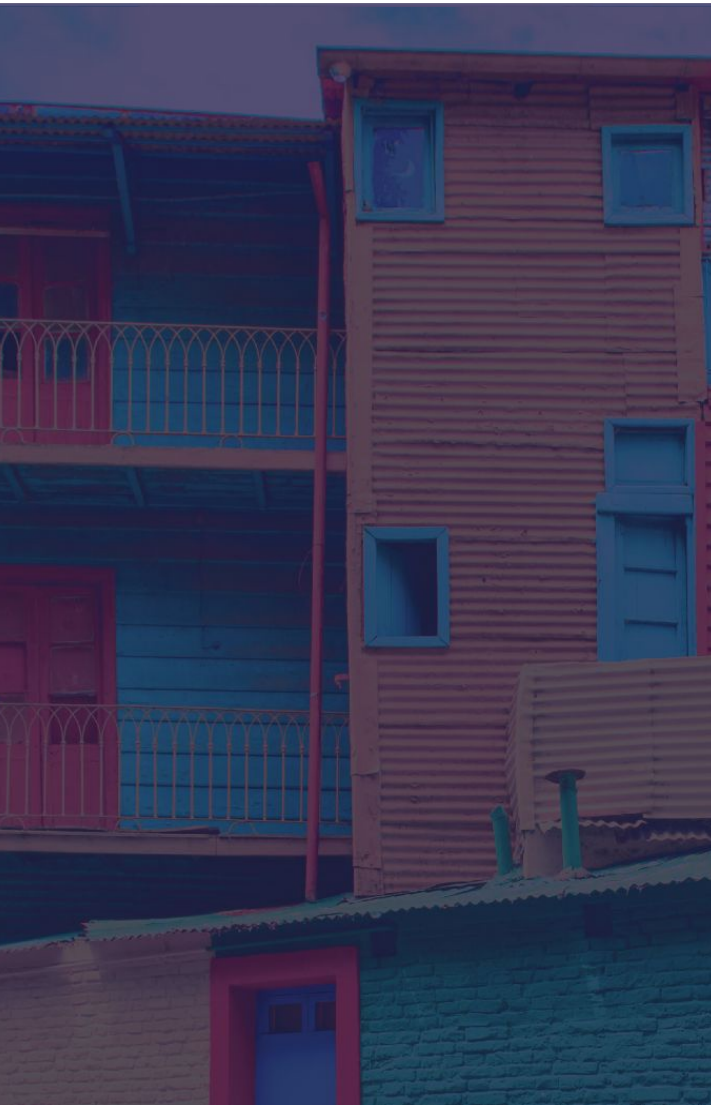
MyDumper Internals

David Ducos - Senior Architect Percona

Buenos Aires, Argentina 2023

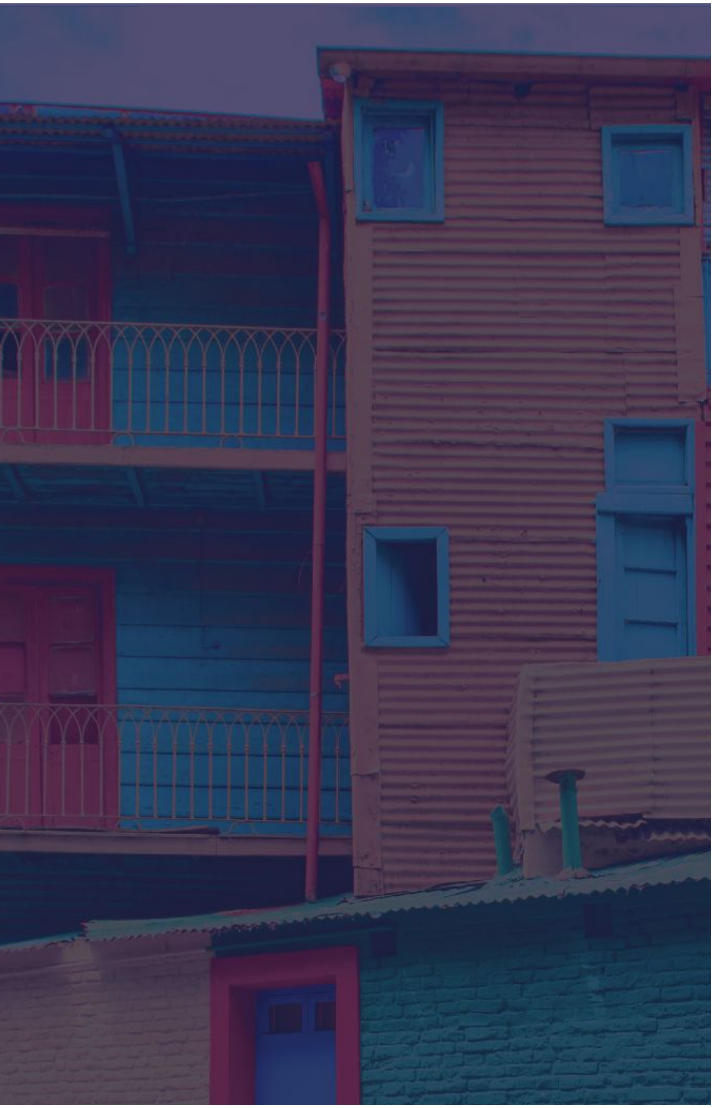
PERCONA UNIVERSITY

PERCONA



MyDumper ha cambiado mucho en los últimos 3 years.

Vamos a revisar en profundidad cambios más importantes, las motivos de estos cambios y que es lo que nos espera en el futuro.



