Alibaba Cloud

Worldwide Cloud Services Partner

Percona XtraBackup at Alibaba Cloud

Bo Wang

Alibaba Cloud

About me

- Bo Wang (Fungo Wang)
- Hangzhou, China
- Joined Alibaba Cloud at Apr 2014 after got Master's in CS at Zhejiang University
- Senior Engineer at Alibaba Cloud, develop and maintain AliSQL, TokuDB, XtraBackup, PolarDB

Agenda

1. ApsaraDB on Alibaba Cloud
2. How we use XtraBackup
3. How we improve XtraBackup

Agenda

1. ApsaraDB on Alibaba Cloud
2. How we use XtraBackup
3. How we improve XtraBackup

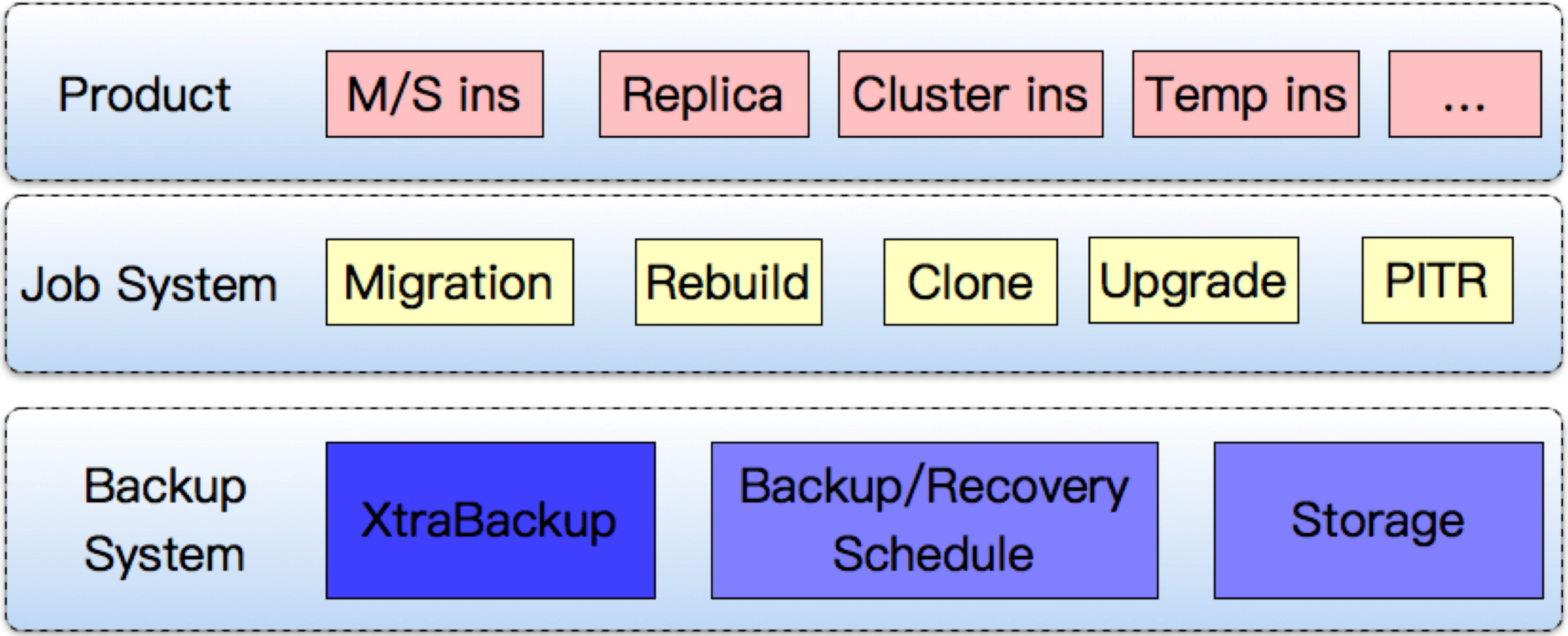
ApsaraDB on Alibaba Cloud

Database As A Service, for your data safety, for your application stability

- RDS for MySQL 5.1 (deprecated)
- RDS for MySQL 5.5
- RDS for MySQL 5.6
- RDS for MySQL 5.7
- RDS for MariaDB (10.3)
- RDS for MySQL 8 (expected in 2019)

ApsaraDB on Alibaba Cloud

Backup is a fundamental facility, it's a basic requirement for our database products.



Product: M/S ins | Replica | Cluster ins | Temp ins | ...

Job System: Migration | Rebuild | Clone | Upgrade | PITR

Backup System: XtraBackup | Backup/Recovery Schedule | Storage

Agenda

1. ApsaraDB on Alibaba Cloud
2. How we use XtraBackup
3. How we improve XtraBackup

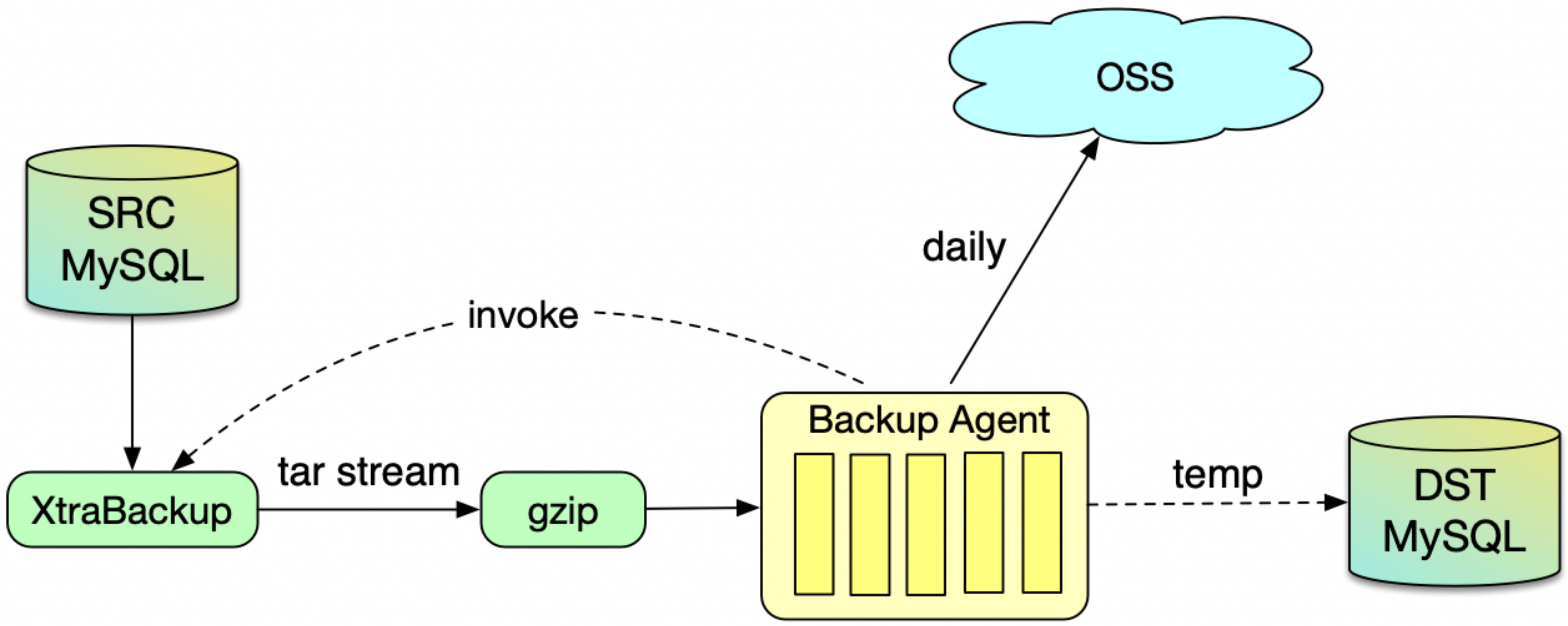
Backup Types

Our MySQL instances can be provisioned on physical machines or ECS VMs.

- Physical backup, used for physical machine instances (XtraBackup)
- Cloud disk snapshot, used for ECS VM instances (disk snapshot)
- Logical backup, an additional product feature available on user portal (mysqldump)

Backup Strategies

- Full backup (incremental backup is in plan)
- Backup regularly on daily base, the cycle is configurable
- Stream backup, no intermediate temp files on local disk
  - Stream to OSS (Object Storage Service)
  - Stream between hosts, in some migrating/rebuilding scenarios



Backup Strategies

- Backup on slave node by default, can also on master node when slave node is not available/suitable
- Backup result can be downloaded and recovered locally by our customers, not locked by ApsaraDB

Backup/Recover Command

- Backup

```
innobackupex --defaults-file=my.cnf --host=host --user=user --port=port --password=pass --slave-info --stream=tar | gzip | backup_agent stream upload to OSS
```

- Download and extract

```
backup_agent fetch from OSS | gzip | tar xvf -C restore_dir/
```

- Recover

```
innobackupex --apply-log --use-memory=bp_memory_size restore_dir/
```

