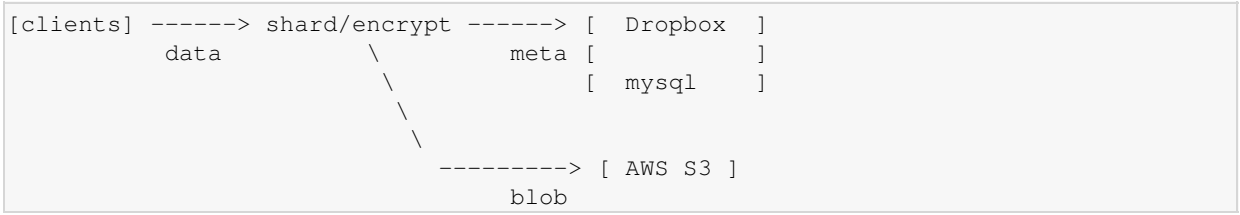
1 MySQL at Dropbox, Renjish Abraham

1.1 Dropbox architecture:



1.2 Key features of MySQL for dropbox:

- Free, most popular
- predictable performance
- Allows workload control tactics to achieve system predictability
  - ♦ Avoid cache churn
  - ♦ Reduce instances of excessive background activity
  - ♦ Stable I/O subsystem performance

1.3 Dropbox backend arch

Webserver boxes, backed by memcached boxes and db(mysqld) boxes

Real performance issues at 20k connections to mysqld, although 10k connections are fine.

So, introduce custom db proxy. Now only 200 connections to mysqld

1.4 "Flexible Schema"

> Technique to store key-values in a blob. So instead of having 10 columns for 10 attributes in a table, use a single blob column. Use a service layer to abstract the db as a black box for (internal Dropbox) app developers.
>
> DW: I spoke with dropbox db people after this: The blob is Protobufs-formatted, which means that they can add/remove key-statically-typed-values whenever they want, as long as it all fits in the blob. Because they have this data service layer, the app people never see this remapping. Apparently the app people dislike db interfaces anyway, and they want a narrow data API that hides db details from them, because they don't care about SQL flexibility.

1.5 Automation

Only 2 DBAs for hundreds of db servers. Automation is clearly needed, so they wrote lots of admin scripts. But they found scripts are dangerous, easy to break, easy to destroy production. So they introduced a common framework, a unified interface to administration (hiding direct use of dangerous scripts). Now, they do code reviews.

1.6 What Dropbox wants from mysqld

- better single-threaded performance
- better replication
- online schema changes
- stored procedures
- faster recovery
- dta fragmentation

1.7 Dropbox future

- Shift whole backend to database as a service (put service layer around entire db layer, not just parts)
- Eliminate use of monolithic db
- Enhance db ops
- Introduce compression, improve performance

Believe "database as a service" common framework was a huge win.

2 Facebook Async mysql client, Chip Turner, Facebook

2.1 History

Dedicated db client team formed in 2013. Before, the db client code was "organically grown", "community-owned", "ugly". No one really owned the code, so different teams just patched on what they needed without trying to make the overall system good.

2.2 Fun facts

Throughput metrics: queries per second, rows per second

Numbers you should know:

- Mem access: 0.1us
- SSD read: 150us
- Datacenter round-trip-time: 500us
- Disk seek: 10ms
- California-Europe RTT: 150ms
- Blink of eye: 150us

2.3 Goal of new client

Minimize code changes when migrating from sync client to async client lightweight, low overhead

2.4 Key abstraction: "await" primitive in PHP

Most facebook code in PHP. To migrate, prepend API call with "await". This makes the call non-blocking. To wait for the call to finish, "await_all()", much like joining threads.

Python client uses gevent library C++ client: other clients built on top of this. Basis for ads, spam detection, search code (C++). Uses callbacks.

2.5 Rough approach

- Extend libmysqlclient: non-blocking sockets
- API indicates read/write
- file descriptor (fd) exposed, so you can use it for select/poll/epoll
- client lib internally uses state machines to keep track of what's going on.
- Same API for async and sync, to allow gradual migration. Some code is more fragile and tricky to port to async.

