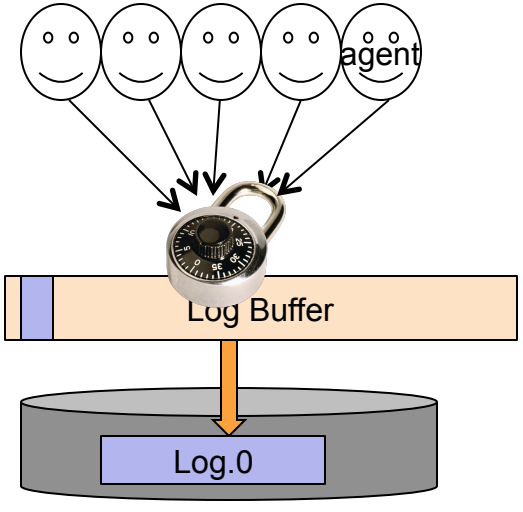
Gaia(primary server) thread model

Gaia thread model

- Decoupled user threads from connections
  - A pool of user threads (agents) serve all connections
    - User threads pull tasks off the incoming task queue
    - Dynamic pool size for better resource management
  - Avoid over committing resource
- Dedicated Log IO thread, i.e. log writer
  - Better IO batch
  - Facilitate flushing pipeline
- Asynchronous commit by dedicated threads, i.e. asynchronous committer
  - Hide sync IO delay
  - Improve CPU utilization
- Multiple page cleaners for better parallelism

Baidu 百度

Scalable logging



Current Logging model (as of MySQL 5.6)

- Acquire log mutex
- Increment Log Sequence Number (LSN)
- Copy log records to log buffer
- Flush log to disk if sync IO is required
- Release log mutex

Logging bottlenecks

A. IO delay
  - Need to ensure log gets durable on disk before commit

B. Scheduling overhead
  - User agent scheduled out while waiting for IO completion

C. Log induced lock contention
  - Locks held while waiting for IO completion increase chances of lock conflict

D. Log buffer contention
  - Synchronization cost for parallel access to log buffer

Baidu 百度

Scalable logging

Asynchronous commit (A,B)

- Instead of waiting for IO, user agent could be freed up for more useful work
  - User agent detach from the transaction to be committed, register the transaction to be committed (on a wait queue) and continue to work on other requests
  - After log IO completion, log writer notify the asynchronous committer thread to continue processing the transaction to be committed and send response back to client afterwards

Early lock release (C)

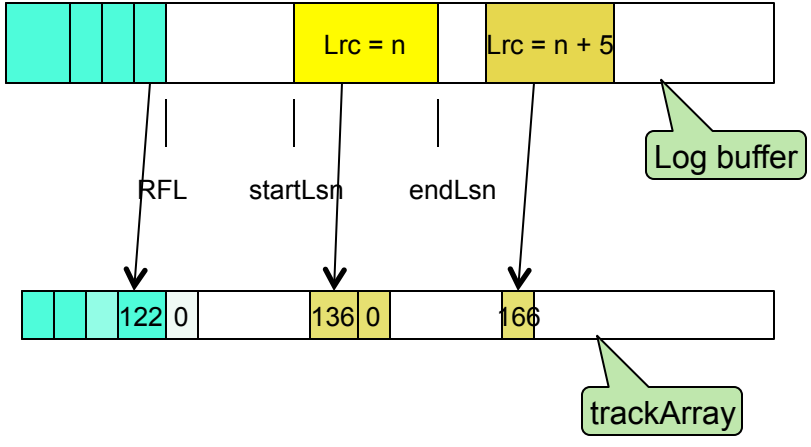
- DeWitt observed that a transaction could release its lock before its COMMIT log records gets written to disk as long as it does not return results to the client.
- Partially strict 2PL (Soisalon-Soininen, E., Ylönen, T) propose that transaction releases locks as soon as its commit request arrives.
  - If a transaction aborts before its commit requests arrives, all accepted histories are recoverable and the recovery properties of strict 2PL are preserved
  - If a transaction must be aborted after its commit request arrives (can only happen in case of rare system failures). These properties does not hold. Suggest to abort all active transactions for simplicity.

