THE OPTIMIZATION OF FTS SQL QUERIES

AUTHOR:

Muhammad Zafar Iqbal, Nazir
IT-SD-PDS

SUPERVISOR(S):

Steven Murray
Luca Mascetti

AUGUST 2023

What is FTS?

File Transfer Service

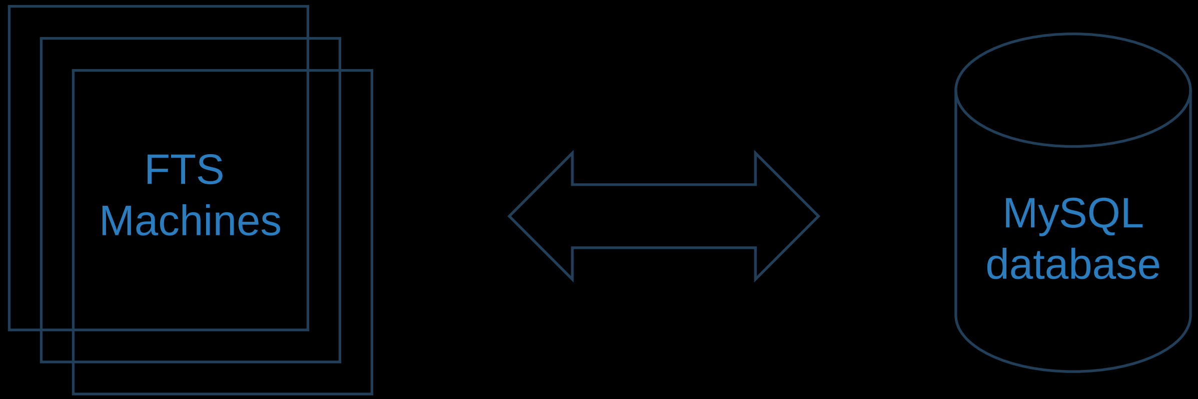
- FTS is a bulk data mover, created to distribute multiple Petabytes of data from the LHC at CERN to storage end points located all around the World.

