**University of Waterloo**

Faculty of Mathematics

**Improvement on the versatility of**

**industrial 3D camera server-side data transmission**

Mech-Mind Robotics

Camera Team

Beijing, China

Prepared by

Mingze Xu

ID 20806115

3A Computer Science

April 20, 2021

**MEMORANDUM**

To: Evaluator

From: Mingze Xu

Date: April 20, 2021

Re: Work Report: Improvement on the versatility of industrial 3D camera server-side data transmission

As we agreed, I have prepared the enclosed report, "Improvement on the versatility of

industrial 3D camera server-side data transmission" for my 3A work report and for Mech-mind Robotics. This report is my third report and has not received academic credit.

The camera team of Mech-Mind Robotics is managed by YouShuang Ding and aims to develop industrial 3D cameras that generates high-quality 3d data in various situations. My job as a software developer required me to implement new features, as well as stabilizing the product and improving the performance. This report aims to provide an analysis on improving the versatility of the server-side data transmission.

The Faculty of Mathematics requests that you evaluate this report for command of the topic and technical content/analysis. Following your assessment, the report, together with your evaluation, will be submitted to the Math Undergrad Office for evaluation on campus by qualified work report markers. The combined marks determine whether the report will receive credit and whether it will be considered for an award.

Thank you for your assistance in preparing this report.

*Mingze Xu*

# Summary

This report introduced the basis of recovery in Postgresql, identified the demand for performance improvement in Postgresql recovery, and issued an approach of parallel recovery.

By analysis of the native design of Postgresql recovery and replication, the potential performance improvement could be achieved by parallelizing the recovery process. Then parallel replay in MariaDB is researched and analyzed. However, the method used in MariaDB cannot be directly applied to Postgresql due to the difference in recovery facility and logging style. After digging into the design of Postgresql recovery design, an approach of parallel recovery is issued, which dispatches all redo logs in terms of hash on the page number of the page that the redo log records operate on, in order to avoid possible conflicts in parallel replay process. This report also analyzes the disadvantage of the issued approach.

# Introduction

This section aims to provide necessary background information about Mech-Eye, the industrial 3D camera product of Mech-Mind Robotics. It briefly introduces the working principle as well as the server-client software pattern of the camera.

## Working principle of industrial 3D camera

The “industrial 3D camera” occurred in the remaining report refers to the camera that reconstructs the three-dimensional information of the target object based on the two-dimensional images captured. It is commonly used to recognize the target features, which acts an essential role in a wide range of smart industrial applications such as Vision-Guided Machine Tending, Steel Bar Labeling, Welding Spot Detection, etc. (Mech-Mind, Applications, Retrieved from <https://en.mech-mind.net/case_complex.aspx?FId=t4:30:4&TypeId=30>)

The development of 3D cameras fully relies on the three-dimensional measurement technology , which has been proposed and developed for about 70 years, from contact, passive measurement to non-contact, active measurement. The fringe pattern projection method, which is one of the most popular 3D measurement technology, is widely used in modern 3D cameras due to its high speed and accuracy. It involves 2D digital cameras and additional light sources that could generated structural light. Multiple images under phase-shifting grating lights are captures and analyzed to obtain the 3D information of the target object.

## Server-Client data transmission model

This section will focus on the software architecture of industrial 3D cameras and introduce the client-server data transmission model.

Client-server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. (Sun Microsystem, retrieved from //web.archive.org/web/20110406121920/http://java.sun.com/developer/Books/jdbc/ch07.pdf).It is a quite a classic structure that used by many applications such as the world wide web and email. The 3D camera software is also well suited to this model, where the server is rooted in the camera in order to capture and process the images, manipulate the hard device, and reconstruct the 3D information when receiving requests, and the client could be started from some other devices, sending requests to the server through the wired or wireless connection in order to control the camera. The client-server model isolates the computation work from users and offers one-to-many connections.

Clients and servers communicate in a request-response manner, where clients send a request message to the server and the server replies with a response message. Such communication method requires the client and the server to understand what is each other saying, that is, a protocol that both sides acknowledge is required for encoding and decoding the request/response messages. Such protocols may be established from scratch, or stem from some industrial standard.