- Restore

mv files to directories specified in my.cnf

Agenda

1. ApsaraDB on Alibaba Cloud
2. How we use XtraBackup
3. How we improve XtraBackup

Multiple Engines

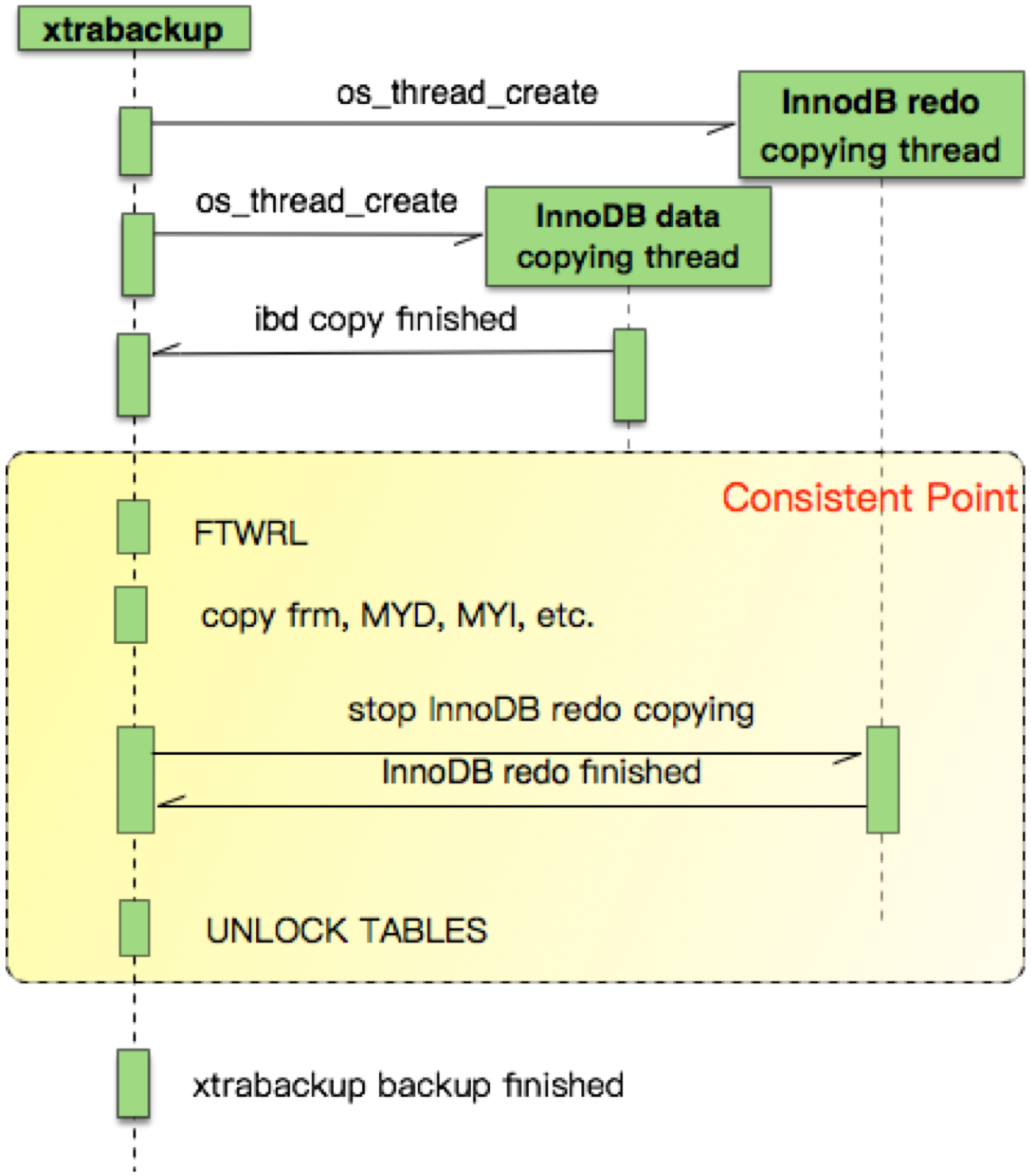
- ApsaraDB provides multiple storage engines for MySQL, RXB can backup data files in all these engines
  - InnoDB
  - MyISAM, CSV, ARCHIVE
  - TokuDB
  - MyRocks

Multiple Engines – Basic Principles

- Backup result must be recovered to a consistent point (binlog pos)
  - Tables inside one storage engine
  - Tables across all storage engines
  - Server layer data (frm, par, etc.)
- Backup should avoid affecting mysqld as much as possible
- Each storage engine has its own characteristics, and should be fully leveraged when design backup solution

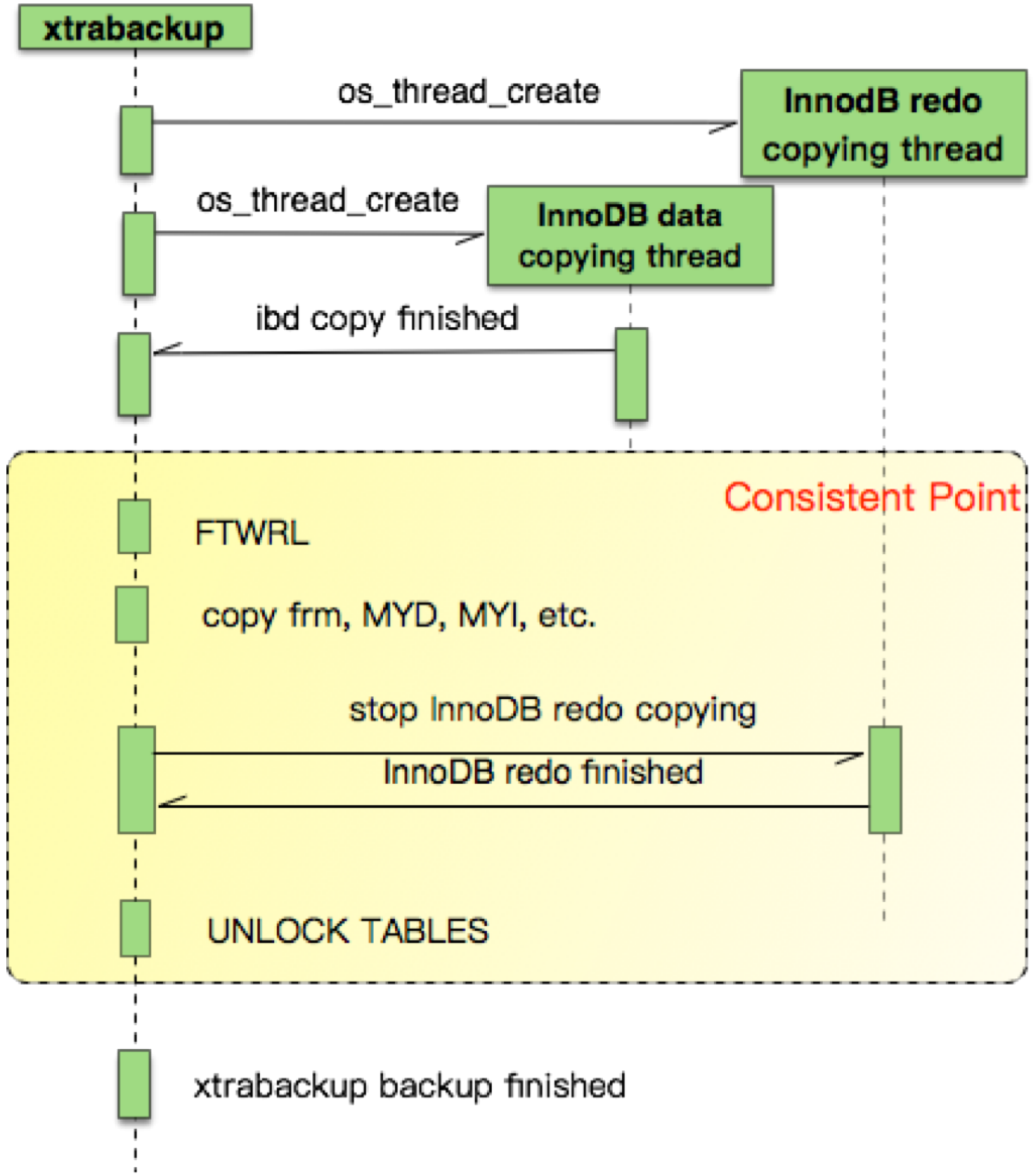
Multiple Engines – Basic Principles

Alibaba Cloud
Worldwide Cloud Services Partner



Multiple Engines – MyISAM

- MyISAM is a non-transactional storage engine, it is simple compared to InnoDB
- No WAL and crash recover process, so the MYDs and MYIs must be in clean/consistent state when copying
- A rough and brute way: freeze MyISAM engine (FTWRL), then copy data
- Simple copy, no need to understand engine detail, and no recover process when prepare

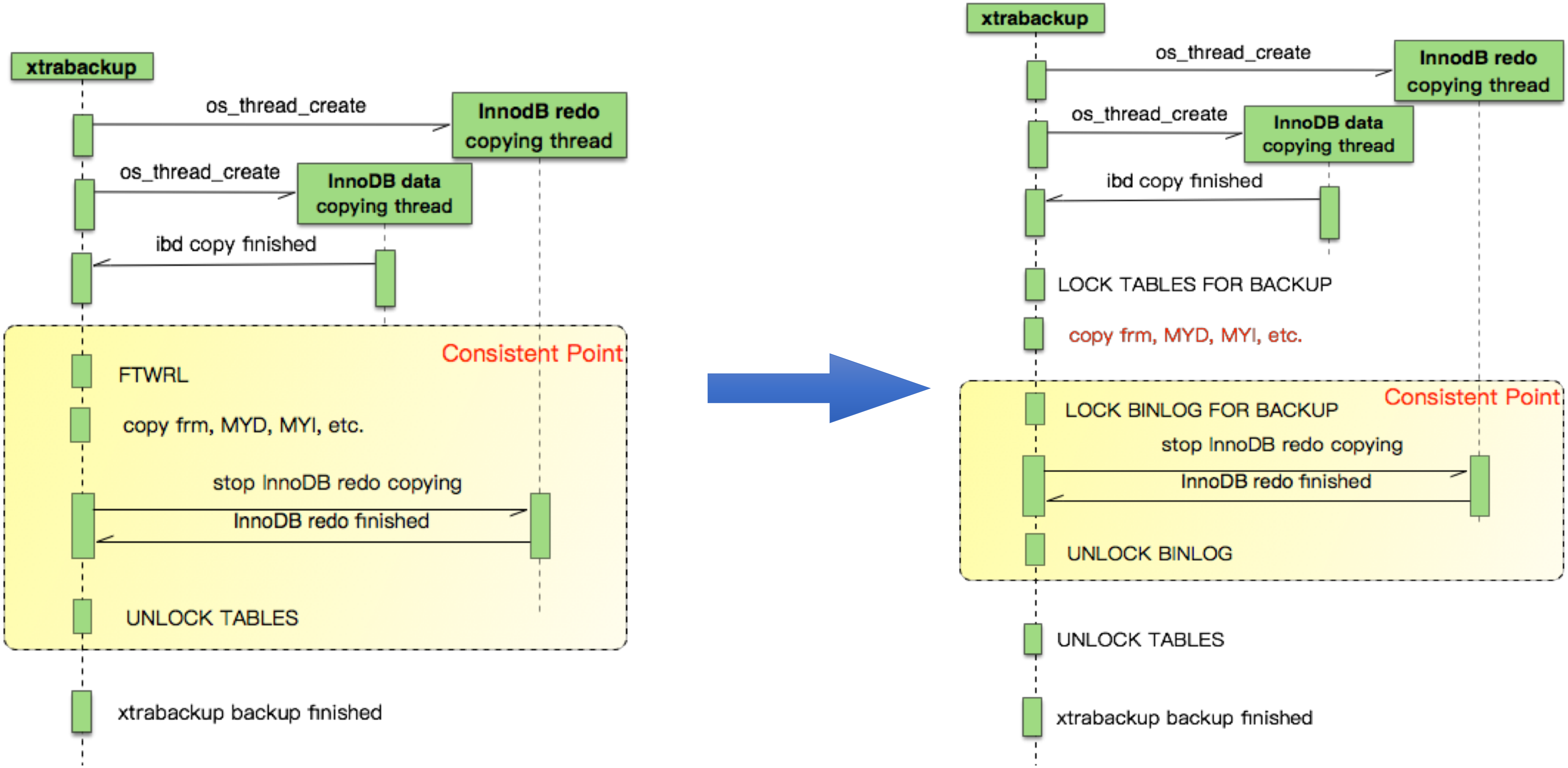


Multiple Engines – MyISAM

- FTWRL is too heavy, all engines are frozen (read only), and all tables are closed (flush).
  - This operation affects all engines, even they do not need it. InnoDB/TokuDB/MyRocks are victims when copying MyISAM files
  - The global lock is only needed to get consistent point