Agenda

- Introduccion
- mydumper
- myloader
- Performance y casos de usos
- Preguntas

MyDumper Internals - Introduccion

Que es MyDumper?

MyDumper es una Herramienta Multithreading para tomar Backups Lógicos.

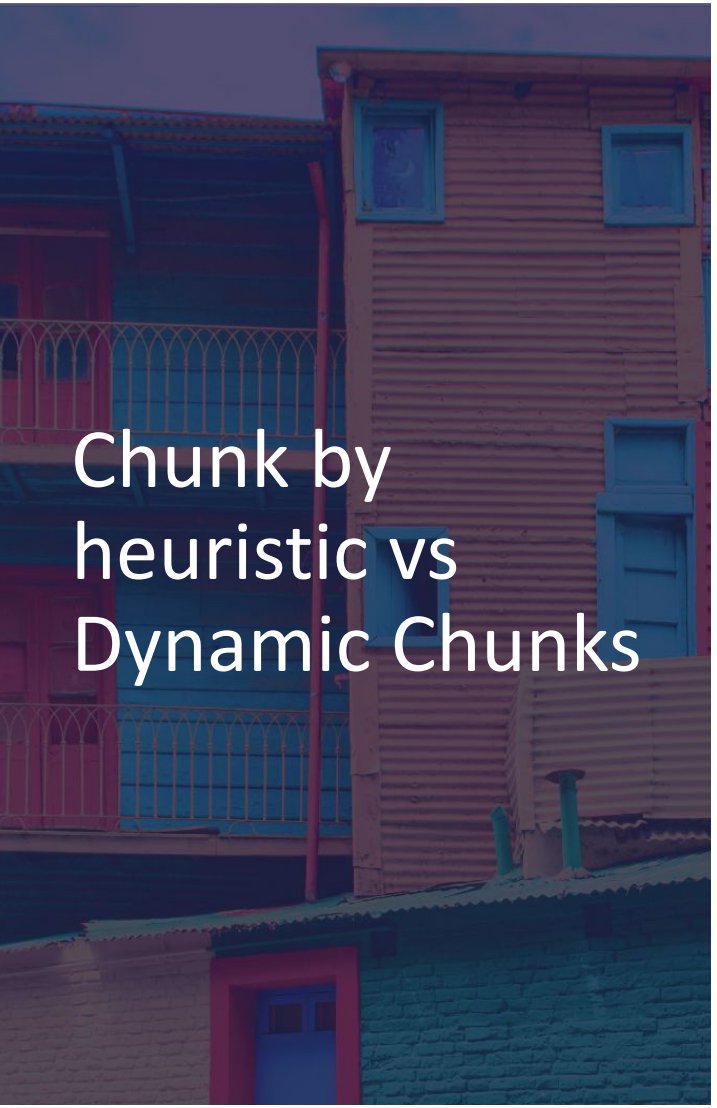
MyDumper es Open Source y es mantenida por la comunidad, no pertenece a Percona, MariaDB o MySQL.

MyDumper? mydumper? myloader?

