The S3 storage engine in MariaDB 10.5

Michael Widenius
CTO @ MariaDB

S3 storage engine in MariaDB 10.5

- Read only tables (perfect for inexpensive archiving or sharing of old data).
  - For some organizations S3 storage is cheaper than local storage
- Data and index can optionally be compressed on S3
- S3 works with MariaDB partitioning for flexible handling of multiple tables
- Very fast, thanks to reading of blocks in big chunks (4M by default)
- Supports all key formats and optimizations that Aria supports
- Data can be accessed by multiple MariaDB servers. Tables are automatically discovered when needed.
- S3 has it's own configurable page cache
- S3 is also backported to 10.3 and 10.4 MariaDB enterprise server

MariaDB

What is Amazon S3

- Storage of files in the cloud
- Works trough http requests (think get/put of whole files)
- Supports basic functions like list, copy to/from and delete
- Move and rename are implemented as copy + delete of the whole file (not fast)
- Quite slow for small files. Optimal file size for retrial is said to be around 4M.
- Many vendors support storage of files according to the ‘S3 interface’
- Not really usable for databases who needs to update blocks withing a file.

MariaDB

S3 storage

- All files are stored in one of several ‘buckets’
- Files are stored as objects in a bucket accessed with a key.
- The key may contain ‘/’ and commands like ‘aws ls’ make S3 storage to feel like a file structure, even it it really isn’t.

MariaDB

Converting a table to S3

Converting a table to S3:

```
ALTER TABLE old_table ENGINE=S3
```

And converting back to local:

```
ALTER TABLE table_on_s3 ENGINE=InnoDB
```

Internally the data is first copied to a local Aria table with ROW_FORMAT=PAGE TRANSACTIONAL=0 and then moved to S3.

MariaDB

Converting a table to S3

One can also use 'aria_s3_copy' to:

- Copy Aria tables of type "ROW_FORMAT=PAGE TRANSACTIONAL=0" tables to S3
- Copy S3 tables to local storage
- Delete tables on S3

((/my/maria-10.5/storage/maria)) aria_s3_copy --help
aria_s3_copy Ver 1.0 for Linux on x86_64
...
Copy an Aria table to and from s3
Usage: aria_s3_copy --aws-access-key=# --aws-secret-access-key=# --aws-region=# --op=(from_s3 | to_s3 | delete_from_s3) [OPTIONS] tables[.MAI]

MariaDB

ALTER TABLE options for S3

```
ALTER TABLE old_data ENGINE=S3
S3_BLOCK_SIZE=#                          // Default 4M
COMPRESSION_ALGORITHM=none|zlib          // Default none
```