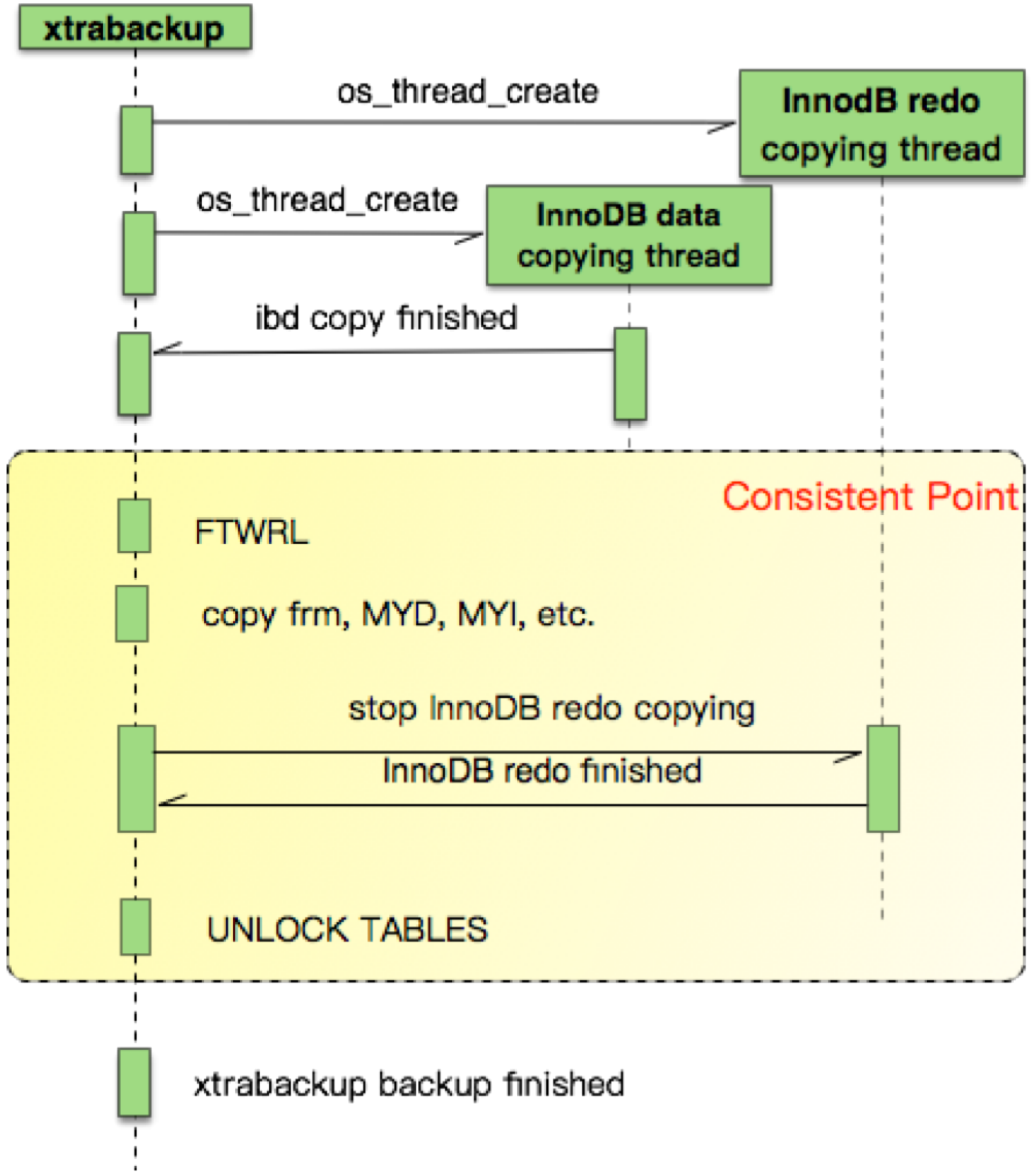
- Use a lightweight way, percona-server has backup locks (MDL):
  - LOCK TABLES FOR BACKUP // block non-transactional IUD and all DDL
  - LOCK BINLOG FOR BACKUP (freezing point) // block binlog position or Exec_Master_Log_Pos advance

Multiple Engines – MyISAM



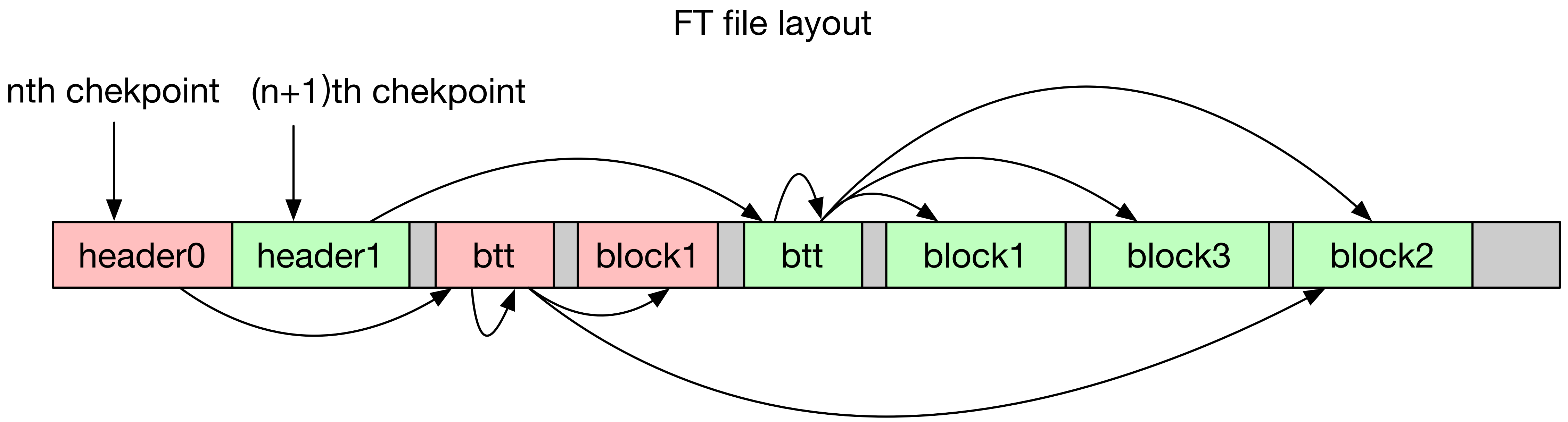
Multiple Engines – TokuDB

- A simple and brute way is treating TokuDB as MyISAM
- Workable, but not acceptable



Multiple Engines – TokuDB

- TokuDB is a transactional storage engine, like InnoDB
- Sharp checkpoint, variable length block, COW at block level
- Use BTT (Block Translation Table) to maintain mapping between block number and block coord(offset, size), BTT is persistent to disk by checkpoint
- Each FT data file contains two copy of data (two BTTs), at least one copy is valid and the data corresponding to the very last checkpoint

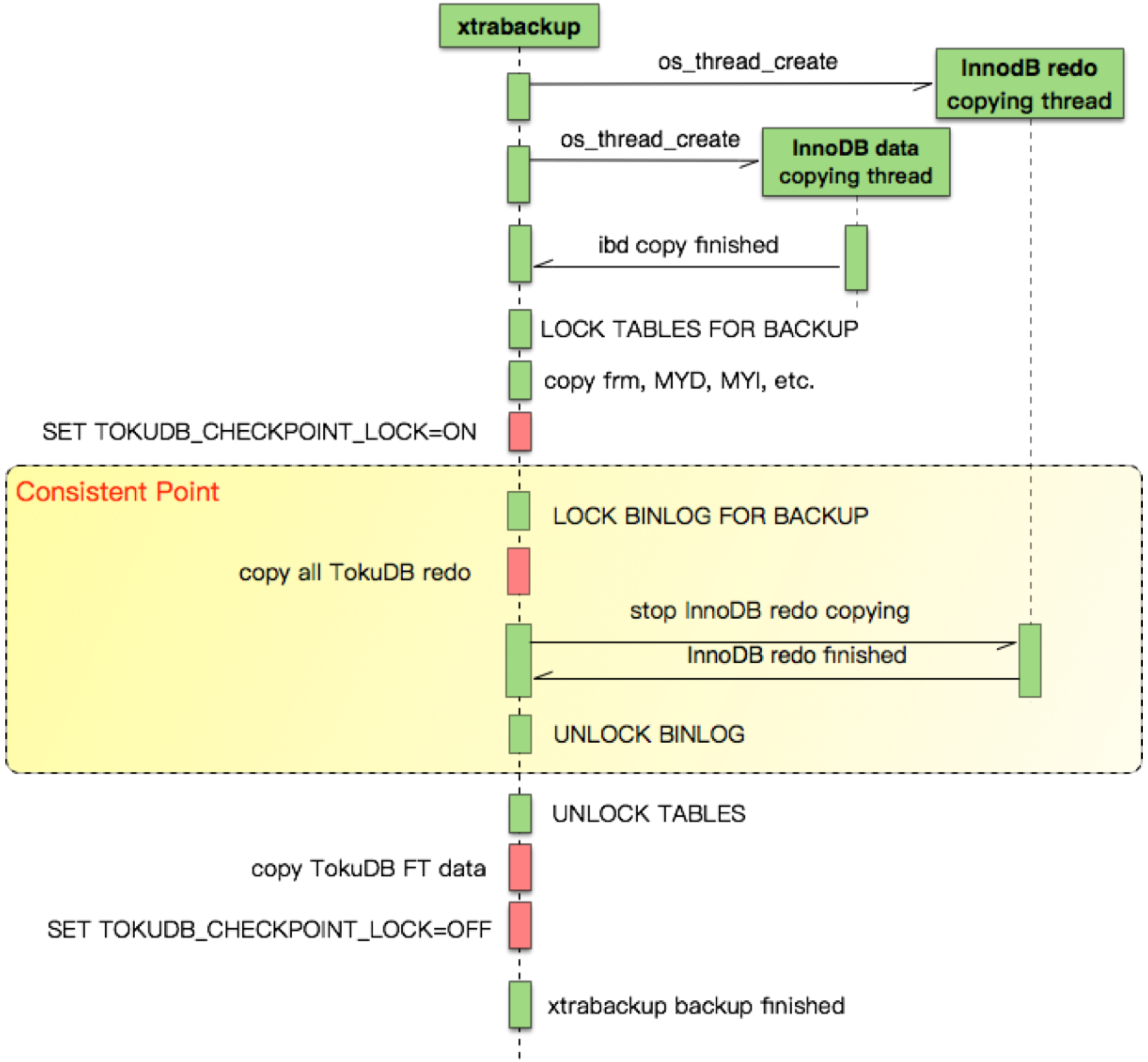


- TokuDB redo log is like binlog, it’s logical log, so the engine data must be in consistent state before applying redo log
- Checkpoint lock can grabbed by user to prevent server from performing checkpoint

Multiple Engines – TokuDB

Alibaba Cloud
Worldwide Cloud Services Partner

- Redo copying finished before copying data
- Use TokuDB sharp checkpoint and COW features, hold TokuDB checkpoint lock while copying TokuDB FT data.
- We may backup many future blocks, TokuDB can't see them when recover, treat them as garbage (unused space). Because checkpoint is blocked, and BTT is not flushed and updated.