PERCONA

mydumper Internals

Antes	Ahora
Chunk by heuristic	Dynamic chunks
Text and GZIP write	External command / PIPE
-r or -F	-r and -F (2 levels)



Chunk by heuristic vs Dynamic Chunks

Tamaños fijos de chunks

Armado del job

Espera del worker

Chunks variables

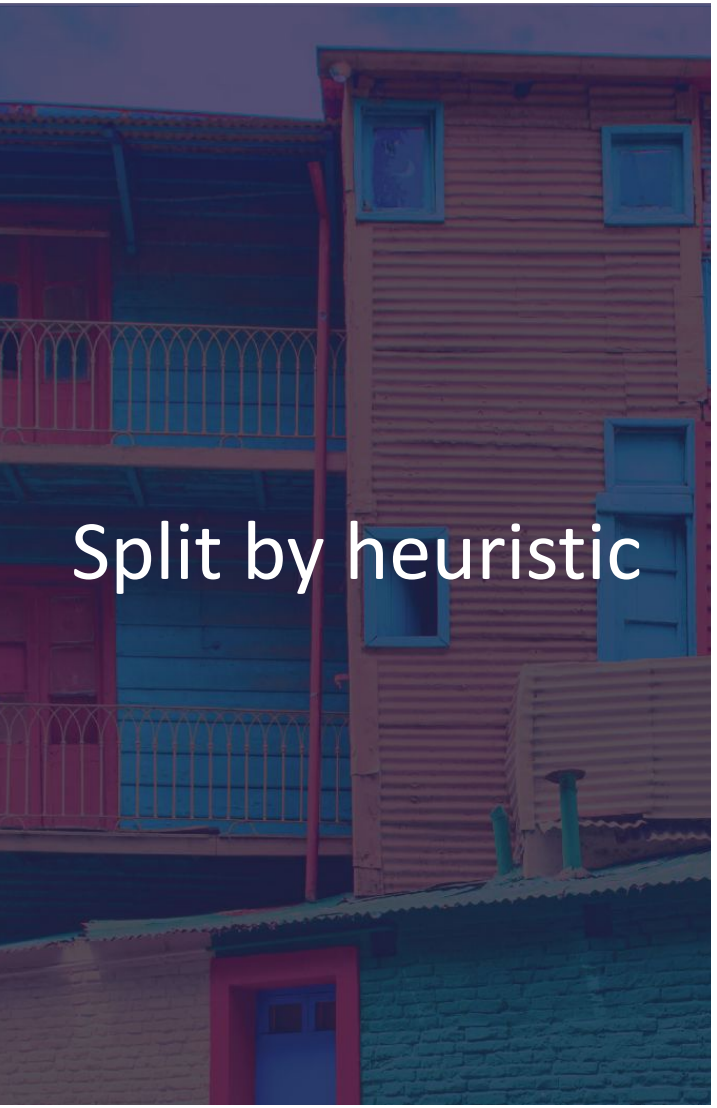
dependiendo del query time

Armado de la query al

momento de procesar

Sincronización del split de

jobs + steps



Split by heuristic

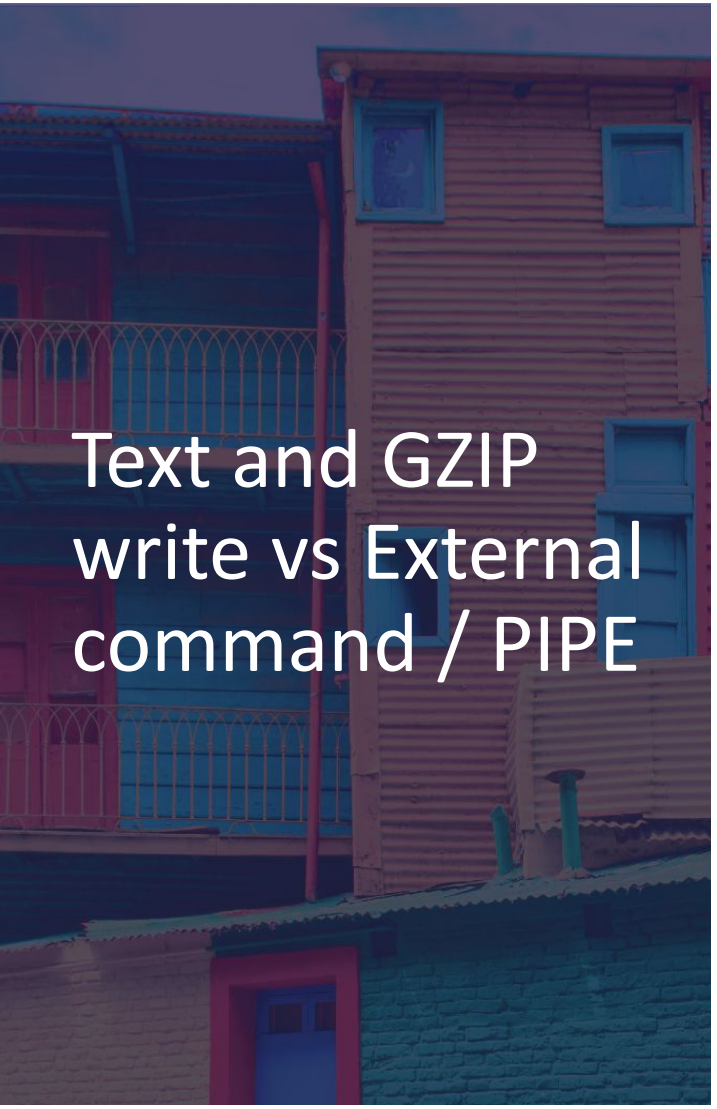
```
CREATE TABLE `t1` (
  `id` bigint unsigned NOT NULL AUTO_INCREMENT,
  `val` int DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB

mysql> insert into t1 (val) select val from t1;
Query OK, 524288 rows affected (5.50 sec)
Records: 524288  Duplicates: 0  Warnings: 0

mysql> select min(id), max(id) from t1;
+---------+---------+
| min(id) | max(id) |
+---------+---------+
|       1 | 1310693 |
+---------+---------+
1 row in set (0.01 sec)

mysql> insert into t1 (id,val) values (pow(2,63), 2);
Query OK, 1 row affected (0.00 sec)

mysql> select min(id), max(id) from t1;
+---------+---------------------+
| min(id) | max(id)             |
+---------+---------------------+
|       1 | 9223372036854775808 |
+---------+---------------------+
1 row in set (0.00 sec)
```

