CS265 Spring 2016 research project
The RUM Conjecture and Access Methods

Keywords: access methods, read overhead, update overhead, memory overhead, RUM conjecture, RUM space

Problem: The database research community has been building methods to store, access, and update data for more than four decades. Throughout the evolution of the structures and techniques used to access data, access methods adapt to the ever changing hardware and workload requirements. Today, even small changes in the workload or the hardware lead to redesigning access methods. This phenomenon has reached its peak as data generation and workload diversification grow exponentially, and hardware advances introduce increased complexity. New workload requirements are introduced by the emergence of new applications, and data is managed by large systems composed of more and more complex and heterogeneous hardware. As a result, it is increasingly important to develop application-aware and hardware-aware access methods.

The fundamental challenges that every researcher, systems architect, or developer faces when designing a new access method are how to minimize, i) read times (R), ii) update cost (U), and iii) memory (or storage) overhead (M). This direction of research is based on the conjecture that when optimizing the read-update-memory overheads, optimizing in any two areas negatively impacts the third. The high level goal is to study the manifestation of the balance of the RUM overheads in state-of-the-art access methods, and pursue a path toward RUM-aware access methods for future data systems.

Project: In this project we first ask the question how does a data structure aggressively optimized for only one of the three aforementioned overheads look like. The project’s goal is to implement the three “optimal” data structures as described in the RUM Conjecture paper [1] (Section 2) and build a system that can alternate between the three based on the workload as input. The workload consists of reads, updates, deletes, and inserts, and we assume a simple key-value model: when searching for a specific key (according to which data is organized) we get back a value (which in our case is the logical id of the corresponding row). As part of this project we expect the students to build a simple demo, that will show the
performance of a workload when executing it with a specific choice of access method, as well as the values of the three basic RUM overheads.

There are more open research problems in this direction. Students may explore any open topic in consultation with the instructor.

References

Supervisor: Prof. Stratos Idreos (stratos@seas.harvard.edu)

Responsible collaborator(s): Manos Athanassoulis (manos@seas.harvard.edu)
Lukas Maas (maas@seas.harvard.edu)