On-Disk Datalog Engine using Semirings

Nick Brown* Oliver Kennedy

University at Buffalo
{ njbrown4, okennedy }@buffalo.edu

1 Introduction

Program analysis is the search for patterns in and/or gathering of statistics over a program's source code, typically expressed in the form of a tree and/or a directed acyclic graph. Such analyses frequently involve very large state spaces — one analysis over the chrome source code conducted by an author generated 64GB of intermediate state. A typical developer workstation (~8 CPU cores and 16-64GB of RAM) simply can not perform such an analysis in-memory. Recent efforts revolve around the Datalog programming language, leveraging its similarity to standard inference rules to identify opportunities for parallelization [4] or incrementality [1]. Efforts to scale-up analysis have focused on the use of graphics cards or large-scale distributed systems. However, moving program analysis off local hardware is expensive, and may simply not be feasible for (e.g., for proprietary code or in restricted settings).

In this talk, we argue for local-only program analysis on commodity hardware by using out-of-core database techniques. We outline our implementation of an on-disk datalog engine based on the Aggregate Calculus [3], specifically focusing on its ability to leverage group-theory for more efficient storage and access of intermediate analysis state.

2 Semiring (and Ring) Relations

In [2], Green et. al. introduced semiring relations; relations where each tuple is associated with an annotation drawn from a mathematical abstraction of addition and multiplication called a semiring. This annotation may encode a range of things, from the presence or multiplicity of the tuple, all the way up to a full derivation of its existence from the source data. Semiring relations, and the related Ring relations [3] admit a lazy evaluation of pipeline blockers; deferring costly materialization to a later stage of a streaming pipeline. Consider the tuple $t = \langle 1, 2 \rangle$; A relation encoded as $R_1 = [\ t \to 1, t \to 1\ ]$ (i.e., a list containing two copies of the tuple, annotated by a multiplicity of 1) is semantically equivalent to the encoding $R_2 = [\ t \to 2\ ]$ (i.e., a list containing a single copy of the tuple annotated by a multiplicity of 2). Deferring pipeline blockers is especially useful when the value of interest is a non-continuous function of the annotation (i.e., a morphism). For example, to test for existence, finding one copy of t in R_1 suffices; a full materialization into R_2 is not required.

3 Semirings on Disk

Our talk will overview our efforts to leverage [semi]ring relations to improve out-of-core performance of existing data structures. Our preliminary explorations have focused on B+Trees, a standard out-of-core index structure. Our key insights are that a single analysis may feature multiple distinct access patterns to a single dataset, some of which can be expressed through non-continuous morphisms (e.g., existential tests) over collections of tuples. Consider a relation $Call(C, F)$ that stores the caller C of every function F. We might access this relation by enumerating every function called by a block of code ($\pi_F(\sigma_{C=c_1}(Call))$), or by testing for the existence of a caller ($\sum_{\text{count}(*)}(\sigma_{F=f_1}(Call)) \geq 1$). The former requires an exact computation over the full set of relational elements where $C = c_1$. However, in the latter case, a nonlinear semiring morphism (≥ 1) is applied to the query result, allowing us to stop as soon as a tuple is found. Such early stop optimizations can be applied manually, but semiring relations allow us to automatically leveragel nonlinearities to stop enumerating tuples early.

References

[1] Darshana Balakrishnan, Carl Nuessle, Oliver Kennedy, and Lukasz Ziarek. Treetoaster: Towards an ivm-optimized compiler. In SIGMOD, 2021.

[2] Todd J. Green, Gregory Karvounarakis, and Val Tannen. Provenance semirings. In PODS, pages 31–40, 2007.

[3] Christoph Koch. Incremental query evaluation in a ring of databases. In PODS, pages 87–98, 2010.

[4] Ahmedur Rahman Shovon, Thomas Gilray, Kristopher K. Micinski, and Sidharth Kumar. Towards iterative relational algebra on the GPU. In USENIX ATC, 2023.

*Presenter