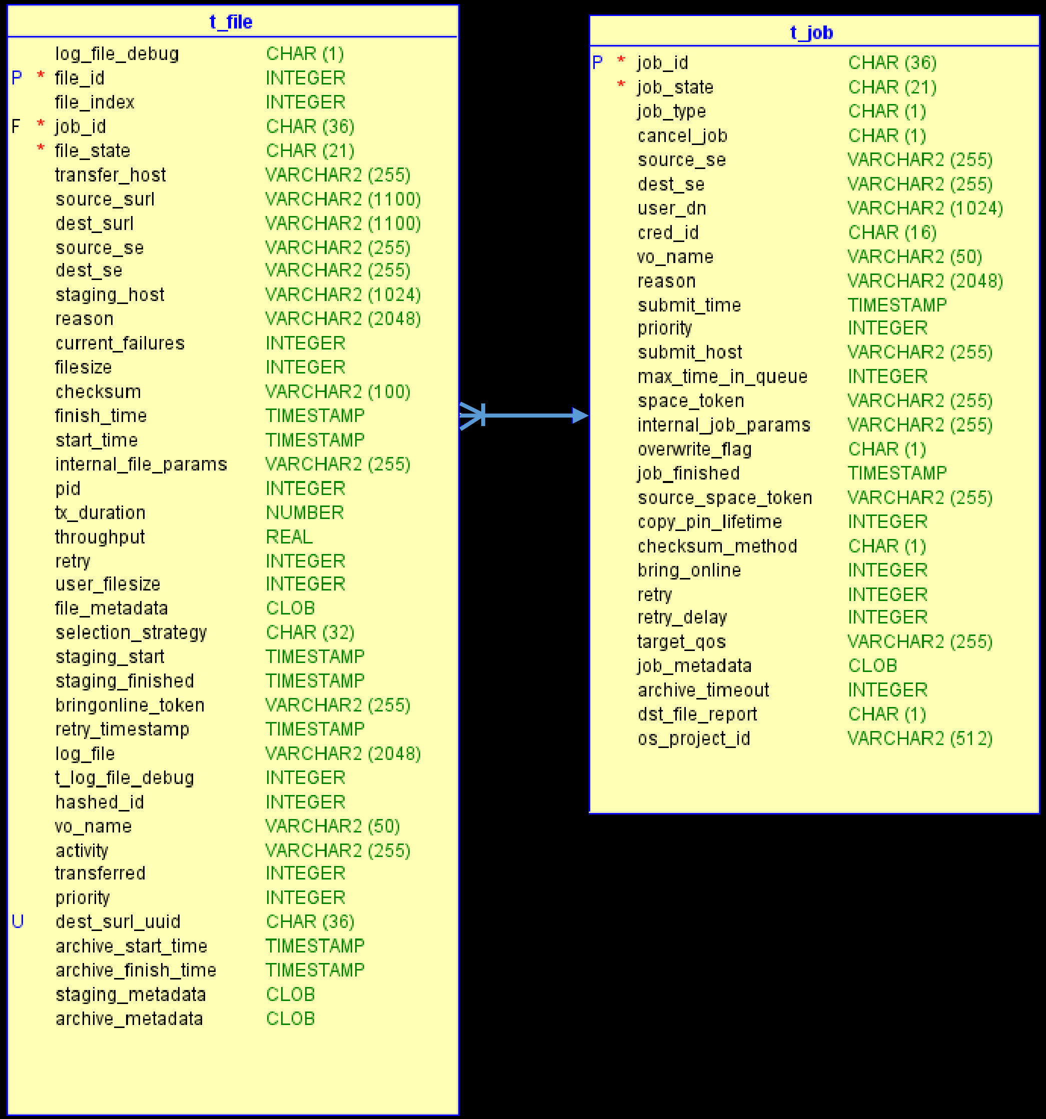
- FTS queues and schedules data transfers, maximising the use of available network and storage resources whilst ensuring policy limits are respected.

- FTS provides a Web interface to monitor and debug the file transfers that it schedules and executes.

FTS Architecture

A cluster of one or more identical machines sharing a single MySQL database. Each machine hosts a web interface using Apache and additional daemons to schedule and execute the file transfers.





t_file

		log_file_debug	CHAR (1)
P	*	file_id	INTEGER
		file_index	INTEGER
F	*	job_id	CHAR (36)
	*	file_state	CHAR (21)
		transfer_host	VARCHAR2 (255)
		source_surl	VARCHAR2 (1100)
		dest_surl	VARCHAR2 (1100)
		source_se	VARCHAR2 (255)
		dest_se	VARCHAR2 (255)
		staging_host	VARCHAR2 (1024)
		reason	VARCHAR2 (2048)
		current_failures	INTEGER
		filesize	INTEGER
		checksum	VARCHAR2 (100)
		finish_time	TIMESTAMP
		start_time	TIMESTAMP
		internal_file_params	VARCHAR2 (255)
		pid	INTEGER
		tx_duration	NUMBER
		throughput	REAL
		retry	INTEGER
		user_filesize	INTEGER
		file_metadata	CLOB
		selection_strategy	CHAR (32)
		staging_start	TIMESTAMP
		staging_finished	TIMESTAMP
		bringonline_token	VARCHAR2 (255)
		retry_timestamp	TIMESTAMP
		log_file	VARCHAR2 (2048)
		t_log_file_debug	INTEGER
		hashed_id	INTEGER
		vo_name	VARCHAR2 (50)
		activity	VARCHAR2 (255)
		transferred	INTEGER
		priority	INTEGER
U		dest_surl_uuid	CHAR (36)
		archive_start_time	TIMESTAMP
		archive_finish_time	TIMESTAMP
		staging_metadata	CLOB
		archive_metadata	CLOB

