Simeon Stoykov
Christ's College
ss2476

Dissertation Project Proposal

[Draft] Snapshot Isolation in Heterogeneous Storage Systems

17 Oct 2019

Project Originator: Martin Kleppmann

Project Supervisor: Martin Kleppmann

Signature:

Director of Studies: Richard Mortier

Signature:

Overseers: Marcelo Fiore and Amanda Prorok

Introduction

Modern large-scale applications often employ multiple data storage systems that are optimised for different access patterns. Some of the examples are:

1. OLTP relational databases, that are used to handle frequent state updates

   Example: PostreSQL

2. Search indices, optimised for full-text search

   Example: Lucene, or derivatives such as ElasticSearch

3. In-memory caches, that store the results of expensive queries in order to improve performance

   Example: Memcached, Redis

Apart from performance, another important aspect of the "polyglot" storage infrastructure are the guarantees it provides in terms of atomicity, consistency, isolation and durability. Storing the same data in multiple forms introduces implicit dependencies across the different storage systems. Even though each one provides consistency/isolation guarantees on its own, using them together could lead to partially updated state and hence expose inconsistencies to the user.

Example: Picture a message service with a database for the messages in a conversation thread, alongside a search index. The lack of isolation might lead to a temporary state, in which the search index has been added messages that are not present in the other database yet. Hence, when the user searches for a message, they could get results which are not found in the conversation thread yet.

Ad-hoc solutions to this and related challenges might be difficult, inefficient or simply incorrect.

The "Online Event Processing" paper by Martin Kleppmann, Alastair R. Beresford, and Boerge Svingen describes a general approach which sequences the transactions through a reliable log, to which all storage systems are subscribed. In this way the updates will be performed in the same order, by all systems, even in the face of transient failures, as long as the log provides those features. A simplified pipeline of the OLEP approach can be seen in figure 1.

One thing that the "Online Event Processing" paper does not discuss in depth is how to ensure the consistency of any read-only queries that are not sequenced through the log. This falls under the topic of isolation, and a common solution is to provide snapshot isolation, which give the follow guarantees:

1. All transactions see a consistent state of the database(s), from beginning to termination

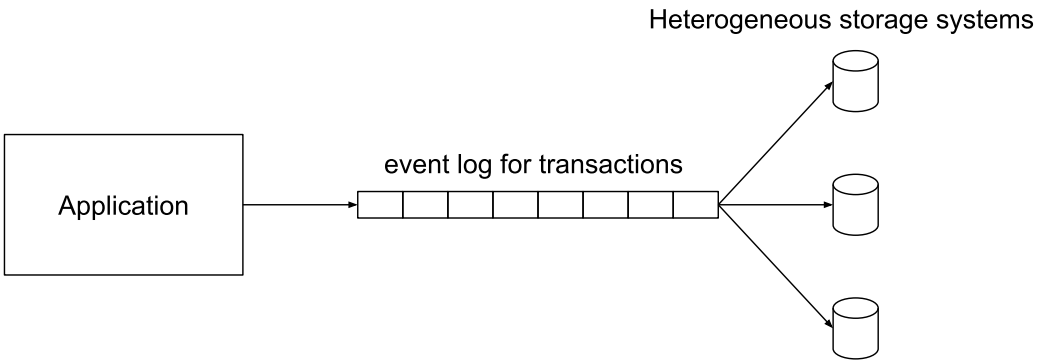

Figure 1: Simplified logical view of the OLEP approach

2. Readers do not slow down writers, and vice versa

My project will implement and build on the "Online Event Processing" paper by also providing snapshot isolation guarantees across multiple heterogeneous storage systems. I will explore various snapshot isolation implementation approaches, discuss the trade offs between them, and also include a comparison to existing solutions for achieving that, if applicable.

Resources Required

The main resources I plan to use are:

1. Apache Kafka — used for implementing the event logs used in the OLEP pipeline between the client and the heterogeneous database systems
2. PostgreSQL — as an example of ubiquitous relational database system used in the industry
3. Lucene — a full-text index which will serve as an example of the different natures of storage systems that could be used

As all of those are open-source, no commercial licenses are required.

The project will be written in Java, which was well covered in the Computer Science Tripos. Additionally, it is a natural fit to the project since all of the aforementioned technologies provide APIs in it.

Starting Point

The inspiration of my project is the "Online Event Processing" paper by Martin Kleppmann, Alastair R. Beresford, and Boerge Svingen, which describes the event-oriented approach for databases, that I aim to use.

