Versatile Optimization of UDF-heavy Data Flows with Sofa

Astrid Rheinländer[1] Martin Beckmann[1] Anja Kunkel[1]
Arvid Heise[2] Thomas Stoltmann[1] Ulf Leser[1]
[1]Humboldt-Universität zu Berlin, Germany [2]Hasso Plattner Institut Potsdam, Germany
[1]{rheinlae, beckmann, akunkel, stoltman, leser}@informatik.hu-berlin.de
[2]arvid.heise@hpi.uni-potsdam.de

ABSTRACT

Currently, we witness an increased interest in large-scale analytical data flows on non-relational data. The predominant building blocks of such data flows are user-defined functions (UDFs), a fact that is not well taken into account for data flow language design and optimization in current systems. In this demonstration, we present Meteor, a declarative data flow language, and Sofa, a logical optimizer for UDF-heavy data flows, which are both part of the Stratosphere system. Meteor queries seamlessly combine self-descriptive, domain-specific operators with standard relational operators. Such queries are optimized by Sofa, building on a concise set of UDF annotations and a small set of rewrite rules to enable semantically equivalent plan rewriting of UDF-heavy data flows. A salient feature of Meteor and Sofa is extensibility: User-defined operators and their properties are arranged into a subsumption hierarchy, which considerably eases integration and optimization of new operators. In this demonstration, we will let users pose arbitrary Meteor queries and graphically showcase versatility and extensibility of Sofa during query optimization.

1. INTRODUCTION

While in the past analytical tasks commonly only involved relational data, many applications today build upon data flows processing large data sets containing both structured and unstructured data, such as web data, text, scientific data, etc. [4]. Still, most Big Data systems focus on the structured case especially when it comes to data flow optimization. Domain-specific functionality is implemented as user-defined functions (UDFs) with the UDF semantics hidden from the query optimizer [3, 8, 13]. There are systems that optimize queries within one domain (e.g., XLog for text [11]), but there the domain-specific optimizer logic is hard-wired into the system making it impossible to mix-in predicates from another domain, such as a web data extraction followed by natural language processing and data cleansing [9].

SIGMOD'14, June 22–27, 2014, Salt Lake City, UT, USA.

ACM ACM 978-1-4503-2376-5/14/06.$15.00.
http://dx.doi.org/10.1145/2588555.2594517.

Recently, we introduced Meteor, a declarative data flow language, and Sopremo, an extensible and semantically rich operator model for user-defined operators [5]. Optimization of Sopremo plans is carried out by Sofa, a logical optimizer for UDF-heavy data flows that seamlessly combines relational and user-defined operators from different domains into a single optimization framework [10]. Sofa builds on a concise set of properties for automatic and manual UDF annotation, which are evaluated by a cost-based optimizer using a small set of rewrite templates to infer semantically equivalent data flows.

In this demonstration, we illustrate the flexibility of Meteor and Sofa by showing diverse queries from different application domains (e.g., information extraction, web document analysis, data cleansing). Users are invited to play with Meteor's mechanisms for query refinement, intermixing operators from different packages, and its extensibility with new operators to support complex analytical tasks. They can step-by-step follow a graphical representation of the Sofa optimizer at work, including operator dependency analysis, inference of rewrites from operator annotations, cost-based optimization, and plan selection. Visitors will also have the option to interfere with query optimization by adding or removing operator annotations and changing cost parameters to study the impact on the resulting plans.

Meteor and Sofa are part of Stratosphere, a system for data analytics at large scale. Some facets of Stratosphere, e.g., the parallel execution model [1] or physical optimization [7] have been demonstrated before. However, this demonstration is the first showing Stratosphere's abilities in optimizing cross-domain UDF-heavy data analytics tasks, and also the first embracing the entire stack from Meteor down to the parallel execution engine.

2. BACKGROUND

In this section, we briefly introduce the Meteor query language, the algebraic layer Sopremo, and the Sofa optimizer. Figure 1 displays the overall system architecture of Stratosphere. Details can be found in [2, 5, 10].