se generaban millones de jobs causando OOM

Dynamic split

```
job1: [1                    ,9223372036854775808] (Step: 1k)
Split 1
job1: [1                    ,4611686018427387904] (Step: 1k)
job2: [4611686018427387904,9223372036854775808] (Step: 1k increasing upto max limit)
Split 2
job1: [1                    ,2305843009213693952] (Step: 1k)
job3: [2305843009213693952,4611686018427387904] (Step: 1k increasing upto max limit)
~~job2: [4611686018427387904,9223372036854775808] (Step: MaxStep)~~
```


PERCONA

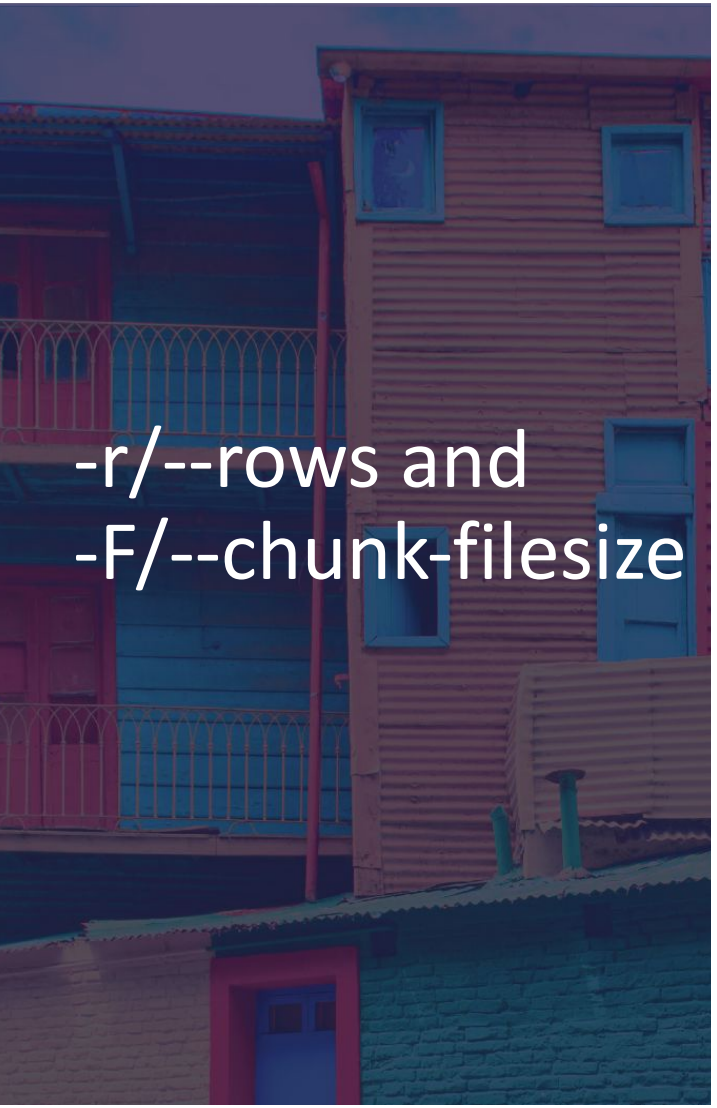


Text and GZIP write vs External command / PIPE

Solo 2 opciones: Text and GZIP

Solamente comando externo

Que paso con ZSTD?