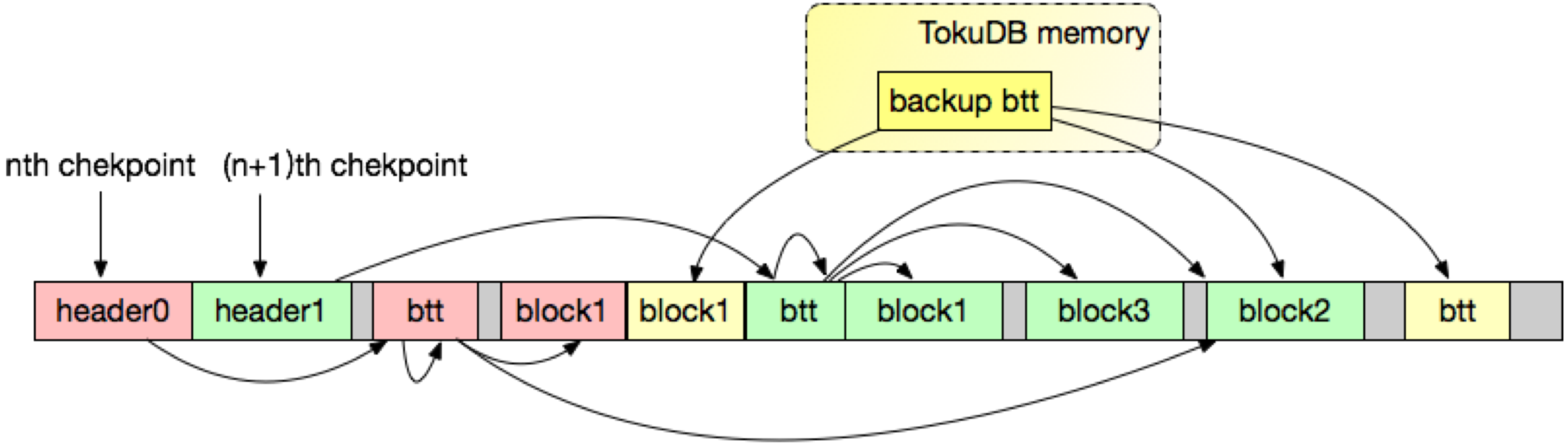
Multiple Engines – TokuDB

- Holding checkpoint lock for a long time may be dangerous
  - Long recover time if crash
  - No checkpoint, no redo log purging, accumulated redo logs will occupy too much disk space

- TokuDB redo logs and FT data files are copied at a coarse level (like MyISAM), RXB do not understand TokuDB format. Redo log recovery is performed by mysqld, not RXB(--apply-log).
  - No validation for redo log entry, and FT block
  - Limiting future feature development

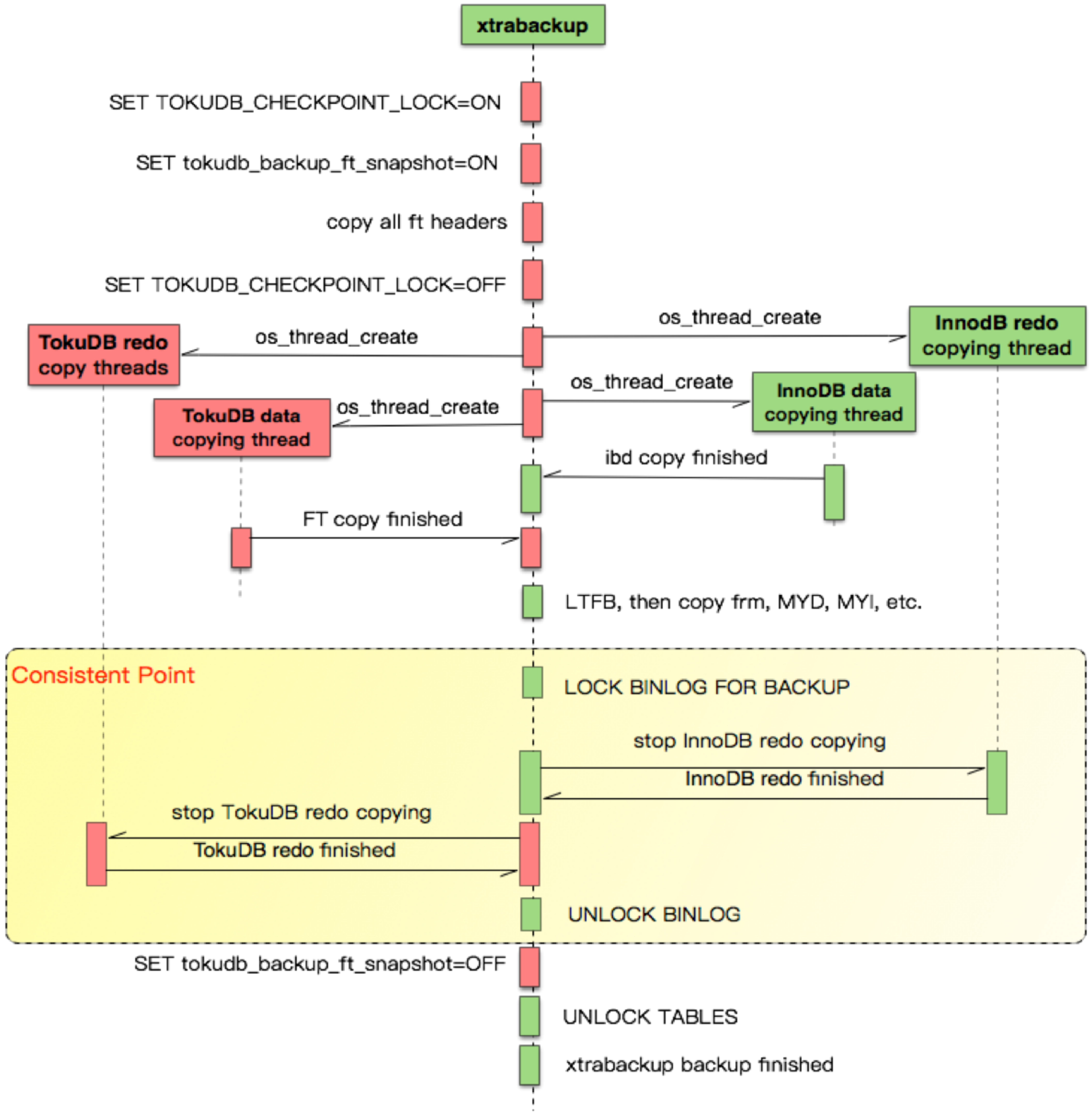
Multiple Engines – TokuDB 2.0

- Checkpoint lock is too heavy, what we need is a FT snapshot. We add a FT snapshot feature to TokuDB to relieve dependence on checkpoint
- Maintain a backup BTT in memory, which is a copy of latest checkpoint BTT, block in backup BTT is protected and will not be free and reused



Multiple Engines – TokuDB 2.0

- Checkpoint lock is also needed, but hold for a very short time.
- TokuDB backup procedure is symmetrical with InnoDB
- TokuDB engine is embedded into RXB just like InnoDB
- Redo log entry is verified
- Only copy necessary FT data
- Redo recover is performed by RXB (--apply-log)

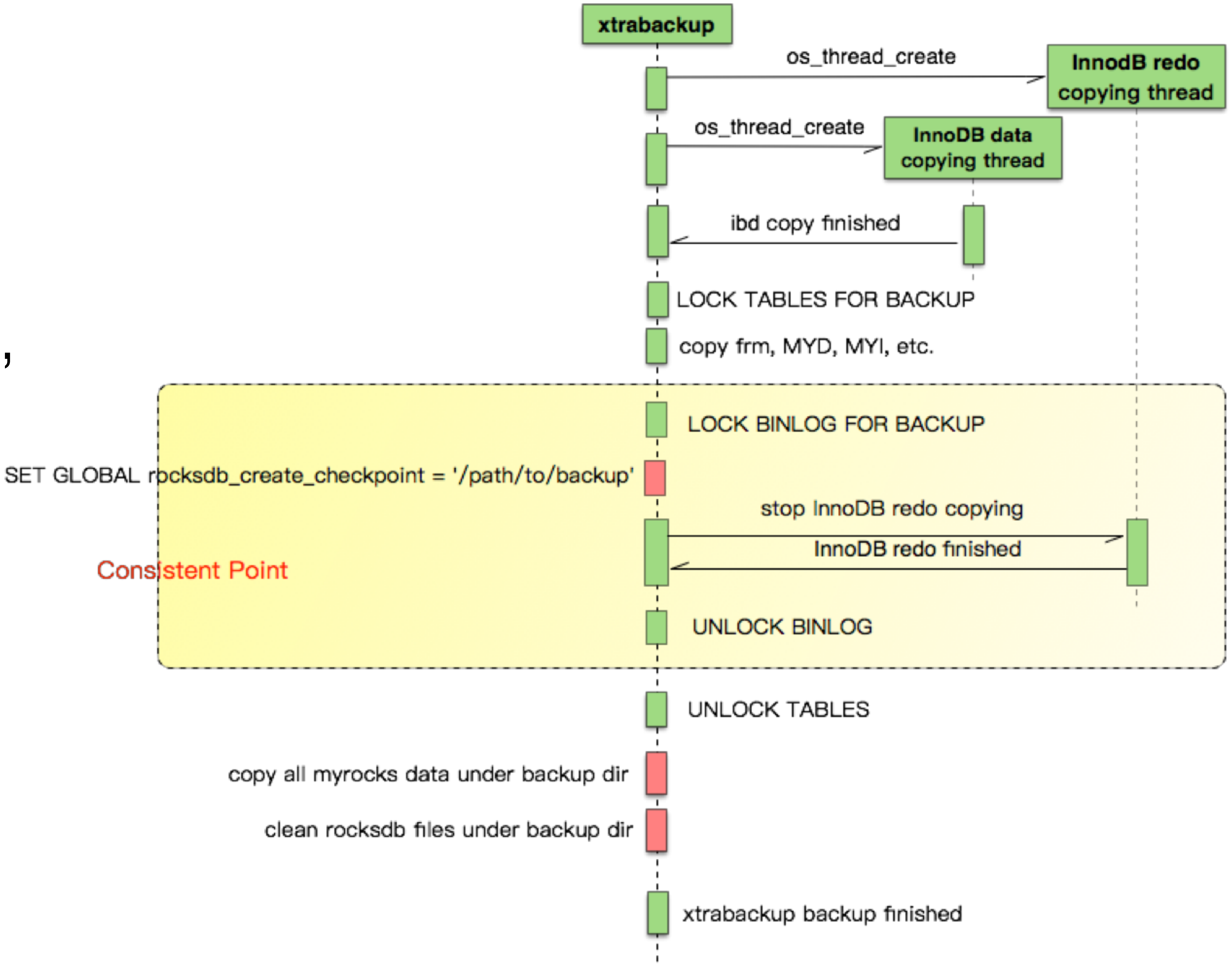


Multiple Engines – MyRocks

- MyRocks is a transactional storage engine, like InnoDB/TokuDB
- COW at file level (SST files), and MyRocks can create a snapshot to a specified dir
- SET GLOBAL rocksdb_create_checkpoint = '/path/to/snapshost'