2.1 The Meteor query language

Meteor is a declarative and extensible data flow language providing the user interface to Stratosphere. Meteor treats domain-specific functions as first-class operators, allowing users to combine these operators with a set of highly optimized relational operators without switching between different programming environments and paradigms. A main advantage of this approach is that the operator's semantics can

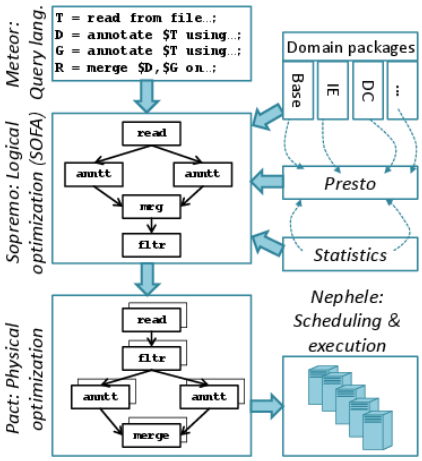


Figure 1: Architecture of Stratosphere.

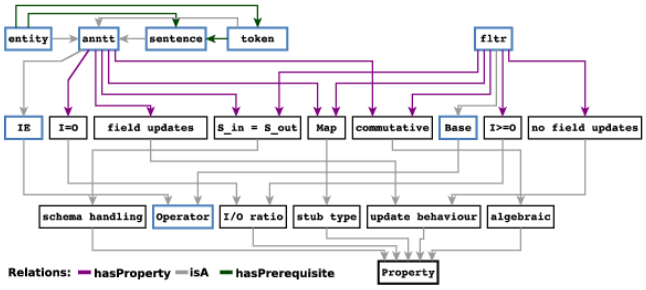


Figure 2: Presto annotations for selection (*fltr*) and IE operators (*entity*,*anntt*,*sentence*,*token*). Boxes depict operators (blue) and properties (black). Grey edges show *isA* relationships between operators and properties, respectively, pink edges display *hasProperty* relationships between operators and properties, and green edges show *hasPrerequisite* relationships between operators.

be accessed at compile time and can be used for data flow optimization (see below). Operator packages are self-contained libraries of the operator implementations, their syntax, and semantic annotations. More than 40 such domain-specific operators from the areas of information extraction (IE), data cleansing (DC), natural language processing (NLP), and data integration (DI) are already available, and more covering web data extraction and machine learning are shortly before release. During demonstration, we show different Meteor queries defining complex data flows build from operators from all available packages. Visitors are invited to formulate own Meteor queries and witness their queries undergoing the complete process of query translation, optimization, and parallel execution.

2.2 The Sofa optimizer

A major issue in optimizing UDF-heavy data flows is the diversity of the contained UDFs. Defining rewrite rules that respect the individual operator semantics for each possible combination of operators is merely impossible in UDF-rich systems such as Stratosphere. A particular problem is extensibility, as every new operator in principle needs to be analyzed with respect to all existing operators to identify rewrite options. Sofa solves this problem by means of Presto, an extensible taxonomy of operators, properties, and rewrite templates, and by reasoning along subsumption relationships encoded in Presto.