2.6 Implementation notes

- mysqlclient changes are invasive
- mysql test suite was useful for verifying correctness and catching bugs.
- partial deployment was very helpful
- Can toggle async-sync on a per-call-site basis
- tight monitoring during rollout, watch for errors after enabling each feature.

2.7 Final notes

Try in webscalesql.org python api: github/chipturner

FB's PHP is a little non-standard: using HHVM and Hack (an enhancement of PHP that adds static types, generics, lambdas, etc.)

FB has a culture of moving-fast/fixing-fast. Using gcc 4.8 and C++11. Using fairly modern Linux: 3.10

3 Scaling a distributed db the right way

Doron Levari, ScaleBase? CTO

3.1 Why consider scale?

Consider: linear growth in user base, linear growth in each user's data --> quadratic growth. But for many startups the growth rates in each are far beyond linear, so dbs for new companies/projects tend to explode in size.

Good distributed systems book: M. Tamer Ozsu, Principles of Distributed Database Systems

3.2 Desired features in distributed db

Trusted (good reputation), Large tools-ecosystem, plentiful skills in the community, wide compatibility, smooth migration, minimized challenges and limitations, minimized magic (unexplained behavior makes things hard for dbas)

3.3 What is the right way?

- NoSQL -- NoProblem?: doc-level storage, no joining to retrieve whole record
- RDBMS -- tuples, relations make data dist more complicated

Right way depends on data itself, user habits, application Want to distribute data such that the sessions are distributed evenly.

For sharded RDBMS, choose a distribution key column, and many things work, but what if we want to look up by another column? We can (1) denormalize, or (2) do pre-lookup.

4 Oracle MySQL 5.7 DMR4 improvements

Downloadable at labs.mysql.com DMR: release candidate quality, already multiple code reviews.

4.1 Changes

4.1.1 Parsing refactored

new SQL parser: true bottom-up parser. Previous version mixed up optimizer with parsing

4.1.2 GROUP BY has implicit default ORDER BY

would like to remove implicit ORDER BY--give more flexibility to execution.

4.1.3 GROUP BY w/non-aggregate cols in SELECT

would like to return error-- results sensitive to implementation row order

4.1.4 EXPLAIN format changed

More descriptive, more useful

4.1.5 Deprecation of \N alias for NULL

Only deprecated for SQL queries. "SELECT NULL IS \N" should fail. \N still okay for LOAD DATA INFILE

4.1.6 Deprecating password encoding

Switching to AES-256.

5 MySQL Extreme Performance BoF

Lots of interesting notes and tidbits

5.1 Percona tools highly recommended

People very impressed with Percona's debugging/administrative tools for debugging and diagnosis. "Percona toolkit is amazing"

Example tools:

- Percona TCP playback : replay TCP stream
- pt-stalk : collects data upon trigger. "its god-like in its capability"

works on older, non-Percona mysqld as well.

5.2 Logging mysql

Recommend saving incoming tcp stream to mysqld rather than query log. Percona tools can extract query stream from a saved TCP stream. But TCP stream includes much more than just queries--also includes things that cause load, but are not SQL queries. Also, using tcpdump is almost no-load on server-- can redirect output to another machine to save to disk, so impact on server is very, very low. Lower impact than general query log.

5.3 Tuning

When tweaking cache numbers, convince yourself that incremental additions are helpful (Tim Callaghan (Mark's brother) "convince yourself that the extra GB of cache matters"). Numbers too big can have penalties.

Usually recommend Direct-IO rather than buffered, but on encrypted filesystems, direct will kill you because of whole-block encryption.

Not sure about setting fadvise

Which fs? no obvious performance difference between xfs and ext4. But xfs is a big win over ext3.

Mysql is not doing filesystem hole-punching (marking regions of file unused--relies on sparse-file support), not yet, anyway.