Multiple Engines – MyRocks

- SET GLOBAL rocksdb_create_checkpoint = ‘/path/to/backup’, to create a snapshot under backup dir, contains MyRocks data, redo log and meta file.
- Currently, MyRocks data is handled at a coarse level, RXB do not understand MyRocks format, recover is performed by mysqld.



Multiple Engines – All in one

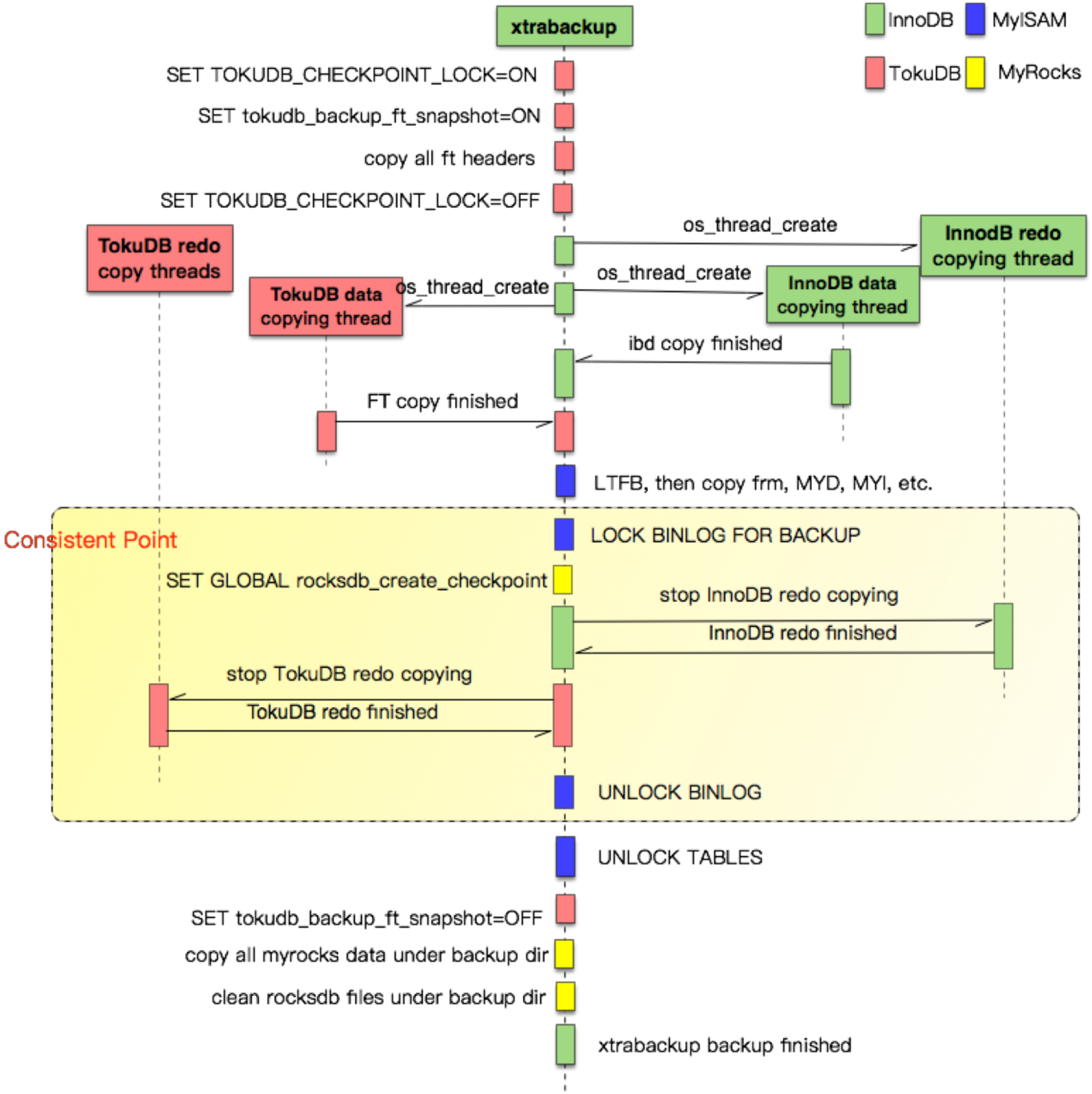

xtrabackup
🟩 InnoDB
🟦 MyISAM
🟥 TokuDB
🟨 MyRocks
SET TOKUDB_CHECKPOINT_LOCK=ON
SET tokudb_backup_ft_snapshot=ON
copy all ft headers
SET TOKUDB_CHECKPOINT_LOCK=OFF
os_thread_create
InnodB redo copying thread
TokuDB redo copy threads
os_thread_create
InnoDB data copying thread
TokuDB data copying thread
os_thread_create
ibd copy finished
FT copy finished
LTFB, then copy frm, MYD, MYI, etc.
Consistent Point
LOCK BINLOG FOR BACKUP
SET GLOBAL rocksdb_create_checkpoint
stop InnoDB redo copying
InnoDB redo finished
stop TokuDB redo copying
TokuDB redo finished
UNLOCK BINLOG
UNLOCK TABLES
SET tokudb_backup_ft_snapshot=OFF
copy all myrocks data under backup dir
clean rocksdb files under backup dir
xtrabackup backup finished

Table Level Recover

- For PITR (Point-In-Time Recovery), the customer may want to recover just a few tables. But the whole backup result file must be downloaded and recovered.
- The time to download backup result take the majority part in the whole recovery procedure, so if we can fetch only the table needed, the PITR will be much faster.

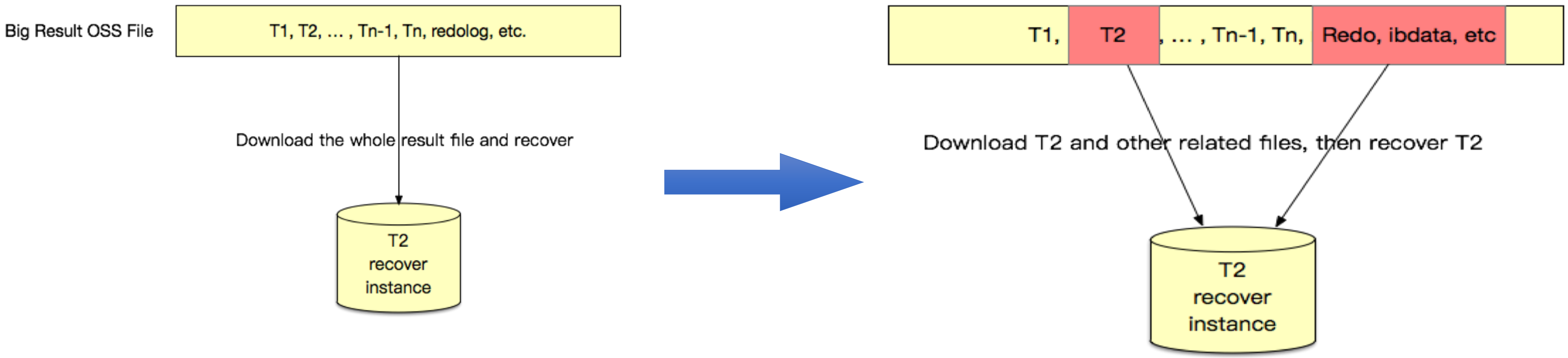

Big Result OSS File
T1, T2, ... , Tn-1, Tn, redolog, etc.
Download the whole result file and recover
T2
recover
instance
T1,
T2
, ... , Tn-1, Tn,
Redo, ibdata, etc
Download T2 and other related files, then recover T2
T2
recover
instance

Table Level Recover

- The OSS file can be downloaded by specifying a position range (begin, end)
- RXB will generate a JSON meta file to expose the detail file organization in the OSS file.

