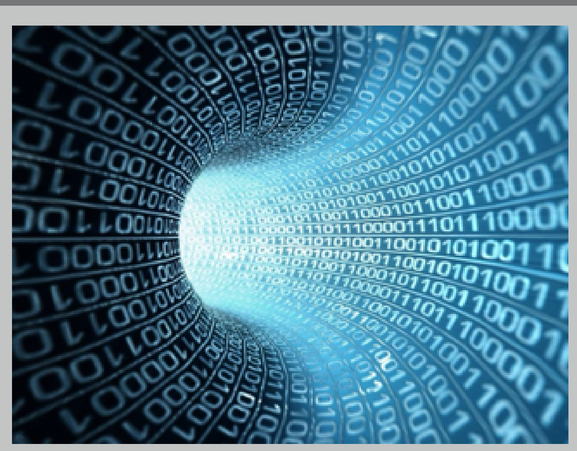
Optimization of Big Data Analytics using Intermediate Representation and Portable Containers

Giannis Tzouros, Michail Tsenos, Vana Kalogeraki

Department of Informatics, Athens University of Economics and Business, Greece

1. Introduction



- Modern applications employ libraries and algorithms like aggregates, top-K results, nearest neighbor machine learning etc. to process big data
- However, new types of data are introduced over time, leading to performance and efficiency challenges, as they are often not compatible with existing implementations
- To deal with these issues,compilers utilize Intermediate Representation (IR) to provide universal computing functions and cross optimizations between different libraries and algorithms
  - An IR provides an abstraction for incompatible libraries, hiding details about the target execution platform and expressing data tasks under a unique interface

2. Our Problem

The deployment of an IR heavily depends on the tools utilized and the scope of the big data environment. This may lead to problems concerning adaptability and portability when deployed on different execution environments.

3. Proposal

We propose a framework that provides a Java-implemented IR, which takes as input a context-free grammar with function structures and creates portable containers via a SableCC compiler

- The containers are easily deployable over multiple environments and multiple data processing frameworks, improving compilation and execution times
- The containers are also deployable on multi-cluster serverless environments, supporting parallel execution

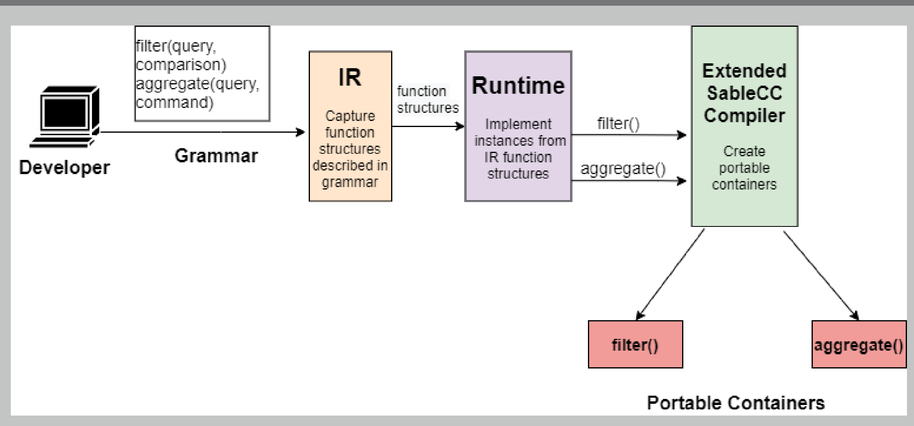
4. Design And Challenges

- Framework objective
  - Compiles computational functions described in an input grammar into portable containers
- Design Challenges
  - Containers must include an operational code which can be executed in multiple computing environments and support standalone usage
  - The output data of a container should be used by other containers without any extra changes, in order to optimize execution and reduced processing times between operations
  - The container's functions must be available for execution as many times as possible without having to rebuild the entire image of the function

5. Components

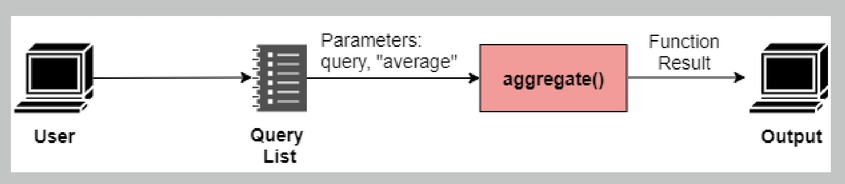
- The framework uses a grammar with function structures as an input, given by the developer
- Intermediate Representation: Captures the structures described in the grammar
- SableCC backend compiler: constructs containers based on function structures
- Portable containers: Deployable in multiple data environments, they execute computations on input queries

6. System



- Intermediate Representation
  - Our IR implementation compiles big data function structures into universal functions, ready to use in big data environments
  - The functions must contain well structured semantics in order to be converted to container executables
- Data Types
  - Scalars: int, long, double, char etc.
  - Lists: multiple values of Scalar types into a single data object e.g. String of multiple char values, dynamic array of numeric values, strings or key-value pair
- Functions
  - Big data operators that take as input a query list with multiple entries of Scalar values, (key-value pair or CSV line of multiple values). May require additional Scalars as parameters
    - filter(query, comparison), aggregate(query, command), valueScore(query), topKValue(query, k), topKScore(query, k), nearestNeighbor(query, x, y, k, g1, g2)
- Runtime Environment
  - Generates code for the function structures captured by the IR, takes the main components of the cached function: input, computations and output/result and recreates them in Java executable code
- Backend Compiler
  - Our framework uses an extended version of the SableCC backend compiler. The compiler converts the executable code created by the runtime for IR function into JAR executable files, which are then enclosed into portable containers
  - The containers can run on multiple environments and execute their code when they receive a query and associated parameters as input

7. Serverless Containers



These containers accept query list inputs from users and output the result either as a new query list (filtering), or within the screen (all functions besides filtering)

- Serverless Container Benefits
  - Lower operational and deployment cost (Pay-as-you-use pricing model)
  - High elasticity (Automatic resource allocation based on the current load demands, which scales to zero during idle periods)
  - Maintenance free services and independence of programming language and tools

Contact Information

- Giannis Tzouros
  - Email: tzouros@aueb.gr
- Michail Tsenos
  - Email: tsemike@aueb.gr
- Vana Kalogeraki
  - Email: vana@aueb.gr
- Real-Time Distributed Systems Group
  - Web: http://rtds.aueb.gr