t_job

P	*	job_id	CHAR (36)
	*	job_state	CHAR (21)
		job_type	CHAR (1)
		cancel_job	CHAR (1)
		source_se	VARCHAR2 (255)
		dest_se	VARCHAR2 (255)
		user_dn	VARCHAR2 (1024)
		cred_id	CHAR (16)
		vo_name	VARCHAR2 (50)
		reason	VARCHAR2 (2048)
		submit_time	TIMESTAMP
		priority	INTEGER
		submit_host	VARCHAR2 (255)
		max_time_in_queue	INTEGER
		space_token	VARCHAR2 (255)
		internal_job_params	VARCHAR2 (255)
		overwrite_flag	CHAR (1)
		job_finished	TIMESTAMP
		source_space_token	VARCHAR2 (255)
		copy_pin_lifetime	INTEGER
		checksum_method	CHAR (1)
		bring_online	INTEGER
		retry	INTEGER
		retry_delay	INTEGER
		target_qos	VARCHAR2 (255)
		job_metadata	CLOB
		archive_timeout	INTEGER
		dst_file_report	CHAR (1)
		os_project_id	VARCHAR2 (512)

The Problem

Reduce the latency of FTS SQL queries and the RAM they consume

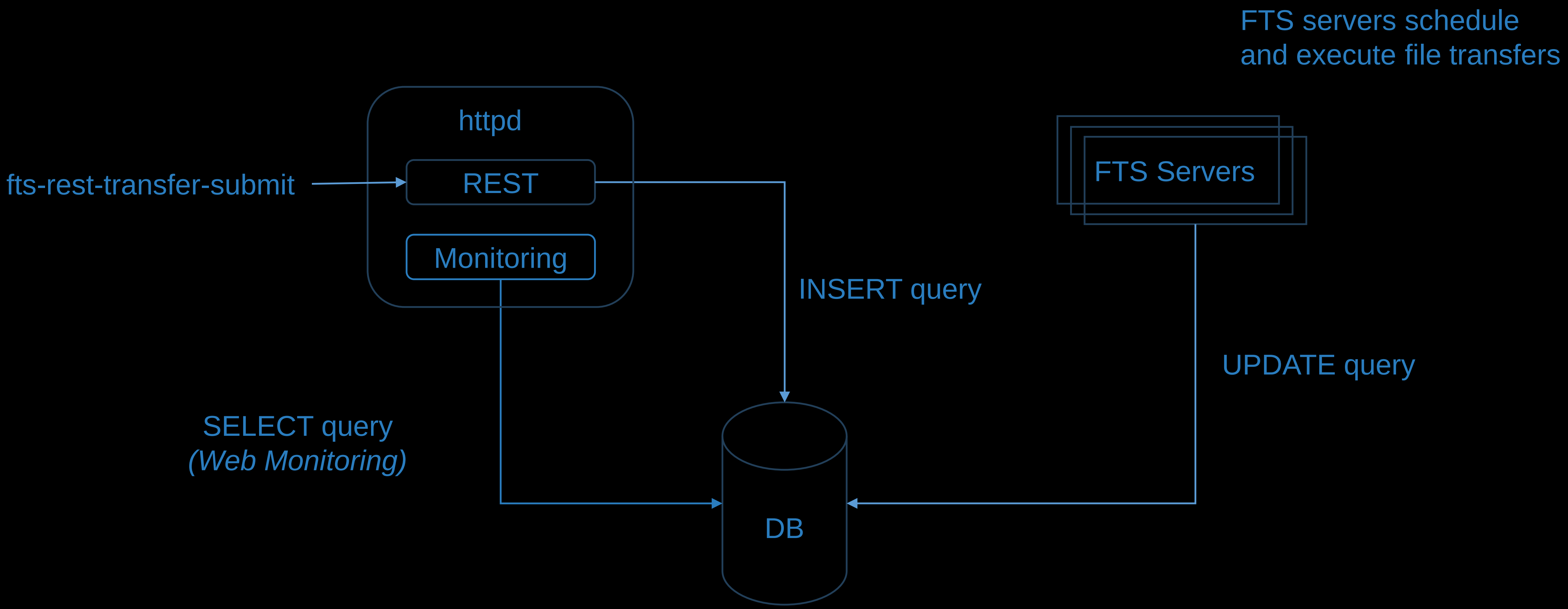
- The majority of the FTS queuing and scheduling logic takes place within the MySQL database.
- FTS must be able to run on top of a cloud-based MySQL database that prioritizes being able running 1000s of average performance databases over running a few high-performance databases.
- Startup constraints such as the maximum time allowed to restart the FTS service after a power cut must be addressed as well as the usual performances of queries executed by a running system.
- A systematic method should be found to help determine the most costly FTS SQL queries.