Both index and data are compressed. Typical savings up to 70%

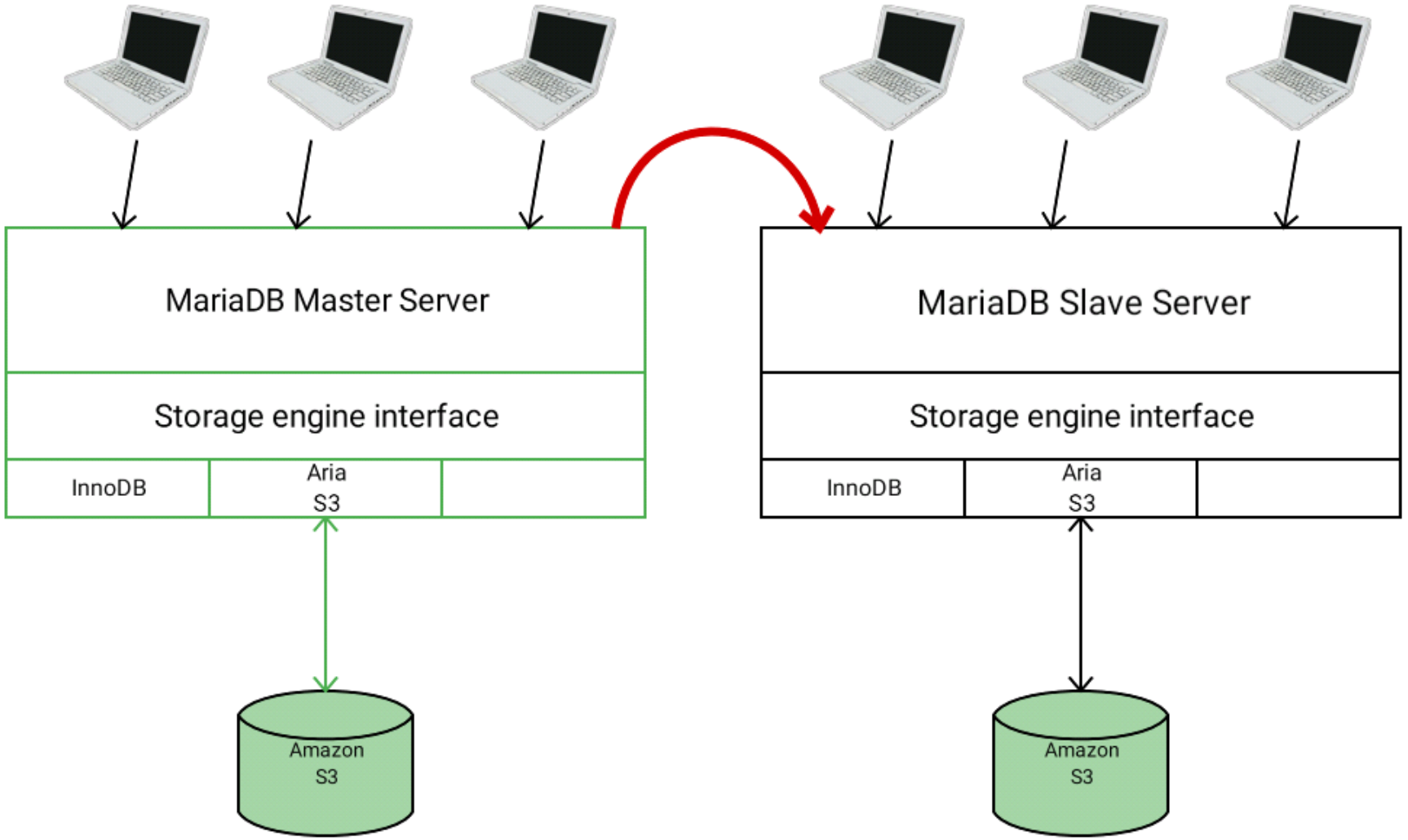
MariaDB

Setting up S3 storage engine

Add to your my.cnf file something like:

```
[mariadb-10.5]
s3=ON
s3-host-name=s3.amazonaws.com
s3-bucket=mariadb
s3-access-key=xxxx
s3-secret-key=xxx
s3-region=eu-north-1
#s3-slave-ignore-updates=1
#s3-pagecache-buffer-size=256M
```