Compiler optimizations make a difference. One guy using a Gentoo Linux system with Intel Xeon, compiled everything (OS, libs, everything) with Intel compiler and sees a 20-22% improvement in performance "across the board", meaning every application (including mysql), compared to generic linux binary distro.

5.4 Sharding

Three questions to ask when deciding to shard:

- 1. How am I going to shard (hash, range, list)?
- 2. How do I add shards, remove shards, or split shards?
- 3. How do I move rows from one shard to another.

5.5 Cloud/in-house

Guy who used to work at amazon: we had to justify use of non-AWS resources. But it was easy to show db performance differences when implementing IMDB between AWS and bare metal. (Aside: IMDB data set is a really nice public data set for testing, benchmarking, etc.)

Advantage of using AWS: It forces you to be repeatable from day one, so you make things automated and are used to adding/removing nodes for scalability from the beginning. Big win. But everyone is frustrated with the AWS performance (comparing AWS instance with similar bare hardware specs).

Everyone in session doing bare-metal mysqld

- VM imposes too many context switches, also have to deal with 2 I/O schedulers.
- Use "noop" or "deadline" I/O scheduler. Don't use "cfq" ("it sucks")
- But recommend "MySQL Sandbox" for running multiple mysqld on single node. Much easier than figuring out how to isolate them manually.

Rick James (yahoo guy): 20% overhead from using VM. If you want to isolate, kernel cgroups good enough to prevent runaway processes, trampling.

Most believe that multitenant fires (problems) from dbms on VMs make the VM approach just not worth it for percieved cost efficiency--spend too much time diagnosing performance/stability problems, which happen regularly. Don't colocate db on same instance--too hard to diagnose and isolate problems.

5.6 Storage

Direct-attach flash on PCIe is best. Do not fill SSD. "performance dies at 70%". Thus, you should make your fs partition 70% of the full size, and never touch the leftover 30% with data

Enterprise SSD vs Consumer SSD: sometimes 20% difference in overprovisioning--big difference when you fill up the drive and are constantly writing.

Always RAID10 on spinning disk. RAID5 only for backups, because it's terrible on failure. One guy says there is "never a good reason to use RAID5".

one guy: using LVM costs 20-40% of I/O performance. (Jeremy: LVM is worth it solely because of the online snapshotting. Who cares about the 20%. Online volume snapshots mean easy replication of instances to more machines. No server load except for I/O--contrast with mysql dump/replication.)

6 Thursday Panel Mysql at scale

Peter Zaitsev(Percona), Nisha Talagala (Fusion-io), Robert Hodges (Continuent)

MySQL scale?

- R: Of course the relational model scales, contrast with Stonebraker
- N: Hardware has enabled scale
- P: Misconception--search for silver bullet is misguided. Scale always takes special care, planning, and effort, although the hardware/storage tech makes small/medium scales more painless and fun.
- P: used to be bottlenecked by storage I/O, but SSD is not so fast that mysql can't keep up. So software work is needed to exploit faster storage. More work is needed in both single query execution speed and parallel execution.

- R: Mysql is being used as always-on appliance/mainframe

Growth of mysql

- P: Mysql is unstoppable; but need more embracing of flash technology: see lots of misconception. Need better networking, 1Gbps is not

enough. Also, should run latest software (mysqld, OS).

- R: Want database automation in OpenStack? like RDS on Amazon
- N: System needs to be balanced. Also need predictability. Need predictable impacs from failure--otherwise can't scale.

How has 5.6 impacted?

- R: Panic attack because 5.6 changed binlog; customers wanted performance improvement from 5.6 from first day of availability.
- N: Customers want to make sure that hardware works with 5.6.
- P: It's taken some time to upgrade. 5.5 is working well for people--MySQL is mature, so people are comfortable iwth it.

5.7?

- N: Anything improving concurrency is good for flash.
- P: We see that Oracle has a closed dev process, but are pleasantly surprised with new features; Oracle accepted suggestions from MySQL

Connect; Oracle takes community patches. Excited about GIS features, pluggable parser.