Another valuable resource on the topic is Jack Wickham's 2019 part-ii dissertation which provides an implementation of the TPC-C benchmark in an "Online Event Processing" style and compares it to existing database systems.

The book "Designing Data-Intensive Applications" by Martin Kleppmann contains a detailed chapter on isolation, and in particular snapshot isolation. In general, the book covers a broad range of topics in distributed storage systems, so I anticipate that it's going to be useful throughout the course of writing my dissertation.

I have a basic high level understanding of the technologies being used (Apache Kafka, PostgreSQL, Lucene). I have no hands on experience with either of them.

Substance and Structure of the Project

The project will be a proof of concept for snapshot isolation across multiple independent storage systems. In order to demonstrate the usefulness of the guarantees it offers, I will also implement an example application which requires snapshot isolation and employs heterogeneous distributed storage systems.

The example application will be a simple messaging service, that provides a full-text search across conversations. In order to achieve that, it will make use of the following storage systems:

1. PostgreSQL — relational database that stores all messages sent
2. Lucene — full-text search index which will handle full-text search queries
3. *(optional)* Redis — an in-memory cache, that will maintain an unread messages counter

In general, in order to achieve snapshot isolation across storage systems, the following invariant should be preserved: *At each point in time, there should be a consistent version*

of the data that is contained in all storage systems in use. One of the advantages of the OLEP-based storage systems, however, is that each system is allowed to progress through the transaction requests on its own, in an asynchronous manner. Thus, both statements together outline the following two approaches for keeping the storage systems in sync and hence providing snapshot isolation:

1. The storage systems which have processed more transactions wait for the ones containing stale data to catch up with them. After that a read transaction block is opened.
2. Every storage system keeps a stale versions of the data up to and including the point in time at which the least up-to-date system is at.

As a result, different implementation branches will stem from the two.

Success Criterion

The project will be deemed successful if it provides an implementation of snapshot isolation for the example application, with quantitative evaluation of performance and comparison to that of existing solutions.

Correctness tests:

1. Ensure that the pipeline correctly processes the exact transactions it is given, without dropping or reordering them
2. Ensure that results returned by different databases are consistent with each other (snapshot isolation)

Performance tests:

1. Based on the number of transactions
2. Based on failure of transactions or nodes

Timetable and Milestones

Milestones:

1. Research (25 Oct 2019 — 01 Nov 2019)

   (a) Explore the technologies used (Apache Kafka, PostgreSQL, Lucene, Redis), gain some hands on experience, research their transaction semantics
   (b) Outline concrete implementation approaches for snapshot isolation

2. OLEP & example application implementation (01 Nov 2019 — 22 Nov 2019)
   (a) Implement the basic OLEP pipeline without snapshot isolation. It should support PostgreSQL and Lucene as data storage technologies
   (b) Implement the example application and integrate it with the OLEP pipeline
   (c) Demonstrate the need for snapshot isolation with a failing test which produces an inconsistent output

3. Implement the simplest form of snapshot isolation (22 Nov 2019 — 06 Dec 2019)
   (a) Demonstrate that the failing use case in 2c is now resolved by employing snapshot isolation
   (b) Quantify the cost of adding snapshot isolation by comparing the performance of the systems with and without it

4. Improve the performance of the system by implementing a better snapshot isolation approach (13 Dec 2019 — 20 Dec 2019, and 03 Jan 2020 — 17 Jan 2020)
   (a) The new implementation should pass the failing use case in 2c
   (b) Includes a performance comparison with the previous version, as well as with the initial system without snapshot isolation

5. If time allows, add an additional storage system to the architecture — a Redis cache (17 Jan 2020 — 14 Feb 2020)
   (a) Add Redis to the data storage systems in the OLEP pipeline
   (b) Add an unread messages counter functionality to the messaging application
   (c) Integrate them together

6. Evaluation & beginning of work on the dissertation text (14 Feb 2020 — 13 March 2020)
   (a) Implement X/Open XA distributed transactions, and compare their performance with the existing snapshot isolation approach
   (b) Provide more performance tests (e.g explore how well the systems scale, or handle failures)
   (c) Improve abstractions and overall code quality

7. Focus on writing the dissertation and make minor additions to the code driven by the text's needs (13 March 2020 — 10 April 2020)

8. Polish the dissertation text (10 April 2020 — 01 May 2020)