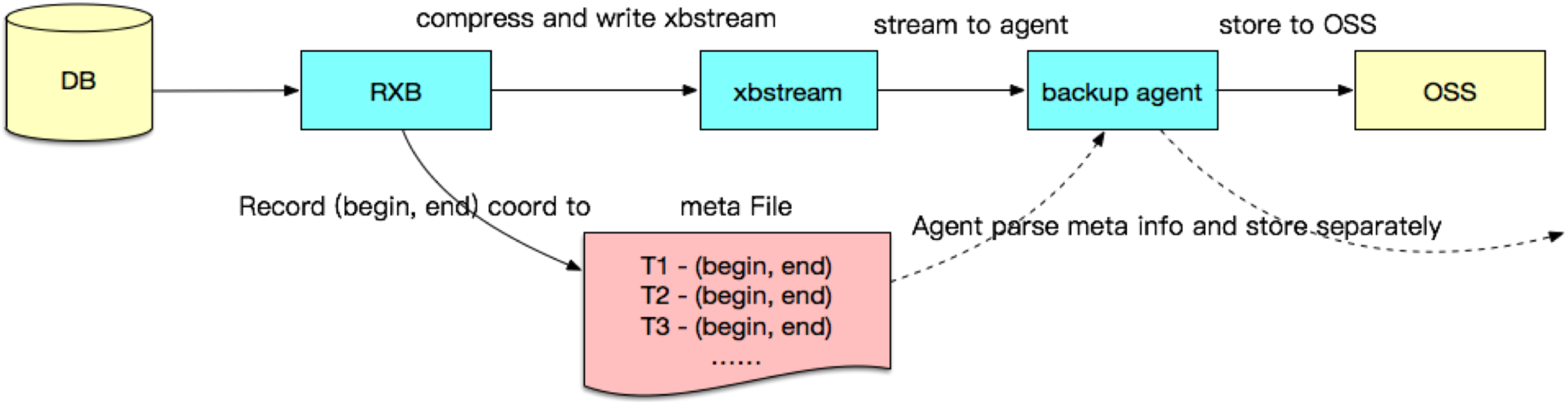
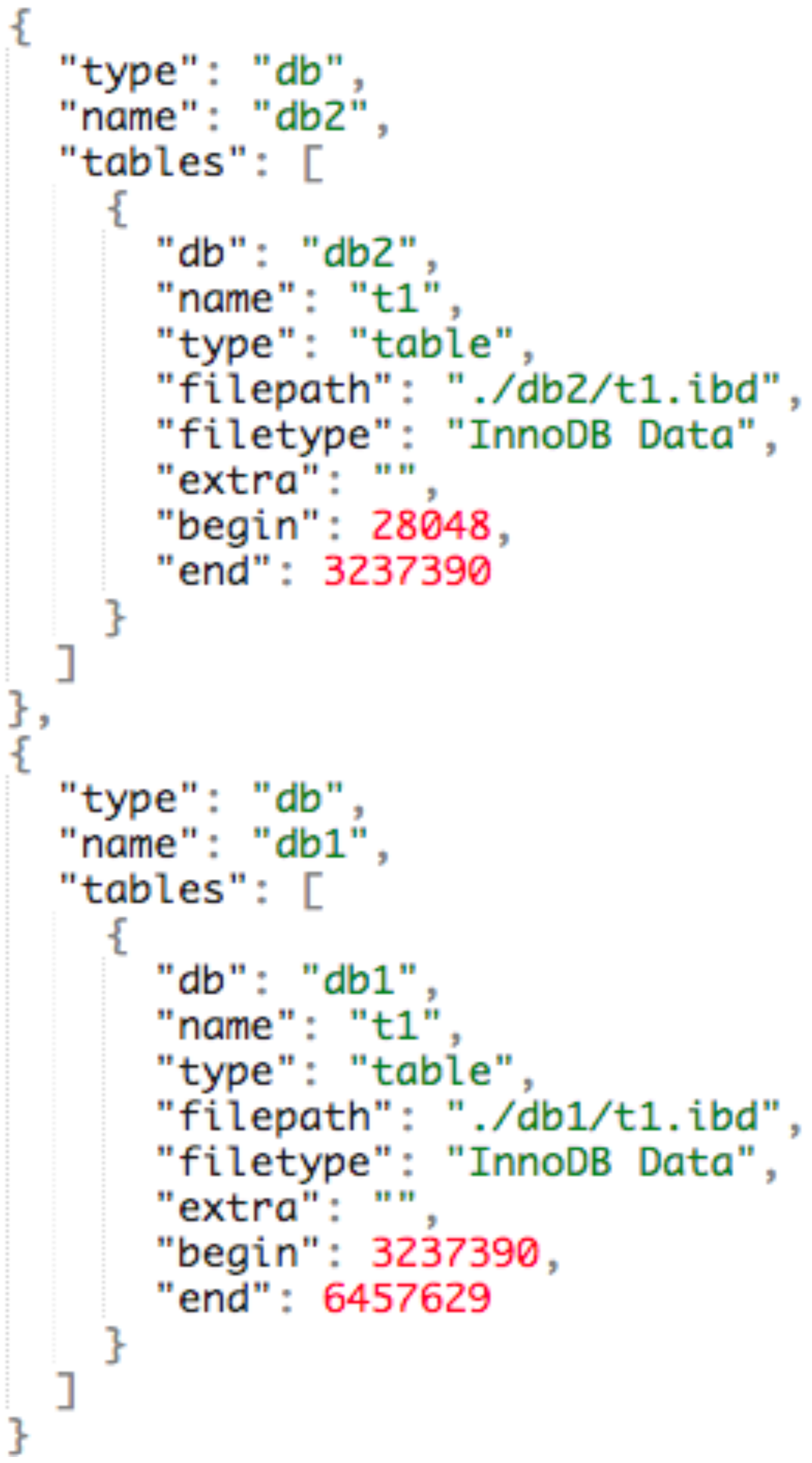
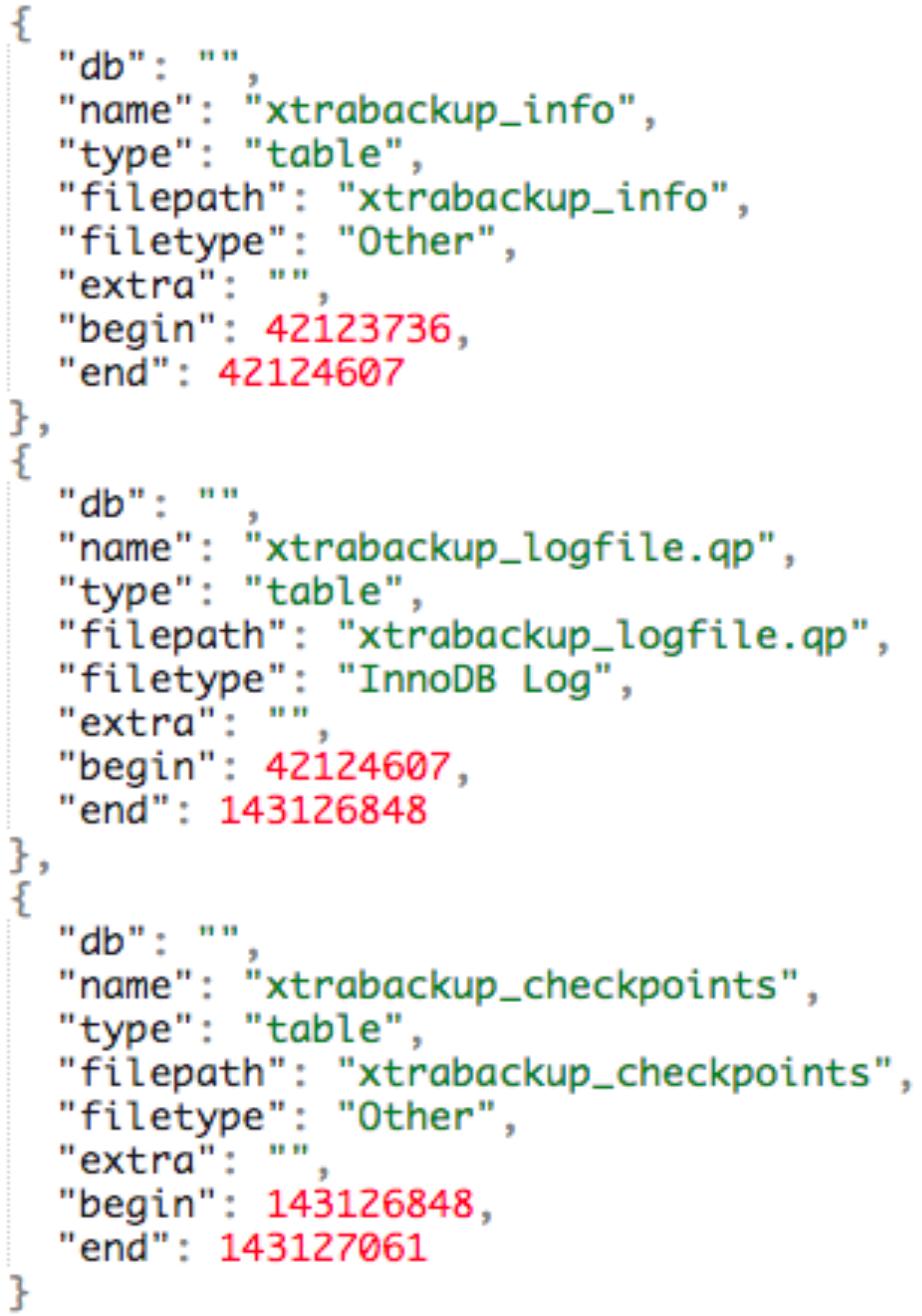


Table Level Recover



```
{
  "type": "db",
  "name": "db2",
  "tables": [
    {
      "db": "db2",
      "name": "t1",
      "type": "table",
      "filepath": "./db2/t1.ibd",
      "filetype": "InnoDB Data",
      "extra": "",
      "begin": 28048,
      "end": 3237390
    }
  ]
},
{
  "type": "db",
  "name": "db1",
  "tables": [
    {
      "db": "db1",
      "name": "t1",
      "type": "table",
      "filepath": "./db1/t1.ibd",
      "filetype": "InnoDB Data",
      "extra": "",
      "begin": 3237390,
      "end": 6457629
    }
  ]
}
```