There are some well-accepted low-level data transmission protocols in the industry of camera such as the GigE Vision and USB-3 Vision, which specifies how to transmit high-speed video and related control data over ethernet networks or USB 3.0. (move to later part)

# Existing Problems

This section aims to identify the existing problems regarding the versality of server-client data transmission in the current model and dig in the root cause of the problems.

## Current protocol in the software

# Analysis

This section aims to analyze the feasibility of parallel recovery in Postgresql. Parallel recovery in another widely used open-source DBMS MariaDB will be referred to.

## How MariaDB Handle Parallel Recovery?

Recall that Postgresql does not log SQL statements in xlog, instead, the instruction of modifying a specific row is logged. This logging method is often named "row-based logging". Replication and Recovery in MariaDB by default uses mixed logging, which is a combination of SQL-statement-based logging and row-based logging. Statement-based logging is used by default, but when the server determines a statement may not be safe (eg. CREATE TABLE, TRUNCATE TABLE) for statement-based logging, it will use row-based logging instead. (MariaDB, Mixed-logging, Retrieved from https://mariadb.com/kb/en/binary-log-formats/#mixed-logging)

In-order parallel replication are introduced in MariaDB 10.1.3 (MariaDB, , Parallel Replication, Retrieved from <https://mariadb.com/kb/en/parallel-replication/>). In in-order parallel replication mode, any transactional DML (INSERT/DELETE/UPDATE) is allowed to run in parallel. They are dispatched to different threads and replayed independently. This may cause conflicts, for example, when two statements are modifying the same row. In this case, the latter of the two statements is rolled back, so that the former could proceed. Non-transaction DML and DDL, which are mostly row-based logged, is not safe to be replayed parallelly since it cannot be rolled back. Thus those statements are not applied in parallel.

## Can the same method be applied to Postgresql?

As mentioned above, Postgresql uses row-based logging for all operations, which means only instruction of modifying a specific row is logged. During recovery, the specific row will directly be modified without being locked, which makes it difficult to identify if a given row is modified by two transactions at the same time. Thus, even transactional DML cannot be run in parallel using the same method in MariaDB.

## A possible approach to the problem

Since the native structure of xlog makes it hard to detect conflicts in parallel recovery, it is critical to avoid conflicts in advance. Notice that conflicts only happens when the same row is modified by multiple threads at the same time, then a possible approach is to dispatch all xlog records by hashing the page number of the page where the row being modified is located, as a result, all the xlog records operating on the same page are grouped together and assigned to the same process. Then each process replays the record assigned to it sequentially from the oldest to the latest. That could guarantee that no conflicts will happen between two different processes since they have to operate on different pages.

Furthermore, even non-transactional DML and DDL replay can be parallelized in this method after decoupling from the rollback feature. For example, when replaying the CREATE TABLE command, the xlog record can be dispatched by the page number of the first page of the table. However, this kind of recovery needs to be synced with other threads since all operations on this table must be replayed after the table being created.

## Disadvantages of the approach

Since the xlog records are strictly dispatched by hashing on the page number, then it could not handle some extreme cases. For example, when most of the operations are operated on the same page while only a few operations are on other pages, one process has to deal with most of the xlog records, which could significantly drop the performance of parallel replay.

# Conclusions & Recommendations

After analysis on both MariaDB and Postgresql recovery design, the conclusion can be drawn that different approach is required in Postgresql. A possible method is to dispatch all xlog records according to the page number so that conflicts could be avoided. Although this method will lose performance improvement in some extreme cases, it could save recovery time in most cases since the redo workload is parallelized.

# References

[1] HUAWEI TECHNOLOGIES. (2017). WO 2017/122060 Al. “PARALLEL RECOVERY FOR SHARED-DISK DATABASES”. Retrieved from https://patentimages.storage.googleapis.com/65/e8/0c/32b4ee42fd81cb/WO2017122060A1.pdf.

[2] MariaDB (2015), Mixed-logging, Retrieved from https://mariadb.com/kb/en/binary-log-formats/#mixed-logging

[3] MariaDB (2015), Parallel Replication, Retrieved from https://mariadb.com/kb/en/parallel-replication/

# Acknowledgments

I would like to sincerely thank my manager, Sherman Lau, for helping me to brainstorm the ideas for this report, providing advice on the structure of this report, and proofreading this report.