-r/--rows and -F/--chunk-filesize

Solo se podía hacer split de la tabla o por tamaño del archivo

Fue necesario 2 niveles

-r/--rows and -F/--chunk-filesize

Formato:

SCHEMA_NAME.TABLE_NAME.{BINARY ORDER}.{SECUENCIAL ORDER}.sql

Full Table scan / llega al limite de -F en el job:

SCHEMA_NAME.TABLE_NAME.00000.sql

SCHEMA_NAME.TABLE_NAME.00000.00001.sql

SCHEMA_NAME.TABLE_NAME.00000.00002.sql

PERCONA

-r/--rows and -F/--chunk-filesize

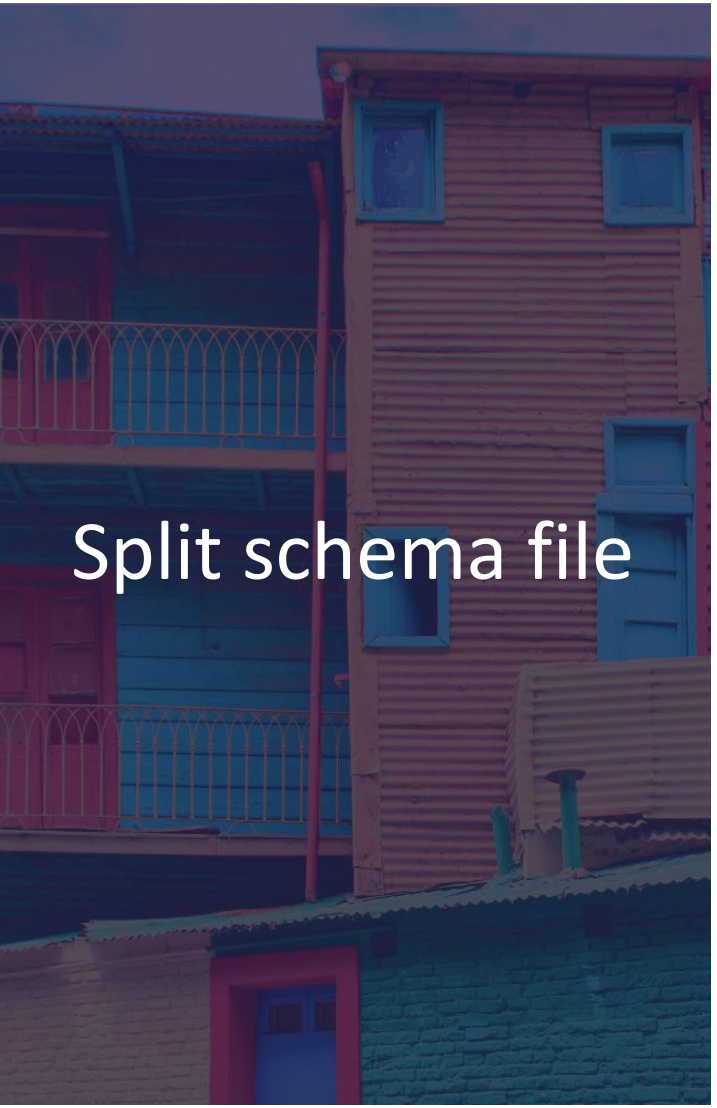
```
job1: [    1, 1000000] [SCHEMA_NAME.TABLE_NAME.00000.sql] 000 / Deep 0

Split 1
job0: [    1,  500000] [SCHEMA_NAME.TABLE_NAME.00000.sql] 000 / Deep 1
job1: [500001,1000000] [SCHEMA_NAME.TABLE_NAME.00001.sql] 001 / Deep 1

Split 2
job1: [500001,1000000] [SCHEMA_NAME.TABLE_NAME.00001.sql] 001 / Deep 1
job0: [     1, 250000] [SCHEMA_NAME.TABLE_NAME.00000.sql] 000 / Deep 2
job2: [250001, 500000] [SCHEMA_NAME.TABLE_NAME.00002.sql] 010 / Deep 2

Split 3
job0: [     1, 250000] [SCHEMA_NAME.TABLE_NAME.00000.sql] 000 / Deep 2
job2: [250001, 500000] [SCHEMA_NAME.TABLE_NAME.00002.sql] 010 / Deep 2
job1: [500001, 750000] [SCHEMA_NAME.TABLE_NAME.00001.sql] 001 / Deep 2
job3: [750001,1000000] [SCHEMA_NAME.TABLE_NAME.00003.sql] 011 / Deep 2

Split 4
job2: [250001, 500000] [SCHEMA_NAME.TABLE_NAME.00002.sql] 010 / Deep 2
job1: [500001, 750000] [SCHEMA_NAME.TABLE_NAME.00001.sql] 001 / Deep 2
job3: [750001,1000000] [SCHEMA_NAME.TABLE_NAME.00003.sql] 011 / Deep 2
job0: [     1, 125000] [SCHEMA_NAME.TABLE_NAME.00000.sql] 000 / Deep 3
job4: [125001, 250000] [SCHEMA_NAME.TABLE_NAME.00004.sql] 100 / Deep 3

Orden: 0, 4, 2, 1, 3
```