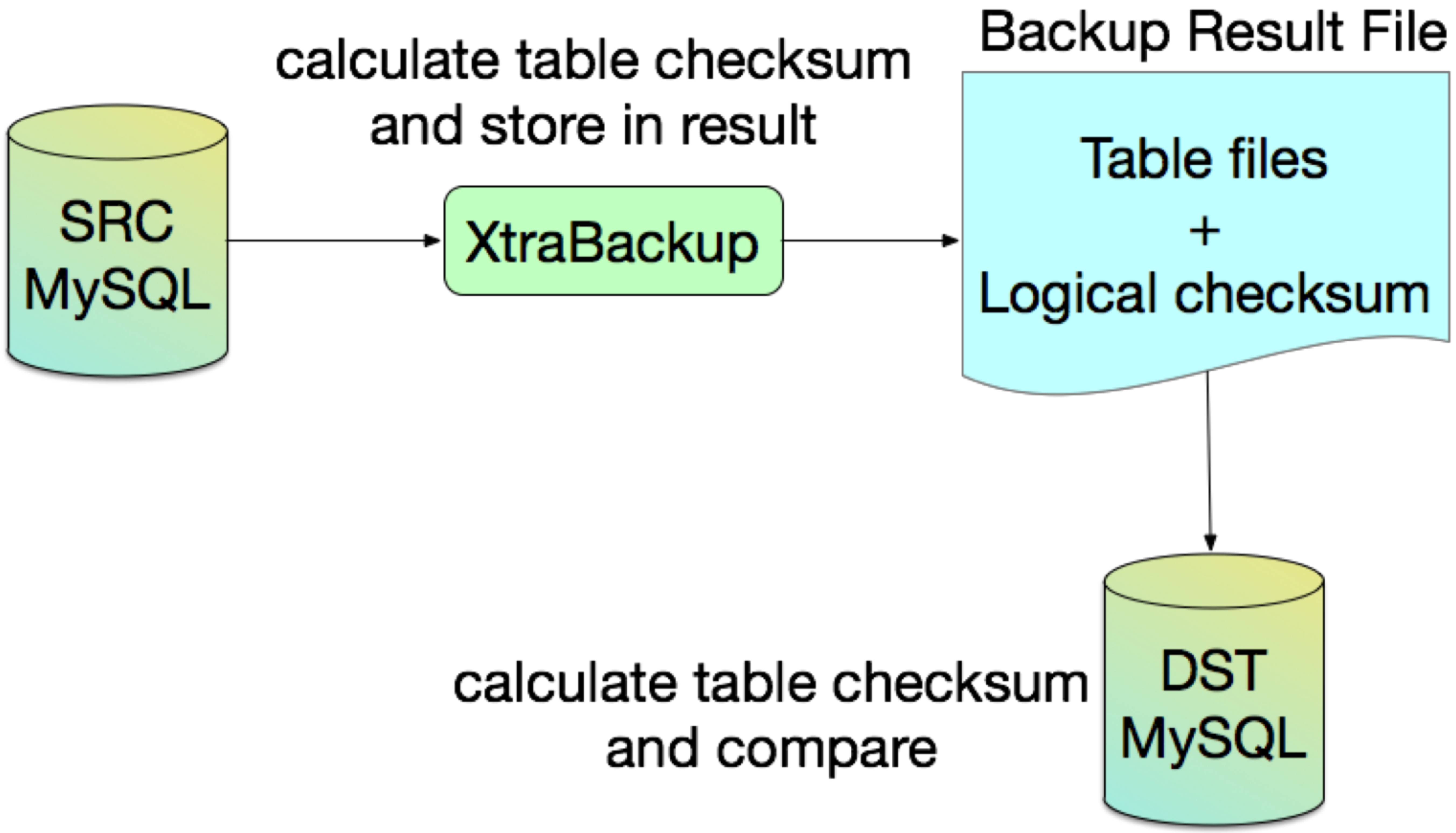
```
{
  "db": "",
  "name": "xtrabackup_info",
  "type": "table",
  "filepath": "xtrabackup_info",
  "filetype": "Other",
  "extra": "",
  "begin": 42123736,
  "end": 42124607
},
{
  "db": "",
  "name": "xtrabackup_logfile.qp",
  "type": "table",
  "filepath": "xtrabackup_logfile.qp",
  "filetype": "InnoDB Log",
  "extra": "",
  "begin": 42124607,
  "end": 143126848
},
{
  "db": "",
  "name": "xtrabackup_checkpoints",
  "type": "table",
  "filepath": "xtrabackup_checkpoints",
  "filetype": "Other",
  "extra": "",
  "begin": 143126848,
  "end": 143127061
}
```

Table Level Recover

- The backup result is still a big OSS file
  - Full backup recovery is not affected at all, just down load the whole file and perform recovery
  - Table level recovery can be achieved by downloading only the files needed, for example to recovery table t1, t1.ibd, t1.frm, ibdata1, xtrabackup_logfile other meta info files need to be downloaded.
- The recovery procedure will be also adjusted, such as to cleanup non-existent table entries in system dictionary (actually PXB already had such logic, we add some tuning).

Backup Result Validation

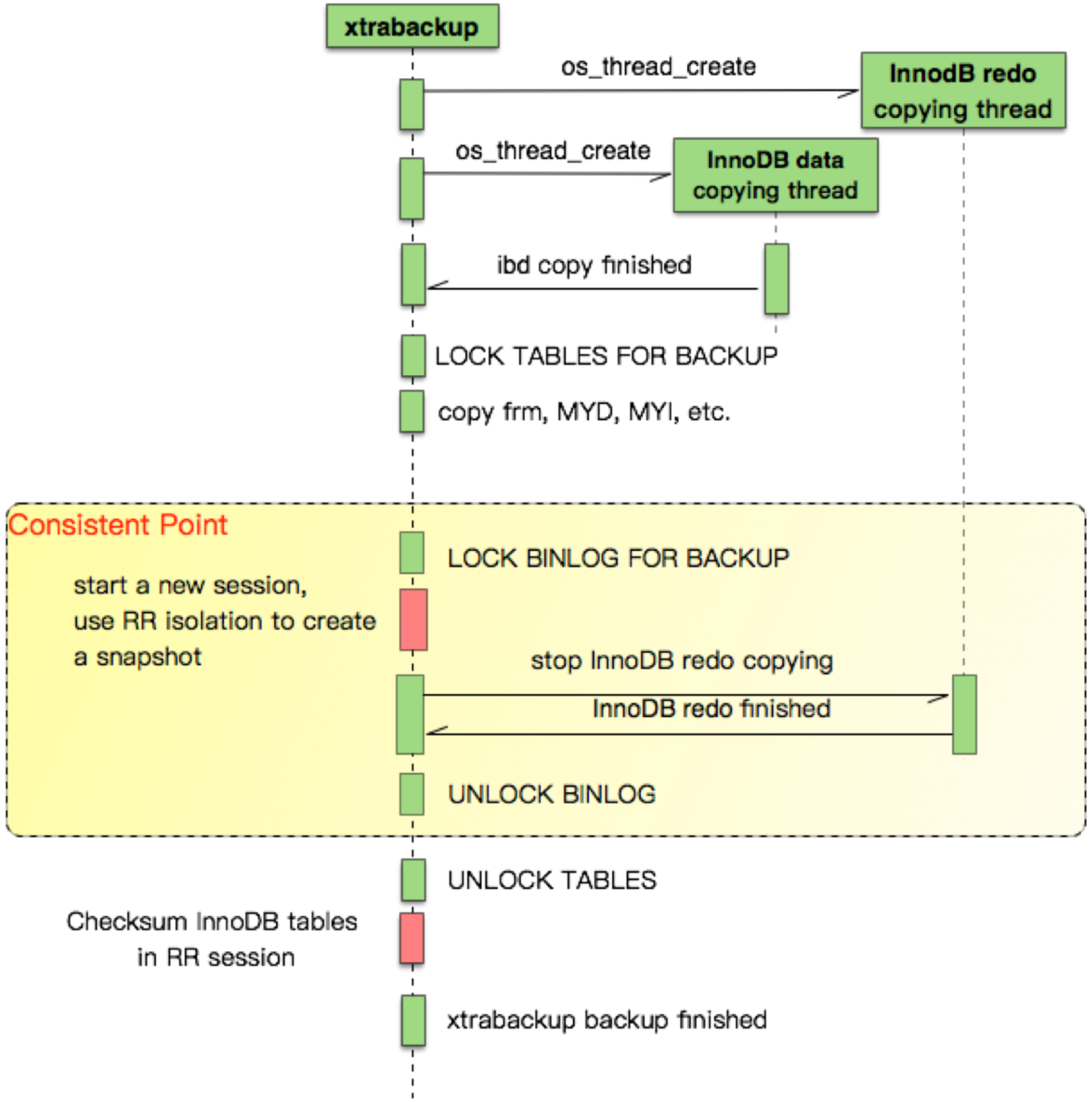
- There is no easy way to validating backup set until it is recovered and restored. We design a validating mechanism from source(backup) to destination(restore).



calculate table checksum and store in result
Backup Result File
SRC MySQL
XtraBackup
Table files + Logical checksum
calculate table checksum and compare
DST MySQL

Backup Result Validation

- The consistent area is the point to which backup result will be recovered.
- Start a RR snapshot use a separate session at consistent point, use this snapshot to calculate table checksum.
- Table checksum result is put into backup set and validated after restore.



Backup Result Validation

- Checksum is a disruptive operation (BP polluted, cold pages need IO)
  - CHECKSUM ENGINE_NO_CACHE TABLE t1;
  - set session rds_sql_max_iops =100;
- Stale records could not be purged if checksum take a very long time for big instance.
  - This validation mechanism is not enabled by default, only used for some selected sentinel instance to validate that our whole backup/restore system is healthy.
- Only validating InnoDB table

Bulk Load Index Creation in 5.7

- Redo log recording is disabled for DDL, and flush tablespace buffer pages to disk before DDL finishing to ensure crashing safe.

  An optimized (without redo logging) DDLoperation has been performed. All modified pages may not have been flushed to the disk yet.

- PXB provides --lock-ddl-per-table to prevent DDL on table

```
BEGIN;
SELECT * FROM t1 LIMIT 1;
backup t1;
SELECT * FROM t2 LIMIT 1;
backup t2;
…

UNLOCK TABLES;
```

Bulk Load Index Creation in 5.7

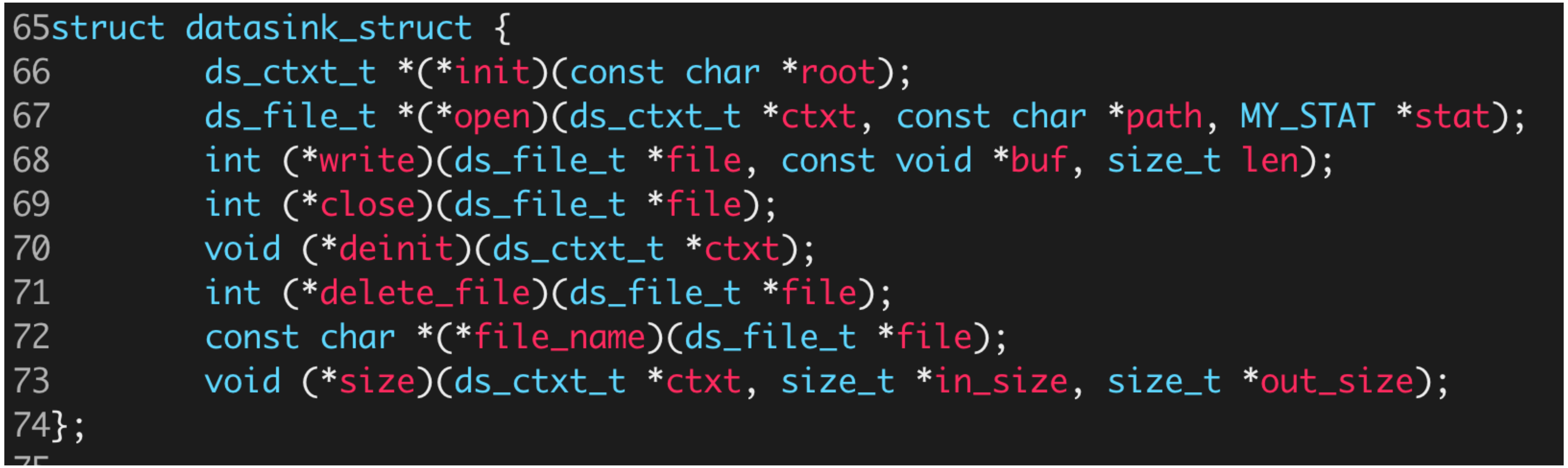
- MariaDB has a feature to switch on redo log recording for DDL, we ported to our AliSQL 5.7 branch, and the RXB will enable redo log recodring for DDL at beginning, and restore at finishing.

```
set global innodb_log_optimize_ddl = off;

Backuing…

set global innodb_log_optimize_ddl = @old_value;
```

