Scaling MySQL and Java in High Write Throughput Environments

How we built Spinn3r

spinn3r

What is Spinn3r?

- Licensed weblog crawler
- 500k posts per hour (RSS+HTML)
- 3.5TB of content
- 10 months of blog archives
- 3B documents
- 80Mb /s - 24/7

spinn3r

Hardware

- ~40 servers
  - Quad Core
  - 8GB memory
  - Gigabit ethernet
  - Dual SATA (software RAID 0)
- Moving to SSD

spinn3r

Write Throughput

- 90% write, 10% read
- MyISAM didn't scale
  - Too many seeks in high write load
- InnoDB with write ahead log
  - 1/5th of effective disk bandwidth
  - Improve the fuzzy checkpointing logic
  - Just continually write memory images (log structured)
  - 1.5 minutes to write an 8G image

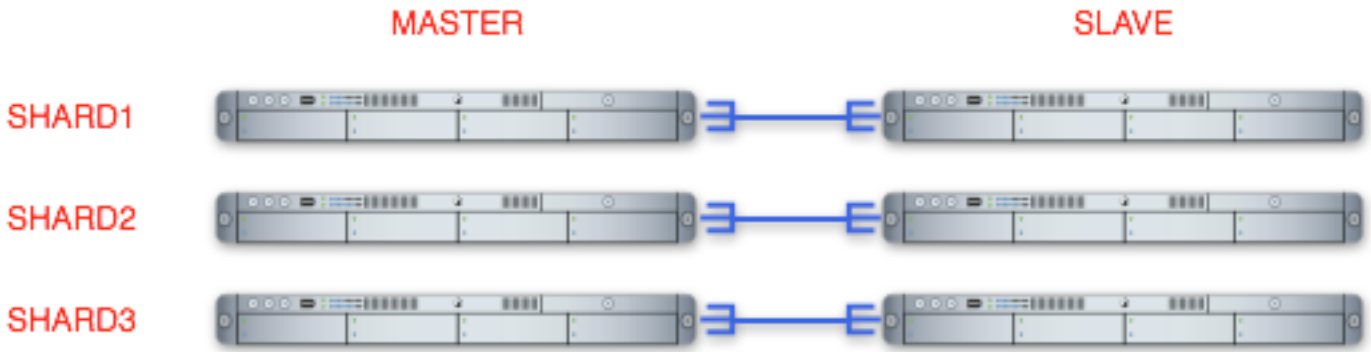
spinn3r

Database Sharding

- Split data across shards based on PK
  - hashcode of URL
- Range routing
- Limitations
  - No triggers
  - No foreign keys
  - No transactions
- Similar philosophy to Bigtable, S3, Dynamo, etc

spinn3r

Shard Architecture



spinn3r

Query Limitations

- No functions in WHERE clauses
- LIMIT required
- Query should be deterministic
  - ORDER BY
  - ID = N
- Must order by some column to page
- No offset
- No aggregate functions

spinn3r

Shard Insertion

- Bulk insert data
  - Custom API
  - Operate on lists, commit every N records or T minutes.
  - INSERT … ON DUPLICATE KEY UPDATE
- Parallel dispatch architecture

spinn3r

In-memory Storage

- Metadata
  - queue
  - graph
- Deprecated memcached
- Allows InnoDB to execute at speed
- WAL allows disk to write at about 40MB/s

spinn3r

On-disk Storage

- 2.5 TB of content (full HTML and RSS)
- Numerous backup copies
- RAID caching controllers with BBU
- InnoDB blobs with to append-only and ‘eventually immutable’ tables.
- Gzip compressed (3x savings)
  - Reduces the # of IOs by trading CPU

spinn3r

Resource/Primary Key

- Key is truncate(SHA1(resource+secret))
- Deterministic mechanism for key generation
  - works across robots
- Works well with shards
- Routable
- Decentralized
- Avoid clustered indexes

spinn3r

Distributed Lock Manager

- acquire(lock)
- renew(lock)
- Similar to Google's chubby
- See Paxos algorithm for distributed consensus
- Good for master servers, failover, etc.
- We use this for master queue promotion

spinn3r

Sequence Generation

- Need monotonically increasing sequences
  - Paging through results
- Settled on global prefix+local suffix with a distributed lock manager
- Used in shards to page across results.
  - paging on time is hard/impossible due to collision

spinn3r

Task/Queue

- Similar to MapReduce
- Central queue
  - Fault tolerant
  - Sharded for scale
- Distributed tasks
- Executes robot jobs over 30 machines
- Supports heterogeneous machines

spinn3r

JDBC Load Balancing

- Created lbpool
  - Licensed to MySQL (Open Source)
- Load balanced connection pool
- Replication aware
- Handles runtime rebalancing
  - slave lag
  - broken slaves
- Fault tolerant

spinn3r

User Defined Functions

- Necessary for distributed databases
- Row level locks to avoid race conditions
- Increment
- Bloom filters
- Zeta codes
- Histographs

spinn3r

Solid State Storage

- NAND based flash devices
- SUPER fast reads
  - 15k 4k reads per second
  - ~250/s for HDDs
- Regular performance writes
  - Small InnoDB buffer pool
- Historically avoided to due high MTBF

spinn3r

Current SSD state

- $30 / GB
- 16/32/64 GB capacity
- Mtron
- Memoright
- STEC
- ~ 100MB/s sequential write
- ~ 120MB/s sequential read

spinn3r

The Future of DB Storage

- SSD for in-memory data
- 10x performance boost for 20% cost increase.
  - $30/GB now -> $15/GB in Q2-Q3
- Mainstream in 2009
- MUCH more data per node
- Log structured databases
- See benchmarks

spinn3r

Questions

- Further reading:
  - feedblog.org
  - spinn3r.com
  - feedblog.org/category/ssd/
  - code.google.com/p/mysql-lbpool/
  - Paxos algorithm
  - Chubby

spinn3r