1 Introduction

Consistency is a key requirement for data-intensive applications. Various schemes exist to express application requirements ranging from traditional transactional requirements to weaker isolation models that sacrifice data safety for performance. In recent years, the rise of interactive applications has brought with it a new generation of programming frameworks. These frameworks enable data management through simplified request APIs to web-based data storage.

Foregoing strong consistency guarantees in favour of scalable performance, the task of ensuring application integrity is transferred to the application logic instead being ensured by a database management system [3]. Accepting anomalies leads to the requirement of compensation actions. For instance, a web shop has to issue cancellations if an inventory is over-sold. For some applications, the cost of compensation could outweigh the performance penalty of running a strongly consistent service. Other applications like social networks might not require any compensatory action at all because typical anomalies like off-by-one errors on follow-counters do not affect user experience. For application developers, a strongly consistent view on the database is the most intuitive approach since in this case application code does not need to consider anomalies. Existing solutions for adaptive consistency rely heavily on manual modification of application code, static invariant analysis and profiling, putting the burden of managing multiple consistency levels on developers.

In this proposal, I plan to outline a potential solution to adaptive consistency that omits consistency overhead from developers. While there has been some exploratory work into consistency SLAs [7], these approaches rely on closed form analytical models to deliver on the desired trade-offs. In my PhD work, I will investigate model-free online learning techniques that can flexibly learn SLAs without relying on explicitly encoded knowledge of the infrastructure.

2 Background and related work

A multitude of adaptive consistency and distributed transactions schemes has been proposed to leverage weaker isolation levels for transactions as well as single record consistency under specific workloads [1]. Recent work has proposed static invariant analysis to identify transaction types that can execute concurrently without requiring coordination (synchronisation through locking and message passing)[2].

Kraska et al. have suggested a consistency rationing scheme [4] where data is pre-classified into three categories. The first category (A) concerns data where serializable guarantees are necessary because inconsistencies cannot be recovered or are too expensive to resolve. The second category (B) concerns data where anomalies can be tolerated but compensation is not free. Finally, the third category (C) comprises all data where inconsistency requires no compensation or cannot occur. The key insight of Kraska et al. is that for data in category (B) an adaptive scheme can be used to switch between various consistency guarantees depending on an estimate of the conflict probability and the compensation cost. Consistency hence becomes a probabilistic guarantee.

Similarly, Li et al. have proposed a coordination scheme called RedBlue consistency [5], which divides operations into blue (fast and eventually consistent) and red (strongly consistent and slow). Operations are classified based on application invariant analysis. Li et al. have implemented RedBlue consistency on a distributed coordination architecture called Gemini. On a geo-replicated multi-site (5 locations) deployment, Gemini achieves superior performance even to a strongly-consistent single-site architecture.

Finally, Terry et al. have described an adaptive consistency scheme for cloud services that offers consistency-based service level agreements (SLAs) [7]. Users can define utility levels for specific response times and consistencies so that for each read operation, an optimal response-time/consistency trade-off can be determined. This is achieved through a client-library that estimates the probability of reading a certain isolation level from a replica based on tracking time stamps retrieved from each replica.

The common theme of these approaches is the reliance on developers being able to classify data, detect invariants and rewrite application code to comply with a specific consistency scheme. To date, none of these schemes has found wide-spread adoption, likely due to the prohibitive manual overhead.
3 Learning Consistency Trade-offs

Instead of having developers categorise data and having to deal with different consistency levels at an application level, adaptive consistency and transaction correctness can be maintained transparently by a middle-ware.

The proposed research will investigate the feasibility of using online learning to dynamically adapt consistency and transactional schemes in lieu of costly manual analysis. Specifically, in environments with dynamically changing workloads (e.g. web workloads on a Database-as-a-Service (DBaaS)) manual profiling of transaction code is not feasible. Instead, reinforcement learning could be used to adapt the transaction protocol in a dynamically changing environment. Reinforcement learning is a machine learning technique that is characterised by software agents that interact with an environment and learn optimal behaviour through rewards on the actions they take [6]. Initially, the agent does not know how its actions change its environment and thus has to explore the space of available actions, i.e. consistency levels, cache placement, transaction protocols (locking or optimistic).

Reinforcement learning is often used in optimal control problems because it only requires as input a reward function that can be comprised of arbitrary performance metrics as defined in an SLA. For instance, the rate of successfully committed transactions within a given time window can be used as an optimization goal (in the case of correctness). Similarly, client latency, number of anomalies, cache miss rates, invalidation rates or request cost can be encoded. An interesting research question is the problem of how to embed various current system metrics into a state representation. Web workloads are non-stationary, both due to predictable cycles (time of day in different regions) as well as due to sudden popularity of content. The proposed research hence must investigate how to determine feasible trade-offs on converging towards a current workload but being able to adapt quickly to changes.

Further, the performance of contemporary web infrastructures relies on caching as a means to reduce physical latency as a performance bottleneck. An automated adaptive consistency scheme must hence also consider web caching infrastructure as a component of transactional semantics, i.e. optimistic transactions can read from local web caches. The decision of which content to cache is hence intimately linked to latency/consistency trade-off. Finally, the use of consistency SLAs as a means to convey application goals by developers will be investigated.

4 Challenges

The primary challenge of this work is to first develop an understanding of how application developers using web services think about consistency and transactions and how they would ideally communicate their data integrity requirements. Current approaches make optimistic assumptions towards developers being both able and willing to analyse and modify client-side code to comply with specific frameworks.

Further research problems concern the heterogeneous nature of backend infrastructures: Applications typically rely not on a single database but use a variety of services such as search servers, caches and so forth. With increasingly complex backend architectures, managing consistency in the application logic slows down development and introduces unforeseen anomalies. DBaaS providers should aim to deliver automated adaptive consistency levels and transactional choices.

References