Different S3 storage

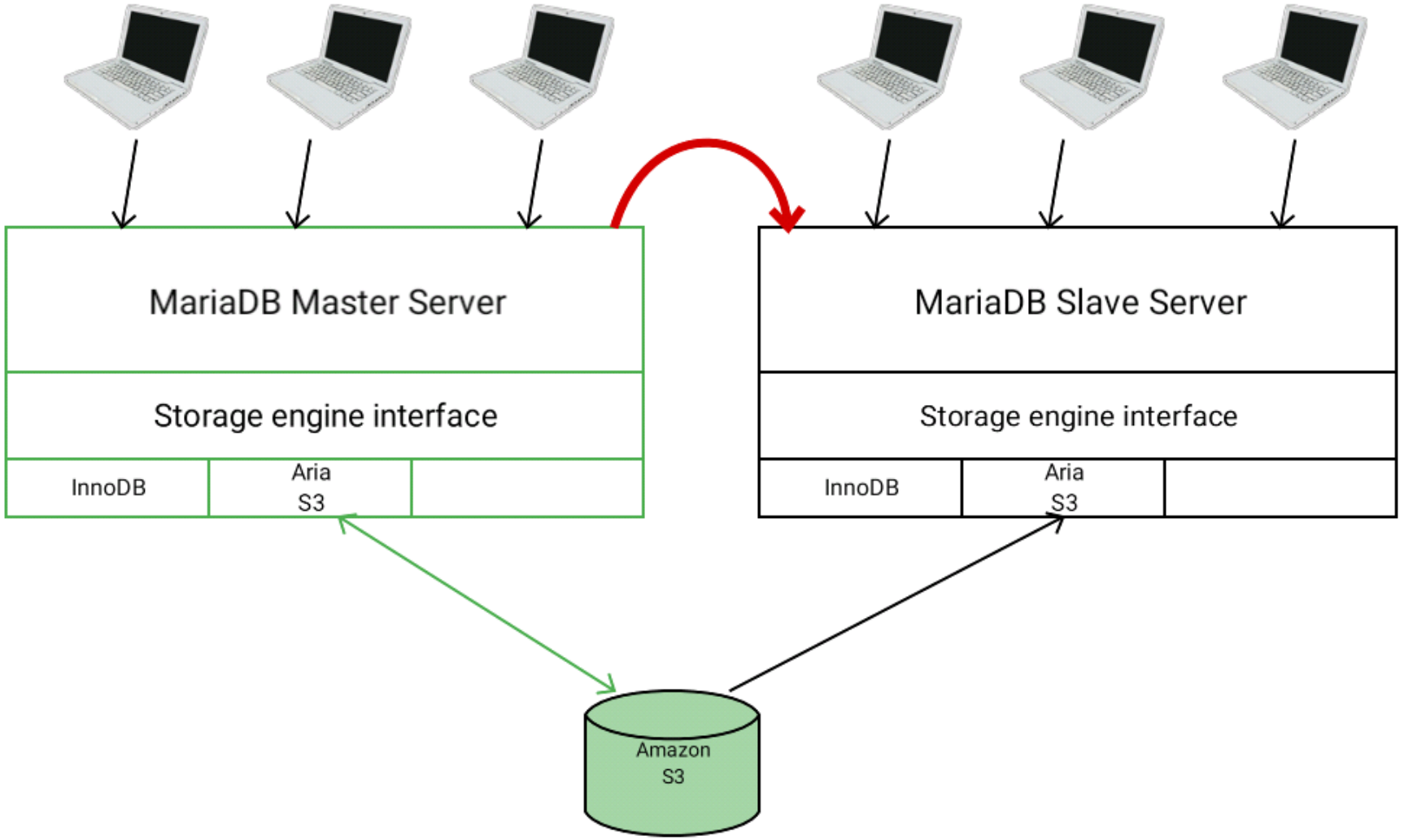


MariaDB

Replication with different S3 storage on master and slave

- In this case the normal replication works.
- The slaves should be able to duplicate any command related to S3 on the master.

Shared S3 storage



MariaDB

Replication with same S3 storage on master and slave

In this case the slave cannot repeat any commands related to S3 as the master has already executed these. Instead the slave has to do:

- All CREATE TABLES of S3 tables should be ignored
- All updates to S3 tables should be ignored (only ALTER is possible)
- ALTER TABLE to S3 should be ignored
- DROP TABLE of a S3 table should only remove any local .frm definition, not touch the S3 data
- RENAME of S3 tables are replicated as RENAME IF EXISTS so that the slave can ignore the rename (as the table is already renamed in S3). Because of this, RENAME IF EXISTS was implemented in 10.5

Replication with same S3 storage on master and slave

On the master:

ALTER TABLE s3_table ENGINE=InnoDB should be replicated as DROP TABLE IF EXISTS followed by CREATE TABLE local_table and the data copied to binary log.

MariaDB

Setting up replication with same S3 storage on master and slave

The master can't know if any of the slaves may will use same S3 storage as the master. Because of this, the following option was introduced:
--s3_replicate_alter_as_create_select (defaults to on)

If the slave is using same storage as master, then the slave should set the option:
--s3_slave_ignore_updates (defaults to off)

Interface used to connect to S3

I first tried to use the AWS recommended interface (aws-dk-cpp.git):

- Lots of templates and templates on top of templates
- Lots of code!
- Hard to work with binary files (optimized for reading line by line or block by block)
- VERY hard to use for what was needed:
  - Reading or writing a complete file
  - Checking if files existed
  - Getting list of files

libmarias3

We decided instead to create our own layer for accessing S3

- Uses libcurl and libxml2 internally
- Developed by Andrew Hutchings as an independent project
- Released under GPL2
- Simple and efficient API:
  - ms3_list(ms3, bucket, prefix, &list)
  - ms3_put(ms3, bucket, key, data, length)
  - ms3_get(ms3, bucket, key, data, &length)
  - ms3_delete(ms3, bucket, key)
  - ms3_status(ms3, bucket, key, &status)

libmarias3 internals

- libmarias3 provides the libmarias3 API, builds the headers and HTTPS requests.
- Curl handles the HTTPS request / response, including TLS (transport layer security)
- libxml2 parses the XML response from the list commands (think ls)