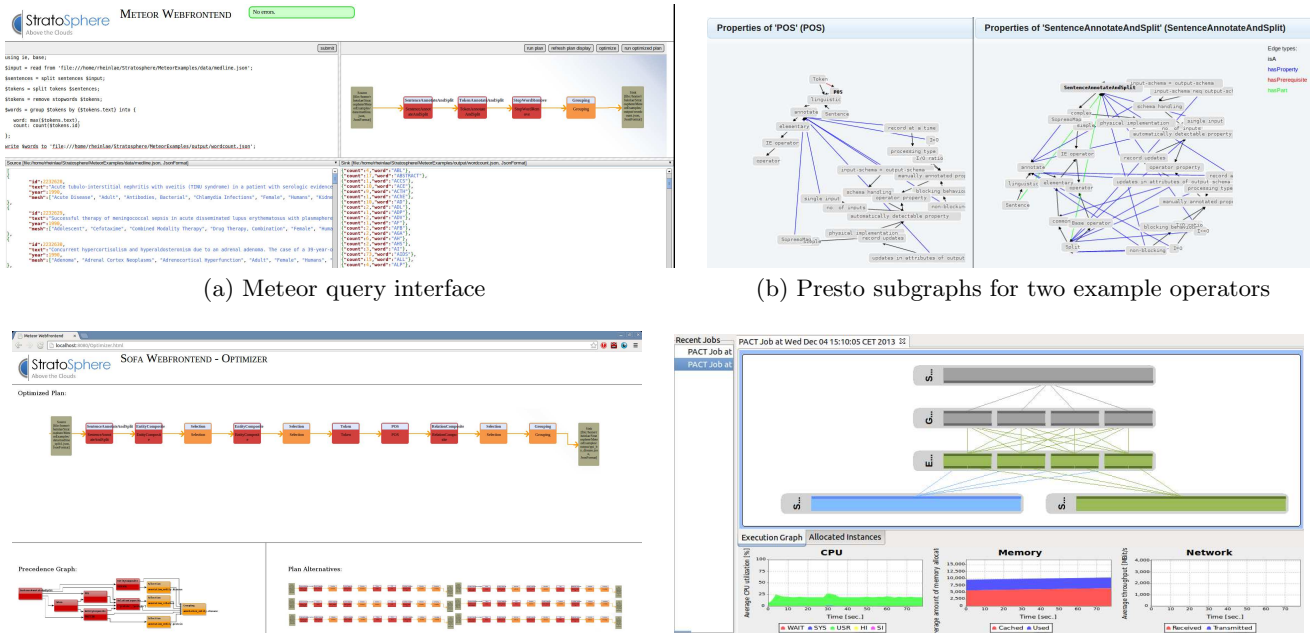
The principal ingredients of Presto are two taxonomies describing generalization-specialization relationships (*isA*) between operators and properties. Leaves in the operator taxonomy describe concrete implementations of the abstract parent operator, like different algorithms for entity extraction. Presto uses three additional relationships (*hasProperty*, *hasPart*, and *hasPrerequisite*) to model relations between operators and properties. Properties relevant for optimization are, for example, algebraic properties such as commutativity or associativity, the parallelization function (e.g., map, reduce), or the read/write behavior at attribute level. Figure 2 displays a snippet of Presto showing annotations for a selection operator (denoted as *fltr*) and different IE operators (denoted with *sentence, token, entity*). Rewrite templates are defined using Presto relationships, operator properties, and abstract operators as building blocks. Reasoning along Presto relationships allows Sofa to automatically instantiate the templates with concrete operators and enables discovery of individual rewrite options for concrete operator combinations on-the-fly. During demonstration, visitors can interactively navigate through Presto to explore relationships between operators and operator properties, and they can observe the effect on query optimization when Presto is changed.

Using Presto, Sofa performs three steps to enumerate alternative plans for a given data flow D: First, D is analyzed for precedence constraints between operators. This analysis yields a precedence graph used in the plan enumeration phase to secondly enumerate, and thirdly to perform cost-based ranking of valid plan alternatives. Finally, the best plan is selected, translated, and physically optimized for parallel execution by Stratosphere's execution engine. During demonstration, we show that Sofa is capable of intermixing predicates from different domains, of rewriting DAG-shaped data flows, and of finding semantically equivalent yet more efficient plans not found with previous work [6, 8, 12].

3. DEMONSTRATION SETUP

Our demonstration focuses on the Meteor language and the optimization process for UDF-heavy data flows using Presto. Visitors will be offered a selection of Meteor queries from different application domains such as biomedical IE, statistical NLP, and DI on Open Government Data to demonstrate the flexibility and adaptability of Meteor to various analytical tasks. Visitors are also invited to formulate their own queries. They will be able to experience Meteor's and Sofa's extensibility by exemplarily adding a novel web analytics package into the system during demonstration. We will also demonstrate how such new operators are seamlessly integrated into the optimization process using subsumption relationships from Presto.

Sofa optimizes Meteor queries and compiles the optimized plans to physical execution plans, which are executed in parallel using Stratosphere. We show in detail how the precedence analysis component derives information for optimization from analyzing operator properties, how plan alternatives are enumerated based on precedence analysis and cost estimates, and show the resulting plans ranked by estimated



(a) Meteor query interface

(b) Presto subgraphs for two example operators

(c) Sofa plan enumeration interface

(d) Stratosphere's parallel execution monitor

Figure 3: Screenshots of user interfaces shown during demonstration.

costs. In the following, we present the setup of our demonstration together with interaction opportunities for the audience.