Systematic slow query detection

- MySQL provides the slow query log
- To switch the log on:
  - SET GLOBAL slow_query_log = 'ON’;
- We can dig down to the possible cause of the problem using:
  - EXPLAIN ANALYZE
- A query can be considered slow based on time and on number of rows accessed.
- To detect slow queries by number of rows accessed:
  - SET GLOBAL min_examined_row_limit = 10000;

File Transfer Lifecycle

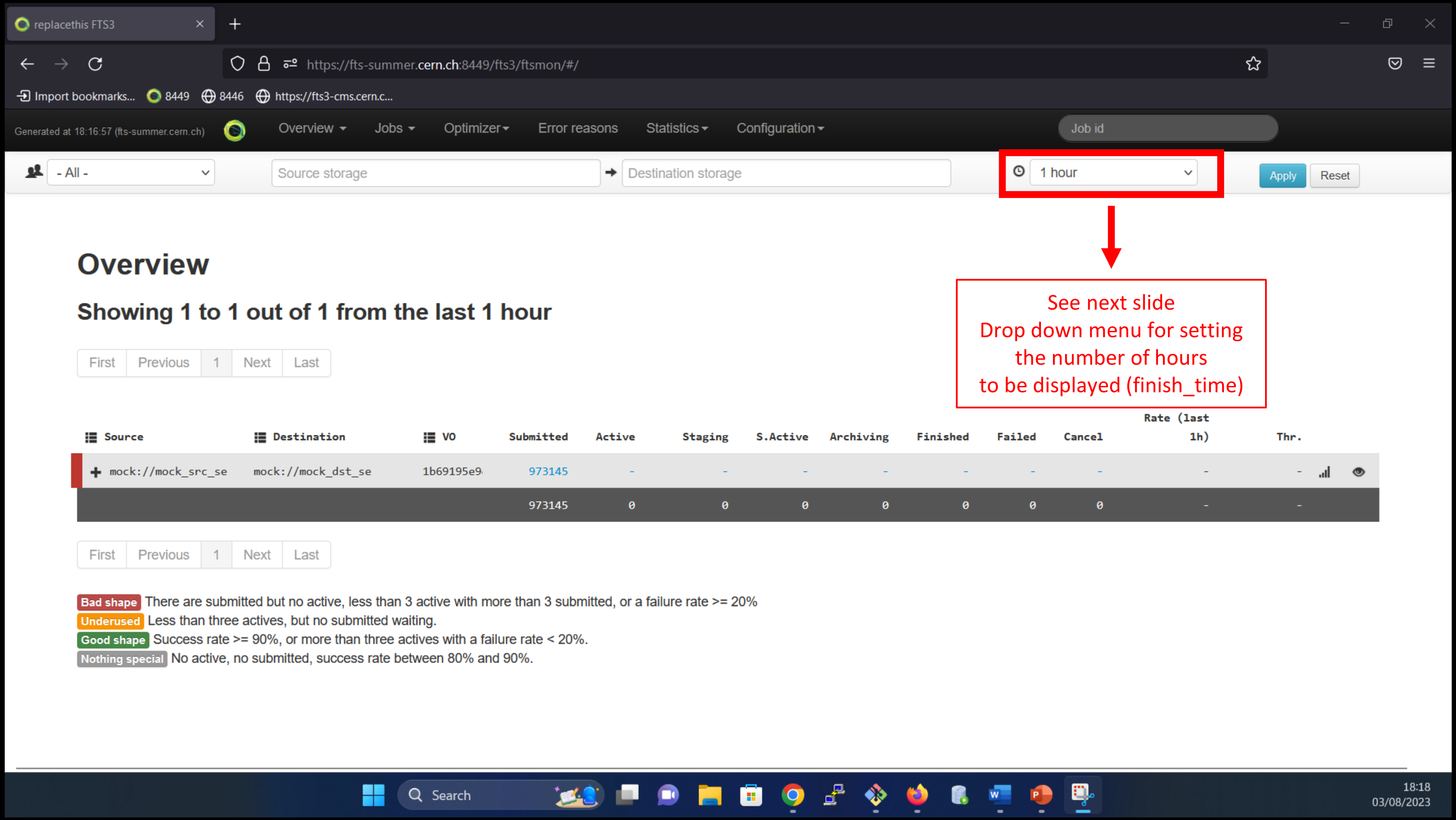
File Transfer Life Cycle



See next slide - Drop down menu for setting the number of hours to be displayed (finish_time)

Identified Slow Web Monitoring

Web Monitoring page shows us the stats for each source and destination pair



replacethis FTS3
https://fts-summer.cern.ch:8449/fts3/ftsmon/#/
Import bookmarks... 8449 8446 https://fts3-cms.cern.c...
Generated at 18:16:57 (fts-summer.cern.ch)
Overview Jobs Optimizer Error reasons Statistics Configuration
Job id
- All -
Source storage
Destination storage
1 hour
Apply
Reset
First Previous 1 Next Last
Search
18:18
03/08/2023

Overview

Showing 1 to 1 out of 1 from the last 1 hour

Source	Destination	VO	Submitted	Active	Staging	S.Active	Archiving	Finished	Failed	Cancel	Rate (last 1h)	Thr.
mock://mock_src_se	mock://mock_dst_se	1b69195e9	973145	-	-	-	-	-	-	-	-	-
			973145	0	0	0	0	0	0	0	-	-

Bad shape There are submitted but no active, less than 3 active with more than 3 submitted, or a failure rate >= 20%
Underused Less than three actives, but no submitted waiting.
Good shape Success rate >= 90%, or more than three actives with a failure rate < 20%.