Baidu 百度

Scalable logging

Lock free WAL (D)

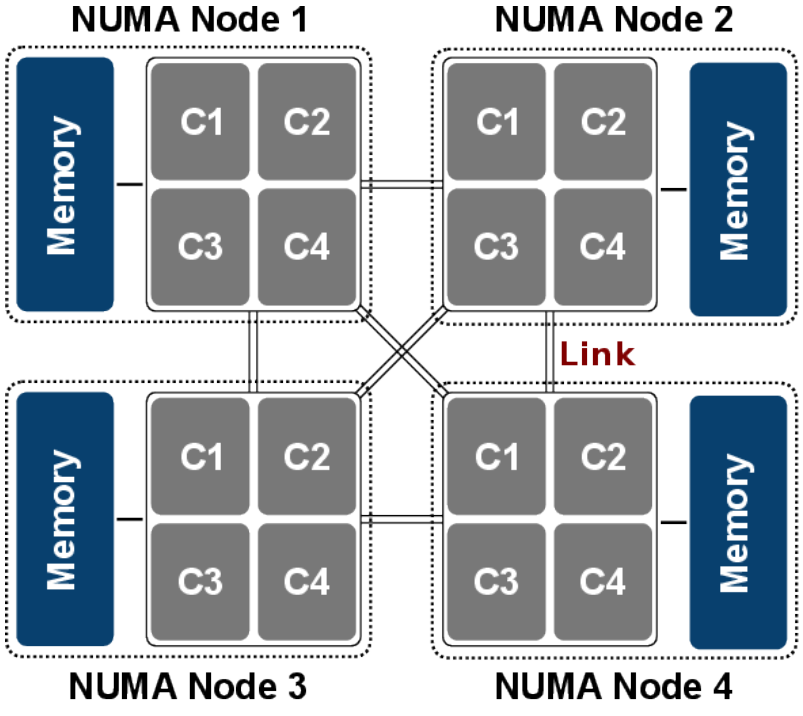
- User agents update an global atomic to generate LSN for its log records
- User agents copy log records into log buffer in parallel
- There might be holes in the log buffer as the agents make progress in parallel
- Need to track the order of log copy so that log writer can flush up to a LSN position up to which all prior log records (in LSN order) are all copied in
- A couple of solutions exist, e.g. use an auxiliary array to track the order by which the log records are copied into the log buffer



Scalable logging

NUMA optimization

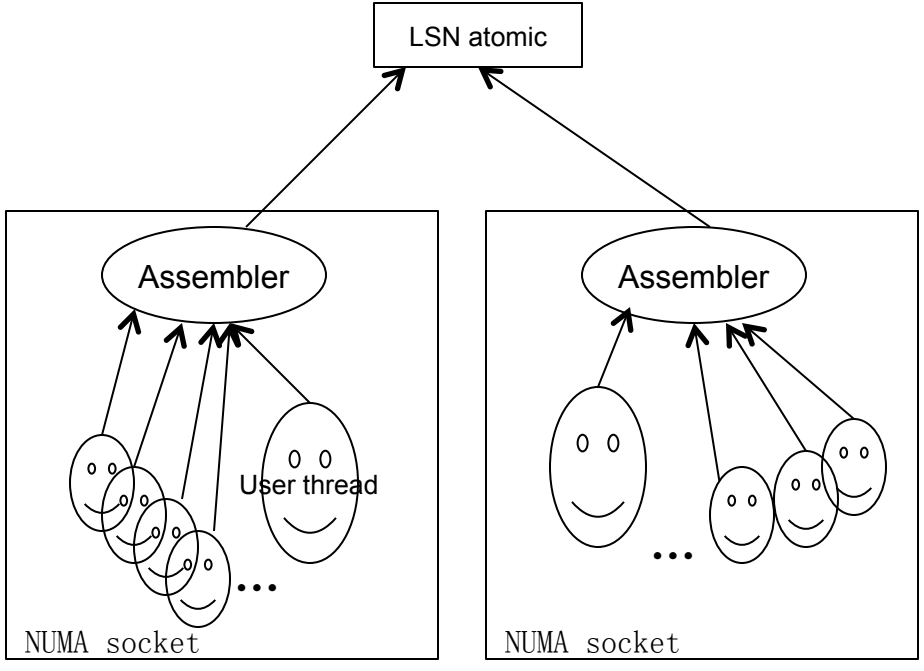
- NUMA (non-uniform memory access) servers becomes more popular now
  - Cores are grouped into socket with dedicated memory controller for fast access to local memory
  - Inter-island memory access (through memory bus) is much slower
- Software design need to adapt to NUMA by paying attention to where the memory resides



