Execution Strategies for Compute Intensive Queries in Particle Physics

Maximilian Berens
TU Dortmund University
maximilian.berens@tu-dortmund.de

ABSTRACT

Data analysis in many scientific domains, such as high energy physics, pushes the available resources of even big research collaborations to their limit, because it not only requires huge amounts of data but also compute intensive calculations. Current approaches require increased storage and computing resources, lack sufficient flexibility and are still far away from interactivity. In this paper we present ideas to cope with challenges posed by compute-intensive analyses of high-volume data in a many-user- and resource-restricted environment. Extreme selectivity of user queries (that is inherent in high energy physics analyses, for instance) permits us to reduce the computational effort to a small portion, when irrelevant data can be discarded by more efficient means. Our focus lies on providing execution strategies for analysis queries, guided by the expertise of the user and executed as a scalable, scan-based preselection (introduced in *DeLorean* [8]). Future research will encompass the development of a compact data summary that facilitates fast, columnar scan queries, as well as a domain specific language that provides us with the required information to generate them.

Keywords

Approximate Query Processing, Expensive Predicates, Big Data, Resource-Constrained Data Analysis, Domain Specific Language, High Energy Physics

1. INTRODUCTION

The ability to accumulate and use increasing amounts of data in many science domains opens enticing prospects of answering open questions of humanity. High energy physics presents itself as a prominent example of such a domain, where scientific conclusions are drawn from statistical evidence that is gained by analysing huge quantities of data. Analysis tools at this scale have to cope not only with the sheer volume of data, but also with the complexity of involved computations, the limited availabilty of resources, as well as the number and variety of user requests. The CERN corporation's Large Hadron Collider (LHC), close to Geneva, Switzerland, houses multiple experiments, each one dedicated to different questions in particle physics. One is the LHC beauty experiment (LHCb), where various aspects of the differences between matter and antimatter are studied. A common LHCb physics analysis concerns itself with a particular (type of) decay or particle. In order to observe them, protons are accelerated to very high energies and brought to collision. Over the course of a year, up to 40 million collisions are measured every seconds, prefiltered and stored. Before performing the actual analysis, specific recordings of these collisions, termed *events*, have to be selected from a global *event store*. Decays of interest are usually very *rare*. For example $\mathrm{B}_\mathrm{s}^0 \rightarrow \mu^+\mu^-$ was found only 3 times in 10 billion events [7]. A significant portion of a physics analysis consists of isolating particular types of events by carefully defining query predicates.

Detector measurements, in their initial, *raw* state, are not immediately useful and require computationally expensive and decay-specific *reconstruction* into physically meaningful entities. Filtering events for user analyses is done by enforcing predicates on these high level features.

The overall productivity of many scientific projects is limited by the amount of data that can be processed and stored. At the LHCb, this is because the measurements can not be retained at the same rate as they are taken [12]. Still, around 20 petabyte of "raw event" data goes to a tape storage every year. Also adding the long reconstruction time per event into the equation, naive on-the-fly computations over all available events for individual user requests are intractable. Offering tools to quickly query the available data without qualitative drawbacks and optimal utilization of the (limited available) resources is essential for the success of a collaboration, such as the LHCb, and impacts the pace of scientific discovery in general.

A major upgrade of the detector will start recording new data in 2021, increasing the data volume (both rate and size of events [12]) even further and signifying the necessity of new ideas.

The remainder of this paper is structured as follows. First, we give a brief overview of the concepts currently in place at the LHCb, its drawbacks and consequent, general directions of our research. Preliminary and other related work are covered in section 4. Section 5 discusses open challenges that we are going to address in the upcoming years. Finally, section 6 summarizes these ideas in a roadmap.

31st GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 11.06.2019 - 14.06.2019, Saarburg, Germany.

2. DATA PROCESSING AT THE LHCB

In order to prevent analysts to query and reconstruct the whole set of available data for every query, a centrally scheduled reconstruction-and-preselection procedure, called *stripping*, is performed. Several hundred ”stripping lines”, predefined filter queries, reduce the volume of data to an acceptable size by *materializing* their results in separate subsets.[1] A stripping line typically tries to reconstruct a certain decay and filters events in the process. The criteria to select events tend to be “vague” (in their predicates), because multiple analyses are supposed to be done on the result of a single line or small subset thereof. The stripping is initially applied during data acquisition and has to be redone later, when changes in the reconstruction software or overall user demand requires it.[2] Generally, this approach is problematic for multiple reasons:

- In addition to the raw-event data necessary for reconstruction, materialized results occupy scarce disc capacity.
- Results have to conform with user requirements, which are generally more specific/strict. This usually requires users to “restrip“ a selected subset with a customized configuration.
- Stripping line predicates are designed with respect to strict limits on available resources. This can conflict with physically motivated predicate parametrization and negatively impact the quality of conclusions, as important data might be kept out of the analysts reach. Even small changes in predicates (i.e., “loosening” of inequalities or “shifting” of ranges) are not directly implementable, because the stripping is rarely redone.
- Predefined selections are unlikely to cover unforeseen analysis use cases and thus require the definition of a completely new stripping line.

Approaches that restrict the queryable data set by predefined criteria are bound to lack flexibility, because assumptions can change and individual users have different requirements. In addition, the stripping still requires long job waiting periods and additional resources.

3. OUR OBJECTIVES

The computationally expensive reconstruction is trivially parallelizable, because events are completely independent and small in size (∼hundreds of kilobytes). However, relying on data parallelism alone does not guarantee neither general scalability nor sufficient resource utilization. New solutions need to *scale up* by leveraging available hardware features. In addition, they need to *scale out* in order to avoid bottlenecks caused by resource contention in large-scale multi-user environments.