Nothing special No active, no submitted, success rate between 80% and 90%.

See next slide
Drop down menu for setting
the number of hours
to be displayed (finish_time)

The Slow Web Monitoring Queries

```
SELECT COUNT(file_state) as count, file_state, source_se, dest_se, vo_name
FROM t_file
WHERE file_state in ('SUBMITTED', 'ACTIVE', 'READY', 'STAGING',
'STARTED', 'ARCHIVING’)
GROUP BY file_state, source_se, dest_se, vo_name order by NULL;
```

```
SELECT COUNT(file_state) as count, file_state, source_se, dest_se, vo_name
FROM t_file
WHERE file_state in ('FINISHED', 'FAILED', 'CANCELED’)
        AND finish_time > '2023-08-03 15:16:48’
GROUP BY file_state, source_se, dest_se, vo_name  order by NULL;
```

The Web Monitoring Query Problems

- FTS uses InnoDB which is a Multi-Version Concurrency Control (MVCC) based database storage engine. It prevents reads from blocking writes and vice versa but
  - Encourages counters to be implemented as the memory intensive solution of counting millions of rows every time.
  - No counters stored in MySQL database to fetch in constant time.
  - Only works due to excessive amount of database RAM/cache.
  - Power cut requires 80 GBs of RAM to be filled/warmed up.

Proposed Solution

Sliding window histogram of monitoring counters

- Web monitoring requires counts per source and destination and during last 1, 2, … 6 hours
- Counter will be incremented whenever there is an insert in the DB and decremented in case of delete or update.
- Constant time counters instead of recounting rows within "entire" database.

Thank You!