Safe Combination of Data-centric and Operation-centric Consistency

Mirko Köhler
Technische Universität Darmstadt
koehler@cs.tu-darmstadt.de

Guido Salvaneschi
Universiät St. Gallen
guido.salvaneschi@unisg.ch

Distributed system programming necessitates dealing with maintaining consistency across replicated data. When dealing with such data, there is a trade-off between two key properties: *availability* and *consistency*. Availability implies that a process can access some data irrespective of the system's state [1]. Instead, consistency implies that all processes maintain a shared view of the system.

Strong consistency, which makes development simpler by giving the illusion of a single process, means they are interacting with data that is up-to-date, with a clear order of modifications. However, achieving consensus requires coordination among replicas to ensure a globally consistent view, leading to substantial performance costs that negatively impact availability. To address these issues, local-first software leans towards *Weak* consistency (availability) and eases the constraint of a shared view. Instead, processes operate on local state, which can be integrated into the shared view eventually. Accessing local state is efficient, possible during network failures, and enhances privacy [2]. Software systems typically prioritize availability for efficiency, but certain functionalities still necessitate consistency. For instance, in a shop application, it's acceptable to use Weak consistency for parts of the application, e.g., displaying a slightly outdated quantity of available items is generally tolerable.

Choosing a single level of consistency, either Strong or Weak, for an entire application isn't practical in real-world scenarios. Instead, developers utilize a mix of consistency levels depending on the functionality. This approach, known as *mixed consistency*, can be observed, e.g., in online stores where it's usually acceptable to use Weak for most parts of the application to boost availability, e.g., displaying a slightly outdated quantity of available items is generally tolerable. However, the application needs Strong consistenc to determine who gets the last item upon checkout.

Over the past years, several languages have been proposed to help developers deal with the challenge of managing consistency. These solutions fall broadly into two categories: *Data-centric solutions* allow developers to explicitly assign consistency levels to data. This aligns well with object-oriented programming, where consistency and object identity often coincide. *Operation-based solutions* allow developers to specify constraints on operations, typically in the form of application invariants. This approach provides the flexibility to access the same data with different consistency levels for specific operations as needed. In reality, both approaches are valuable within the same application, as developers often need to balance the requirements of object-oriented design (which is best served by data-centric consistency) with the need for flexibility in accessing data (which is best served by operation-based consistency). However, as of now, there's no existing solution that allows for a mix of data- and operation-based consistency.

Our approach. We present `ConOpY`, an object-oriented language that offers support for mixed consistent replication. The novelty lies in the innovative approach to incorporating operation-centric consistency, thereby enabling seamless reasoning about both data-centric *and* operation-centric consistency through a unified mechanism.

`ConOpY` offers to define consistency for entire replicated objects (data-centric) or for individual methods (operation-centric). These choices of consistency are enforced through the type system, enabling static reasoning about consistency. For the data-centric approach, developers annotate classes with specific consistency levels, resulting in replicated objects instantiated with the designated consistency level. All operations performed on these objects inherit the chosen consistency, ensuring a consistent treatment of the object's state. In the operation-centric approach, consistency is specified at the granularity of methods. Developers attach consistency annotations to method declarations, and `ConOpY` infers the required consistency levels from these specifications. Importantly, `ConOpY` facilitates the seamless interoperability of these two approaches, allowing developers to combine them within the same application as needed. To ensure the correct mixing of consistency levels and prevent violations, `ConOpY` is equipped with a consistency type system that verifies the correctness of specified consistency levels with respect to each other, specifically ensuring that Strong data does not depend on Weak(er) data.

References

[1] Peter Bailis, Aaron Davidson, Alan Fekete, Ali Ghodsi, Joseph M. Hellerstein, and Ion Stoica. 2013. Highly available transactions: virtues and limitations. *Proceedings of the VLDB Endowment* 7, 3 (Nov. 2013), 181–192. https://doi.org/10.14778/2732232.2732237

[2] Martin Kleppmann, Adam Wiggins, Peter van Hardenberg, and Mark McGranaghan. 2019. Local-first software: you own your data, in spite of the cloud. In *Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward! 2019)*. Association for Computing Machinery, New York, NY, USA, 154–178. https://doi.org/10.1145/3359591.3359737