- R: We appreciate the HA features, schema changes, rolling upgrades, improvements in replication; global tid.

In 2-3 years, what changes would help the community?

- P: Have seen a great reduction in hostility in MySQL community. Really need to see it continue to be more accepting and welcoming. Competition has been great.
- R: Not much change is needed: community is respectful and yet competitive. But High Availability needs to be improved--amazed at how

much comes out-of-the-box in MongoDB. Switched to Apache licensing to be more accessible.

Impact of OpenStack on operating MySQL at scale?

- R: Although AWS is big in public cloud, still a lot of resistance, and OpenStack? is good there. OpenStack? needs: Cinder block storage doesn't have provisioned IOPS
- N: Usability is still much needed for OpenStack?
- P: OpenStack? still has a long way to go; community likes open-source if it's good enough. Cinder block storage is not good enough, but can use local high-performance storage.

Competition among forks?

- P: Closer you are to customers, easier to understand problems and implement solutions quickly. Lots of collaboration among engineers at competitors. Webscalesql is vendor-neutral--this is exciting.

Final thoughts on scale?

- N: Efficiency, predictability, manageability: Improvements in any of these help scalability.
- P: Hardware keeps getting better at the single node; software gets better at scaling, and scaling people. Maturity--MySQL has been used

at extreme scale for 5-10 years already.

- R: Improvements in the last two MySQL releases have been amazing, so don't be afraid to upgrade. Automation has gotten much better, very important.

7 Building Data Warehouse with Hadoop and MySQL

Zburivsky Danil (Pythian)

Columnar file formats:

- RCFILE
- Parquet
- ORC

Beats music streaming arch:

```
Mobile/Desktop ------> Kafka cluster
```

```
 \                          \
Solr                     Camus
 \                          \
mysqld --------------->  Hive

                        Hadoop
```

8 Mariadb 10 storage engines, Sergei Golubchik (SkySQL)

Several plugins available for monitoring, performance-analysis (cache info, locale, metadata lock, query response, server audit)

8.1 Cassandra storage engine

- flexible schema, dynamic columns
- several mariadb nodes --> cassandra ring
- auto-replicated fault-tolerant tables

8.2 Connect storage engine

- "CSV & federated"
- CSV, BIN, FMT, INI
- ODBC, MySQL
- one row per key-valuye pair
- can also append
- remote mysql query results.
- table transformations: XCOL, PIVOT

8.3 OQGraph v3

- create links table(edges)
- create graph table
- Now can find shortest path, all reachable nodes.

8.4 Sequence table engine

Virtual pseudo tables: "SELECT seq FROM seq_1_to_1000;

8.5 Spider storage engine

Federates multiple mysql nodes together.

Can push down parts of query to spider nodes, but not in the current release.

9 Lightnings

Rick James (Yahoo): mysql.rjweb.com optimizing neighbor search: O(N) to O(1)
?http://mysql.rjweb.org/doc.php/latlng

User aggregation rollups: estimate using bitstring, set bits according to hash in bitstring, inflate for collisions --> 2% error

10 Notes from Jeremy

Currently--adding full on-disk encryption, enhancing ruby innodb tool innodb page file vizualizer

11 Side notes

Most people are very negative about MyISAM: "So you don't care if you lose data?" "MyISAM corrupts data all the time, what if you crash during an update." Sunny Bains (Oracle MySQL mgr) wants to drop support for MyISAM. Don't believe that MyISAM is better in any way. Wants to try our tests. Jeremy thinks that if they can turn off the transaction updating, InnoDB should be competitive in table scanning. Jeremy also thinks that InnoDB is space-overhead-competitive, but notes that tree node fragmentation is a killer and almost guaranteed when rows are inserted in random order (his tool shows that most tree nodes end up half-full if insertion is random, but much better if insertion is pkey-ordered).

Replication is big. very big. Several vendors for replication solutions between mysqld and also other dbms (and also hadoop)