Scalable logging

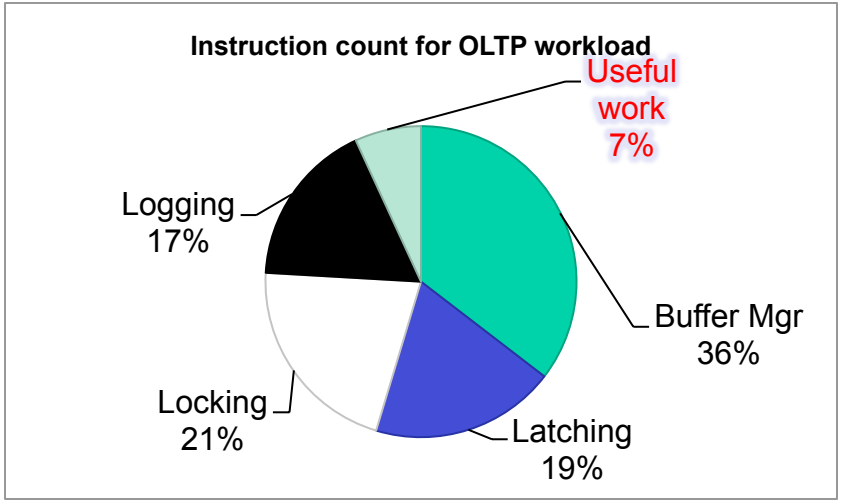
NUMA optimization

- Assign 1 log assembler thread per socket
  - Assembler coalesce log record requests from the local user agents
  - Update the LSN atomic on their behalf
  - Issue the LSN range to each of the agent so that the agent can copy its log records into the log buffer accordingly
- User agent request LSN from the local assembler to reduce inter socket access overhead
- Full scalability requires more: distributed log buffer & multiple log stream



Future out look

- Better performance
  - Where did time go ?
    - Improve under current DBMS arch
      - ~~scalable logging~~
      - Scalable lock manager & cache
      - Latch free data structure & algorithms
      - NUMA optimization, e.g. memory management
    - A complete rewrite for 10x better TPS for next generation Gaia?
      - Decoupled compute & data persitence
      - MVCC + copy on write update
      - Lock free record cache, e.g. lock free hash table, Bw-Tree index
      - Flexible choice of storage, e.g. log structured store



"OLTP through the looking glass" sigmod08

Baidu 百度

Future out look

- More on elasticity
  - Multiple writes
  - HTAP capability & dynamic scale for analytics
  - Database live migration & Instance consolidation
    - 90% of the DBMS start small and never grow up
    - Consolidate small instances onto fewer boxes for cost reduction
    - Migrate out to more powerful nodes in case of workload rise
- Autonomous DBMS
  - Auto resource management, e.g. self managed memory tuning
  - AI for autonomous DB, learned caching policy, learned stats, learned workload …
- Integrated AI services
  - In-Database R/Python analytics run as store procedure
  - ...

Baidu 百度

Q&A

Thank you!


Thanks
2019年1月12日
百度云
Cockroach DB