In for a Surprise when Migrating NoSQL Data

Uta Störl, Alexander Tekleab
Darmstadt Univ. of Applied Sciences, Germany
uta.stoerl@h-da.de

Meike Klettke
University of Rostock, Germany
meike.klettke@uni-rostock.de

Stefanie Scherzinger
OTH Regensburg, Germany
stefanie.scherzinger@oth-regensburg.de

***Abstract*—Schema-flexible NoSQL data stores lend themselves nicely for storing versioned data, a product of schema evolution. In this lightning talk, we apply pending schema changes to records that have been persisted several schema versions back. We present first experiments with MongoDB and Cassandra, where we explore the trade-off between applying chains of pending changes *stepwise* (one after the other), and as *composite* operations. Contrary to intuition, composite migration is not necessarily faster. The culprit is the computational overhead for deriving the compositions. However, caching composition formulae achieves a speed up: For Cassandra, we can cut the runtime by nearly 80%. Surprisingly, the relative speedup seems to be system-dependent.**

Our take away message is that in applying pending schema changes in NoSQL data stores, we need to base our design decisions on experimental evidence rather than on intuition alone.

***Index Terms*—NoSQL databases; schema evolution; data migration; composite migration**

I. Question

We consider the scenario of data evolving over time and stored in a schema-flexible NoSQL data store. Different from related work [1], we assume that legacy data resides in one of *several* legacy schema versions. This is a scenario often faced by agile development teams, as frequent releases imply equally frequent schema changes. Consequently, when a record is loaded into the application, several schema pending changes may have to be applied.

While NoSQL data stores are popular in agile application development, there is still little systematic tool support for large-scale data migrations. In building such a tool, we set out to explore a seemingly straightforward hypothesis:

> *Carrying out composite operations that perform several schema changes at once, rather than applying them one by one, is more efficient.*

In earlier work, we have introduced the theory for composing operations which *add*, *remove*, and *rename* a property, as well as *copy* and *move* properties [2].

As discussed next, in our experiments with MongoDB and Cassandra, we were in for a surprise.

II. Results

We run our experiments against MongoDB v3.4.2 and Cassandra v3.9. As hardware, we use a Dell PowerEdge C6320 with 2 Intel Xeon E5-2695v2 CPUs and 128GB RAM. We carry out schema changes by calling the native database APIs.

Figure 1 shows the total accumulated runtimes for stepwise and composite migration, executed on 10,000 persisted entities. For the evaluation, we consider a chain of five operations: `add`, `add`, `rename`, `rename`, and `delete`. This chain can be collapsed to a single, composite `add` [2]. As described in [2], we can also collapse more complex chains into composite migration operations.

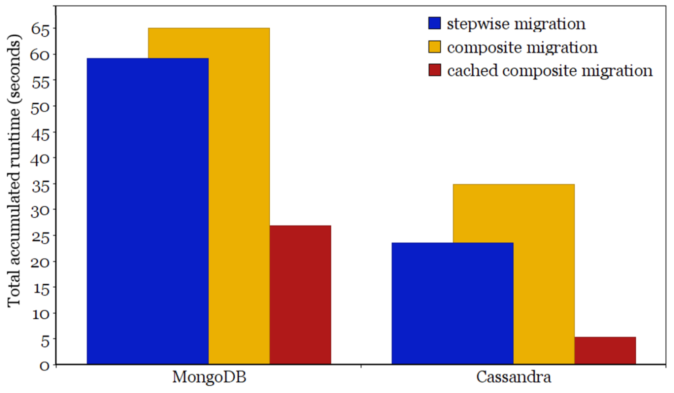

Fig. 1. Total accumulated runtime for applying five pending schema changes to 10,000 entities, realized as stepwise or composite migration.

Surprisingly, composite migration is not immediately faster: Closer analysis reveals that it is the overhead of computing the compositions repeatedly, which falsifies our hypothesis. Interestingly, the relative difference in runtime is more striking for Cassandra, and not as noticeable for MongoDB. Thus, the runtime behavior is highly system-specific. By introducing a cache and storing the computed composition formulae, we can effectively reduce the runtimes when compared to the stepwise approach: As shown in Figure 1, we achieve a reduction by about 50% for MongoDB, and by nearly 80% for Cassandra.

Our take away message is that in architecting a scalable data migration tool for NoSQL data stores, careful engineering is called for. In particular, a thorough experimental analysis is required, since tackling this task is not as straightforward as might be expected.

Acknowledgement

This project was funded by the *Deutsche Forschungsgemeinschaft* (DFG, German Research Foundation), grant #385808805.

References

[1] K. Saur, T. Dumitras, and M. W. Hicks, "Evolving NoSQL Databases Without Downtime," in *Proc. ICSME'16*, 2016.

[2] M. Klettke, U. Störl, M. Shenavai, and S. Scherzinger, "NoSQL Schema Evolution and Big Data Migration at Scale," in *Proc. SCDM'16*, 2016.