PERCONA

- **Mezclar diferente niveles de split / Partitioning / VAR/CHAR**
- **Otro/s nivel/es de pipe**
- **Automatizar --rows**

myloader Internals

Antes	Ahora
File content agnostic	Split schema file
Only restore backups from disk	Disk and Stream same code
Metadata	Metadata per table

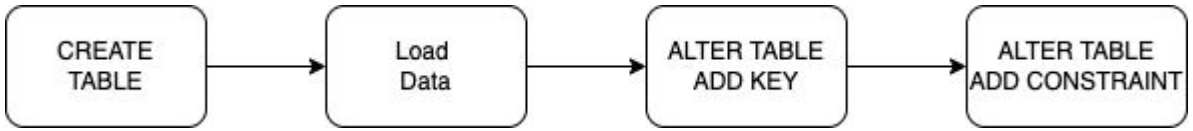


Dividimos el CREATE TABLE en:

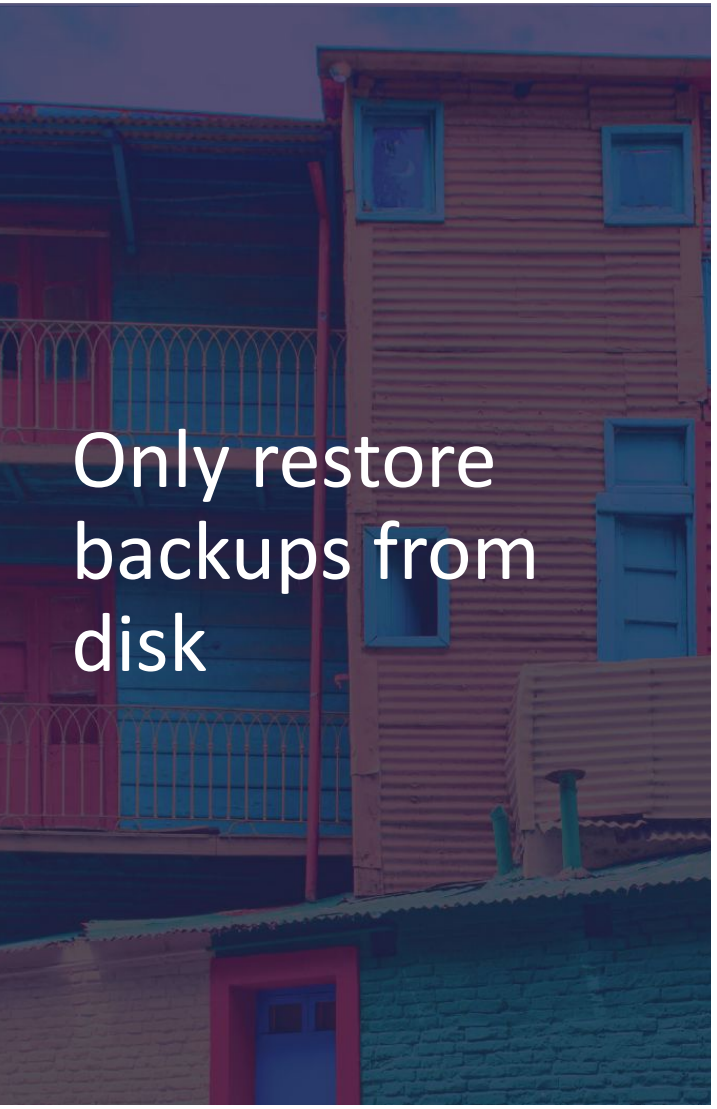
- CREATE TABLE con solo las columnas + PK
- Los índices secundarios
- Foreign Keys

```
CREATE TABLE `Products` (
  `id` int NOT NULL AUTO_INCREMENT,
  `productName` varchar(255) DEFAULT NULL,
  `productDescription` varchar(2048) DEFAULT NULL,
  `productCode` varchar(255) DEFAULT NULL,
  `productBarCode` varchar(255) DEFAULT NULL,
  `productStatus` char(1) DEFAULT NULL,
  `ManufacturerId` int DEFAULT NULL,
  `ProductCategoryId` int DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `ManufacturerId` (`ManufacturerId`),
  KEY `ProductCategoryId` (`ProductCategoryId`),
  CONSTRAINT `Products_ibfk` FOREIGN KEY (`ManufacturerId`)
REFERENCES `Manufacturers` (`id`),
  CONSTRAINT `Products_ibfk` FOREIGN KEY (`ProductCategoryId`)
REFERENCES `ProductCategories` (`id`));
```

```
CREATE TABLE `Products` (
  `id` int NOT NULL AUTO_INCREMENT,
  `productName` varchar(255) DEFAULT NULL,
  `productDescription` varchar(2048) DEFAULT NULL,
  `productCode` varchar(255) DEFAULT NULL,
  `productBarCode` varchar(255) DEFAULT NULL,
  `productStatus` char(1) DEFAULT NULL,
  `ManufacturerId` int DEFAULT NULL,
  `ProductCategoryId` int DEFAULT NULL,
  PRIMARY KEY (`id`));

ALTER TABLE `Products`
ADD KEY `ManufacturerId` (`ManufacturerId`),
ADD KEY `ProductCategoryId` (`ProductCategoryId`);

ALTER TABLE `Products`
ADD CONSTRAINT `Products_ibfk_2` FOREIGN KEY (`ManufacturerId`)
REFERENCES `Manufacturers` (`id`) ON DELETE SET NULL ON UPDATE
CASCADE,
ADD CONSTRAINT `Products_ibfk_3` FOREIGN KEY (`ProductCategoryId`)
REFERENCES `ProductCategories` (`id`) ON DELETE SET NULL ON UPDATE
CASCADE);
```