3.1 Versatile data analytics with Meteor

Visitors of our demonstration experience the flexibility and ease of use of Meteor by means of different tasks in large-scale data analytics in and across different domains.

Domain-specific analyses. Starting with an IE query recognizing person names from Wikipedia articles, the audience can participate in evolving this query, for example to extract relationships between persons, or to change the subject of recognized entities and relationships, or even to perform complex extraction tasks such as determining the set of companies mentioned in Wikipedia, which went bankrupt in a certain time frame. They may switch the application domain towards biomedical texts by only slightly modifying the query. Visitors can also test the ability of our system to define tasks in the area of statistical NLP. Starting with a simple word count query on literature fiction, visitors can expand this query with a few NLP operators for stopword removal or stemming and further downstream operators to detect important topics or language motifs.

Cross-domain data analytics. We demonstrate a complex Meteor query analyzing a set of news websites for mentioned persons, locations, and companies. Our data set stems from a news crawl containing many duplicate articles, as different news articles are often copied from reports prepared by news agencies. For duplicate removal we apply different DC operators, which are followed by a series of IE and NLP operators performing linguistic analysis, entity and relation extraction. After each extraction operator, relational operators for filtering out texts containing no relevant mentions of entities are applied.

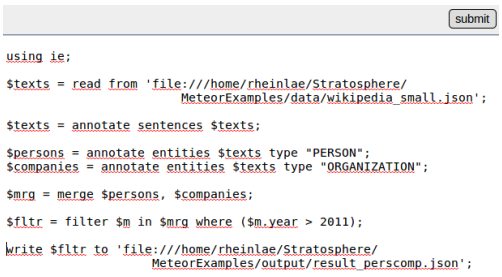
Figure 3(a) displays the Meteor user interface. Queries are typed into the text field on the upper left side, which is shown in greater detail in Figure 4(a) with a Meteor query for task-parallel annotation of person and company names in texts. After submitting the query, the translated yet unoptimized Sopremo data flow is displayed in the upper right part of the client (see Figure 3(a)). The bottom of the client shows a preview of the data to be analyzed (left side) and, after the query has successfully finished, a preview of the result set (right side). Next to viewing query translation, optimization, and execution of various queries, visitors will have the opportunity to write and submit own queries. During compilation, they get direct feedback from the system on lexical, syntactical, and semantically soundness of their queries.

3.2 Query Optimization and Execution

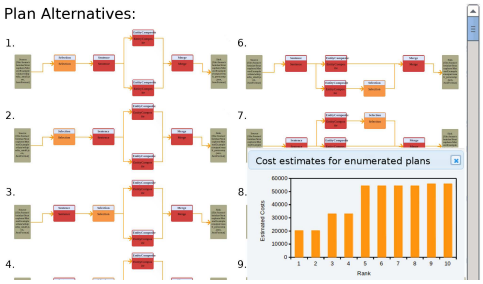
Sofa is designed to optimize complex, UDF-heavy data flows such as those introduced above. Visitors can experience the entire optimization process step-by-step and visually explore each phase of optimization carried out with Sofa.

Presto taxonomy and precedence analysis. After submission, a query is translated into an initial algebraic data flow as shown in the upper right of Figure 3(a). Boxes depict operators, data sources, and sinks, edges indicate the flow of the data. By clicking on an operator, relevant properties and relationships modelled in Presto can be inspected (see Figure 3(b)). During demonstration, the demonstrator explains available Presto annotations and relationships and their implications that enable or prevent operator reorderings. Visitors can interfere with query optimization by removing or adding facts to Presto and witnessing how this reduces or increases the set of possible plan alternatives for their queries.