Extract Common Datasink Lib

- Datasink is a pretty elegant mechanism/component

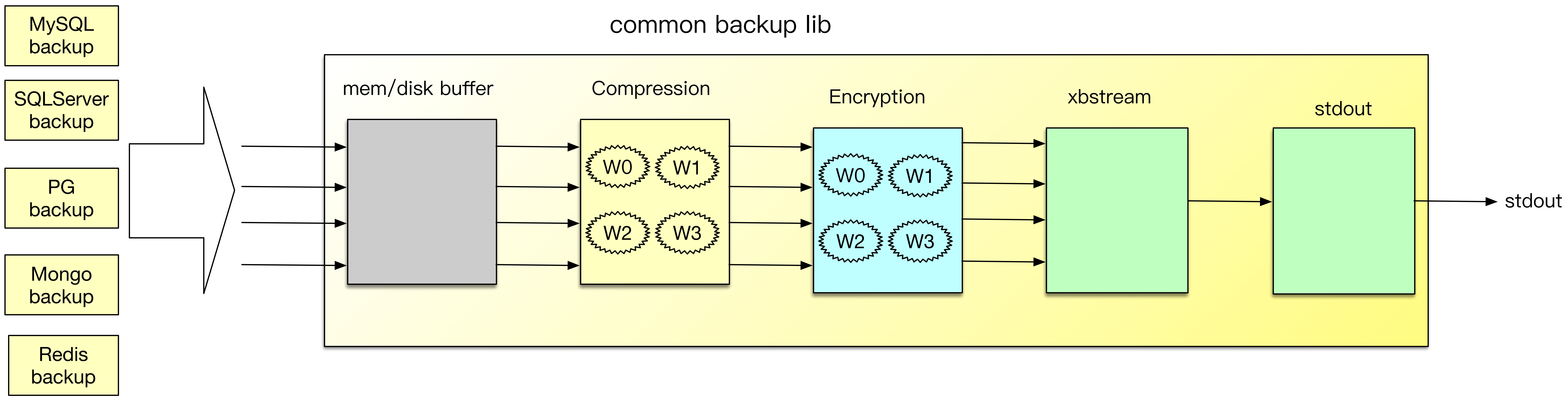
datasink.c
ds_archive.c
ds_blackhole.c
ds_buffer.c
ds_compress.c
ds_decrypt.c
ds_encrypt.c
ds_local.c
ds_stdout.c
ds_tmpfile2.c
ds_tmpfile.c
ds_xbstream.c



```
struct datasink_struct {
        ds_ctxt_t *(*init)(const char *root);
        ds_file_t *(*open)(ds_ctxt_t *ctxt, const char *path, MY_STAT *stat);
        int (*write)(ds_file_t *file, const void *buf, size_t len);
        int (*close)(ds_file_t *file);
        void (*deinit)(ds_ctxt_t *ctxt);
        int (*delete_file)(ds_file_t *file);
        const char *(*file_name)(ds_file_t *file);
        void (*size)(ds_ctxt_t *ctxt, size_t *in_size, size_t *out_size);
};
```

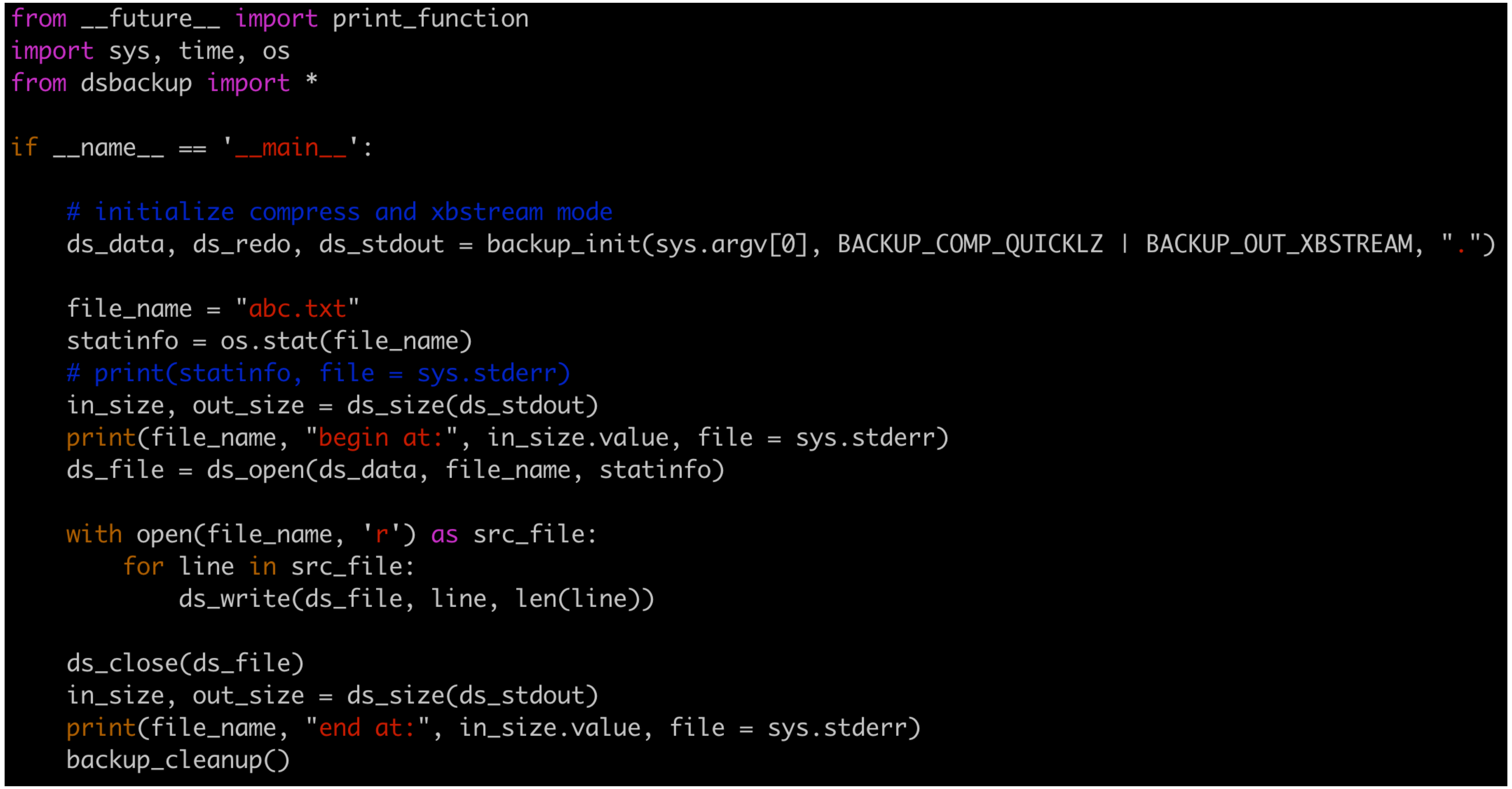
Extract Common Datasink Lib

- Datasinks can be chained together



Extract Common Datasink Lib

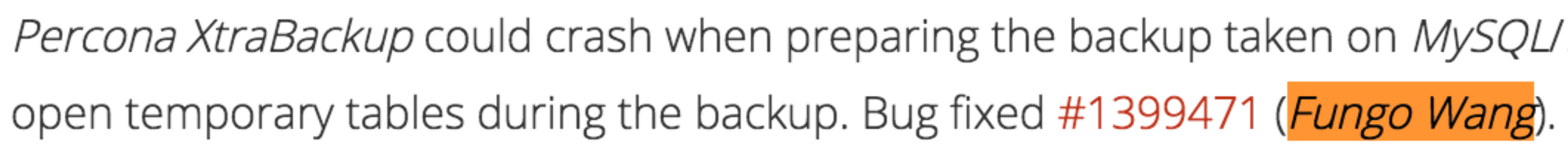
- C lib (RDS for PG)
- Python lib (MongoDB)



```
from __future__ import print_function
import sys, time, os
from dsbackup import *

if __name__ == '__main__':

    # initialize compress and xbstream mode
    ds_data, ds_redo, ds_stdout = backup_init(sys.argv[0], BACKUP_COMP_QUICKLZ | BACKUP_OUT_XBSTREAM, ".")

    file_name = "abc.txt"
    statinfo = os.stat(file_name)
    # print(statinfo, file = sys.stderr)
    in_size, out_size = ds_size(ds_stdout)
    print(file_name, "begin at:", in_size.value, file = sys.stderr)
    ds_file = ds_open(ds_data, file_name, statinfo)

    with open(file_name, 'r') as src_file:
        for line in src_file:
            ds_write(ds_file, line, len(line))

    ds_close(ds_file)
    in_size, out_size = ds_size(ds_stdout)
    print(file_name, "end at:", in_size.value, file = sys.stderr)
    backup_cleanup()
```

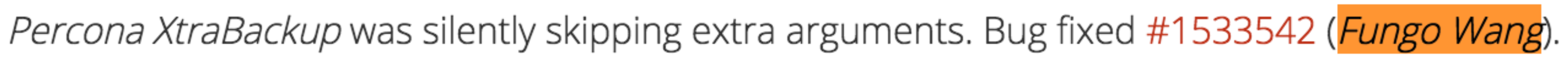
Contribute to PXB Upstream

- Report bugs, and submit PR to PXB



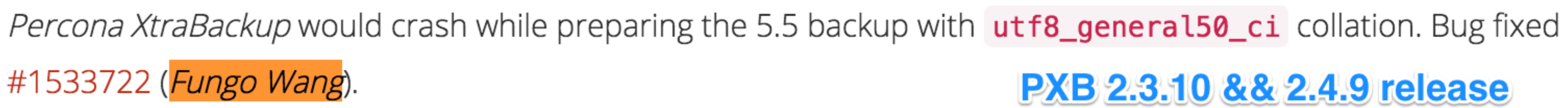
Percona XtraBackup could crash when preparing the backup taken on *MySQL*/*Percona Server* 5.5 if there were open temporary tables during the backup. Bug fixed #1399471 (*Fungo Wang*).

PXB 2.2.11 release



Percona XtraBackup was silently skipping extra arguments. Bug fixed #1533542 (*Fungo Wang*).

PXB 2.3.4 && 2.4.1 release



Percona XtraBackup would crash while preparing the 5.5 backup with `utf8_general50_ci` collation. Bug fixed #1533722 (*Fungo Wang*).

PXB 2.3.10 && 2.4.9 release

- TokuDB backup feature is opensourced

https://github.com/alibaba/AliSQLBackup

Alibaba Cloud | MORE THAN JUST CLOUD