PERCONA



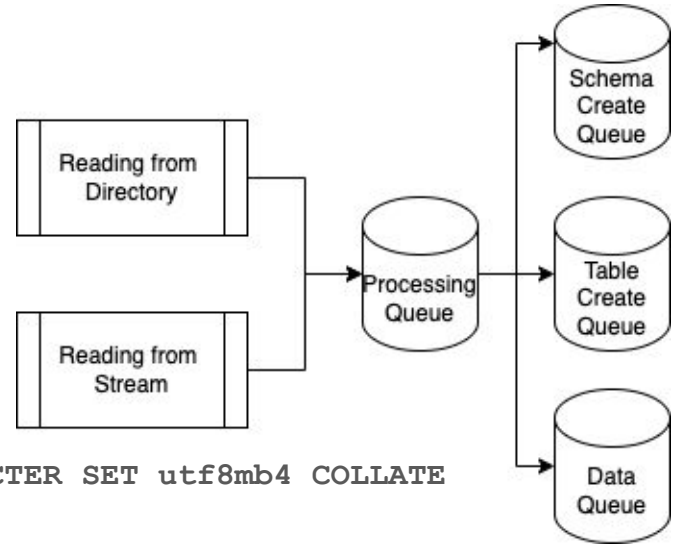
El directorio se leía en 3 pasadas

No comenzaba hasta procesar todos los archivos

Disk and Stream same code

Una sola pasada

Procesar como si fuera un stream




```
-- sakila-schema-create.sql 155
CREATE DATABASE /*!32312 IF NOT EXISTS*/ `sakila` /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE
utf8mb4_0900_ai_ci */ /*!80016 DEFAULT ENCRYPTION='N' */;

-- sakila.actor-schema.sql 501
/*!40101 SET NAMES binary*/;
/*!40014 SET FOREIGN_KEY_CHECKS=0*/;
/*!40103 SET TIME_ZONE='+00:00' */;
CREATE TABLE `actor` (
  `actor_id` smallint unsigned NOT NULL AUTO_INCREMENT,
  `first_name` varchar(45) NOT NULL,
  `last_name` varchar(45) NOT NULL,
  `last_update` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`actor_id`),
  KEY `idx_actor_last_name` (`last_name`)
) ENGINE=InnoDB AUTO_INCREMENT=201 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Old metadata file