Cost-based plan enumeration. Figure 3(c) displays



(a) Meteor query for task-parallel person and company annotation



(b) Enumerated plan space and pop-up window showing cost estimates for enumerated plans

Figure 4: Zoom into demo interfaces. Left: Meteor interface showing a query for task-parallel annotation of persons and companies. Right: Sofa interface showing plan alternatives ordered by estimated costs.

an overview of the plan enumeration visualization. The upper part in this window displays the best ranked plan. Plan alternatives are visualized in the bottom right part of the optimizer interface ordered by costs (see Figure 4(b)). Visitors can inspect any plan alternative together with its estimated costs select it for execution. The precedence graph, which results from precedence analysis in the previous step, is displayed in the bottom left part of the client.

Parallel execution. The selected logical data flow is translated into a Pact program, physically optimized, and submitted to the Stratosphere execution engine, which distributes and executes the given data flow in parallel. The interface shown in Figure 3(d) visualizes the execution of the data flow program. It features the parallel execution graph in the top part of the interface together with the color-indicated status of tasks (waiting, running, finished, failed), and information on resource consumption at the bottom of the interface.

3.3 Extensibility

A key feature of Meteor and Sofa is their extensible design. During demonstration, we show and explain how new operators can be added to the system by exemplarily adding a new operator to cope with web data (e.g., markup removal, body extraction, table extraction, etc.). Visitors can then use these operators in their queries, for example, to perform entity recognition from a set of news websites. Visitors will experience that including new operators into the optimization process requires only to specify a single subsumption relationship in Presto. They can learn how iteratively adding more operator annotations in Presto increases the number of rewrite options for this new operator.

4. CONCLUSION

We demonstrate the Meteor query language together with the optimization of MapReduce-style data flows by means of the extensible logical optimizer Sofa. In our demonstration, we present a query interface and job client visualizing all steps in the end-to-end process of query formulation, compilation, optimization, and execution, all providing diverse opportunities for the audience to interact with the system, i.e., by writing own queries, by interfering with the optimization process, and by learning about the integration of novel operators into the Meteor query language and the optimization process.

5. ACKNOWLEDGMENTS

This research was funded by the German Research Foundation under grant FOR 1036. We thank Felix Naumann, Fabian Hueske, and the Stratosphere team for valuable discussion and support.

6. REFERENCES

[1] A. Alexandrov et al. Massively Parallel Data Analysis with PACTs on Nephele. *PVLDB*, 3(2):1625–1628, 2010.

[2] D. Battré et al. Nephele/PACTs: A Programming Model and Execution Framework for Web-Scale Analytical Processing. In *SOCC 2010*, pages 119–130.

[3] K. S. Beyer et al. JAQL: A Scripting Language for Large Scale Semistructured Data Analysis. *PVLDB 2011*, 4(12):1272–1283.

[4] A. Cuzzocrea et al. Analytics over Large-Scale Multidimensional Data: The Big Data Revolution! In *DOLAP 2011*, pages 101–104.

[5] A. Heise et al. Meteor/Sopremo: An Extensible Query Language and Operator Model. In *BigData 2012*, Istanbul, Turkey.

[6] F. Hueske et al. Opening the Black Boxes in Data Flow Optimization. *PVLDB 2012*, 5(11):1256–1267.

[7] F. Hueske et al. Peeking into the Optimization of Data Flow Programs with MapReduce-style UDFs. In *ICDE 2013*, pages 1292–1295, 2013.

[8] C. Olston et al. Pig Latin: A Not-So-Foreign Language for Data Processing. In *SIGMOD 2008*, pages 1099–1110.

[9] E. Rahm and H. H. Do. Data Cleaning: Problems and Current Approaches. *IEEE Data Engeering Bulletin*, 23(4):3–13, 2000.

[10] A. Rheinländer et al. SOFA: An Extensible Logical Optimizer for UDF-heavy Dataflows. *CoRR*, abs/1311.6335, 2013.

[11] W. Shen et al. Declarative Information Extraction Using Datalog with Embedded Extraction Predicates. In *VLDB 2007*, pages 1033–1044.

[12] A. Simitsis, P. Vassiliadis, and T. Sellis. Optimizing ETL Processes in Data Warehouses. In *ICDE 2005*, pages 564–575.

[13] A. Thusoo et al. Hive: A Warehousing Solution over a Map-Reduce Framework. *PVLDB 2009*, 2(2):1626–1629.