Storage layout on S3

frm file (used by discovery):
s3_bucket/database/table/frm

First index block (contains description of the Aria table):
s3_bucket/database/table/aria

Rest of the index file:
s3_bucket/database/table/index/block_number

Data file:
s3_bucket/database/table/data/block_number

Example of data stored on S3

```
shell> aws s3 ls --recursive s3://mariadb-bucket/
2019-05-10 17:46:48       8192  foo/test1/aria
2019-05-10 17:46:49    3227648  foo/test1/data/000001
2019-05-10 17:46:48        942  foo/test1/frm
2019-05-10 17:46:48    1015808  foo/test1/index/000001
```

MariaDB

Implementation of S3

The S3 storage class (ha_s3) inherits from the Aria storage engine.
It uses two external libraries: libmarias3 and zlib.

The changes need in the Aria storage engine where quite small:

First commit (almost all code, except replication and the libmarias3 library):
git show --stat ab38b7511bad8cc03a67f0d43e7169e6dfcac9fa
...
 66 files changed, 4390 insertions(+), 212 deletions(-)

Implementation of S3

```
(/my/maria-10.5/storage/maria) wc *s3*.*
  330  1109  10336         aria_s3_copy.cc
  810  2383 24753      ha_s3.cc
   71   269  2048      ha_s3.h
 1458  3967 41178     s3_func.c
  118   507  4965      s3_func.h
   56   139  1060     test_aria_s3_copy.sh
 2843  8374 84340 total
```

The two main Aria files that where changed where:

storage/maria/ma_open.c	257 (147 non space changes)
storage/maria/ma_pagecache.c	414 (390 non page changes)

MariaDB

Implementation of S3
ha_s3.h

```
class ha_s3 :public ha_maria
{
  bool in_alter_table;
  S3_INFO *open_args;
public:
  ha_s3(handlerton *hton, TABLE_SHARE * table_arg);
  int create(const char *name, TABLE *table_arg, ...);
  int open(const char *name, int mode, uint open_flags);
  int write_row(const uchar *buf);        // Only usable by ALTER TABLE
  void drop_table(const char name) {}  // Only used for internal temporary tables
  int delete_table(const char *name);
  int rename_table(const char *from, const char *to);
  int discover_check_version();
  S3_INFO *s3_open_args() { return open_args; }
  void register_handler(MARIA_HA *file);
};
```

MariaDB

Implementation of S3
Aria changes

ma_open.cc

- If called by S3 engine, read header from S3 instead of from file
- Read specific S3 options from the index file header (block size, compression etc)

ma_pagecache.cc

- Internally both Aria and S3 tables are stored in blocks of aria_block_size (8K)
- Added support of reading big blocks. When reading an index or page block in the middle of a S3 block, read the corresponding S3 block and update all pages read in the dedicated S3 page cache.

Future work

- Store aws keys and region in the mysql.servers table (as Spider and FederatedX). This will allow one to have different tables on different S3 servers.
- Add more compression options (lz4, snappy?)
- Add shared cache for S3 connections instead of having one per open table
  - Will use less memory and speed up opening new S3 tables.

Under consideration:

- Allow batch updates (single user) to S3 tables.
  - For adding new archive data to an existing S3 table instead of having to alter the table back to local, update the table and copy back to s3.

Conclusions

- The S3 storage engine is very good solution if you want to:
  - Archive/backup old data
    - Using partitioning you can store old payment transactions, one table per month, in S3 for fast access to big data. Using partitioning makes it trivial to add more data without having to change the old S3 tables.
  - Share read-only, directly accessible, tables with other users.
  - Utilize cheaper storage of your old data that doesn’t change anymore.

That’s all folks

QUESTIONS ?

The S3 engine is documented in detail at:
https://mariadb.com/kb/en/s3-storage-engine/

MariaDB

Thank you