```
Started dump at: 2023-10-18 16:01:31
SHOW MASTER STATUS:
    Log: binlog.000007
    Pos: 74180387
    GTID:

SHOW SLAVE STATUS:
    Host: 127.0.0.1
    Log: ubuntu-jammy-bin.000002
    Pos: 937
    GTID:

Finished dump at: 2023-10-18 16:01:31
```


PERCONA

New metadata file

```
# Started dump at: 2023-10-18 15:39:14
[master]
# Channel_Name = '' # It can be use to setup replication FOR CHANNEL
File = binlog.000007
Position = 71516407
Executed_Gtid_Set =


[replication]
# relay_master_log_file = 'ubuntu-jammy-bin.000002'
# exec_master_log_pos = 937
# Executed_Gtid_Set =
# Slave_IO_State = 'Waiting for source to send event'
# Master_Host = '127.0.0.1'
# Master_User = 'root'
# Master_Port = 3357
# Connect_Retry = 60
# Master_Log_File = 'ubuntu-jammy-bin.000002'
# Read_Master_Log_Pos = 937
# Relay_Log_File = 'ubuntu-jammy-relay-bin.000003'

-- -- --

# Network_Namespace = ''
# myloader_exec_reset_slave = 0 # 1 means execute the command
# myloader_exec_change_master = 0 # 1 means execute the command
# myloader_exec_start_slave = 0 # 1 means execute the command
```

```
[`sakila`.`actor`]
real_table_name=actor
rows = 200
data_checksum = 60988714
schema_checksum = C75A129E
indexes_checksum = 7E1DB0BA

[`sakila`.`mydumper_0`]
real_table_name=actor.info
rows = 0
schema_checksum = 790AEB99

-- -- --

[`sakila`.`store`]
real_table_name=store
rows = 2
data_checksum = 3119812626
schema_checksum = B7B99B4C
indexes_checksum = B4D31E3

[`sakila`]
schema_checksum = 95DC8DDE
post_checksum = 42085F07
triggers_checksum = A4255BB4
# Finished dump at: 2023-10-18 15:39:14
```


PERCONA

- **Reutilización de las conexiones**
- **Partir INSERT en multiples threads**

Comparación con versiones anteriores

Comparación con otras Multithreading Logical Backup Tools

Comparación con versiones anteriores

- **Más threads**
- **Más estructuras de datos**
- **Mayor complejidad**

Comparación con otras Multithreading Logical Backup Tools

- MySQL Shell
  - dumpTables
  - dumpSchemas
  - dumpInstance

```
Sysbench 5 Tables 1M rows

# rm -rf data; ./mydumper -r 250000 -o data -c ZSTD -L log -v 3 -B sbtest;
head -1 log; tail -1 log
2023-10-19 14:20:30 [INFO] - MyDumper backup version: 0.15.2-4
2023-10-19 14:20:56 [INFO] - Finished dump at: 2023-10-19 14:20:56

# rm -rf data; ./mydumper -r 250000 -o data -c ZSTD -L log -v 3 -B sbtest
--load-data; head -1 log; tail -1 log
2023-10-19 14:24:10 [INFO] - MyDumper backup version: 0.15.2-4
2023-10-19 14:24:39 [INFO] - Finished dump at: 2023-10-19 14:24:39

mysql-py []> util.dump_schemas(["sbtest"], "/tmp/bkp", {"compression":"zstd",
"threads":4, "chunking":True, "showProgress": True})
Duration: 00:00:41s
```

Use cases

Masquerade

```
[`schema_name`.`table_name`]
`phone`=random_format '+1 ('<number 3>') '<number 3>'-'<number 4>
`emails`=random_format <file names.txt>'.'<file surnames.txt>'@'<file domains.txt>
`addresses`=random_format <number 3>' '<file streets.txt>', '<file cities.txt>', '<file
states_and_zip.txt>', USA'
```

--exec-per-thread

```
# rm -rf data; ./mydumper -r 1000:10000:0 -o data -T sbtest.sbtest1
--exec-per-thread="/usr/bin/xz -z" --exec-per-thread-extension=".xz"
2023-10-19 18:45:48 [INFO] - MyDumper backup version: 0.15.2-4
2023-10-19 18:49:17 [INFO] - Finished dump at: 2023-10-19 18:49:17
```

PERCONA