For the purpose of pushing analytics closer towards interactivity, we identify the following goals:

- A significant reduction of the data volume, that is involved in individual user queries.
- Limit the execution of expensive computations to the result set and thus minimize the overall compute load.
- Enable efficient usage of modern hardware features (i.e, deep cache hierarchies, advanced processor instructions and multi-core architectures).
- Reuse of information by caching results and intermediate computations for upcoming queries.

A scan intensive preselection, based on a columnar, compacted representation of data entities (i.e., particle-collision events) will enable us to implement these objectives (see 4 for preliminary work). Given a small expected result cardinality of individual requests, incorporating users and their domain expertise closer into the process potentially offers significant advantages. However, precisely translating the user’s intent into selective preselection queries requires a suitable interface. In contrast to more general query languages, such as SQL, a specialized domain specific language (DSL) offers the required expressivity and easier incorporation of domain specific optimizations. Furthermore, a DSL can support efficient distributed caching strategies, as previous results might be used to answer (a potentially wide) range of upcoming queries [10]. Caching increases data locality and spreads the workload in non-uniform data access scenarios. We will further elaborate on this topic in the related works section. To this end, the development of a suitable query interface for physics analytics will be a major topic in our upcoming research.

Another important aspect of our approach will be the construction of a compacted representation or *synopsis*. Assuming that we are able to filter data just based on this representation, large portions can be discarded efficiently (via scanning) and without involving expensive computations on irrelevant data. Executing the computationally expensive (reconstruction) pipeline only on the pre-filtered, intermediate result gives the same (final) result as applying the pipeline to the whole data set directly. Of course, this requirement prohibits the synopsis-based preselection to reject any true result tuple. We provide more details on synopsis requirements in section 5.1.

4. RELATED WORKS

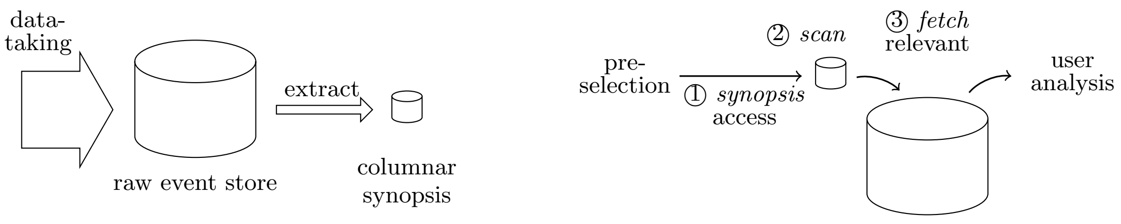
A first approach to address the problem via preselection, named *DeLorean* [8], was developed at our group and is going to be the entry point of this work. For details and a preliminary evaluation of a proof of concept, see Kußmann et al. [8]. In the following, we give a brief description.

The idea is to separate a query into a compute- and a data intensive part. In queries *commonly-used and selective* attributes (i.e., attributes that are expected to involve selective predicates) are precalculated during data acquisition and collected as columns in a single table, resulting in a much smaller representation (see fig. 1a). In order to avoid evaluations of the expensive reconstruction, a fast scan of this *compact synopsis* is supposed to discard a large number of tuples (events), reducing the (query specific) computational efforts to a sufficiently small superset of the true result (fig. 1b).

The synopsis lookup itself is efficiently expressed in relational terms, replacing reconstruction operations with scan intensive ”SQL“ operators.

[1]All lines combined drop 95% of all event data; a single line must not have more then 0.05% of the overall data volume (on average).

[2]Waiting times of several months for a new stripping version are not unusual.



(a) Preprocessing. (b) Query processing.

Figure 1: The *DeLorean* storage layer. At data acquisition time, *DeLorean* extracts a compact synopsis of the events. At analysis time, the synopsis is scanned and only a relevant subset is fetched and reconstructed for user analyses.

In a second step, surviving events are retrieved[3] and fed into the stripping software, specifically configured for individual requests, yielding the final result.

The new synopsis lookup can be implemented by modern cloud execution platforms, such as Apache Drill [1], making use of their scalability and scan-beneficial columnar storage layout [8]. Constructing a synopsis in a columnar layout provides benefits such as reduced data volume, because only relevant attributes have to be loaded and scanned. Furthermore, columns offer improved compressibility and thus also a tradeoff between processor load and bandwidth [8].

Geometric data summarization techniques in general are an essential tool in many domains, having applications in large scale maschine learning and databases, for instance. These summaries can be roughly classified into two types, *coresets* and *sketches* [9]. Similar to *DeLorean*, a summary acts as a "proxy“ for the complete data set and algorithms executed on this smaller set give an approximation of the result. However, these summaries pursue a reduction of the number of tuples, via density estimation or clustering, for instance. In contrast, *DeLorean*, as it is now, applies a *dimensionality reduction*, where less relevant attributes are simply projected out. We conjecture that both fields, dimensionality reduction as well as data summaries, can have interesting applications in our research.

A related and to *DeLorean* similarly motivated concept is that of vector approximation files (or VA files) by Weber et al. [13]. VA files partition the data space into rectangular cells and enumerate them with unique bit-strings, offering scan based preselection on a compact representation. Some cases of nearest neighbor search, for instance, can be decided on this compacted representation alone.

The *scale up vs scale out* topic is discussed by Essertel et al. [3]. The Flare project, an extension of the distributed data analysis platform Apache Spark, tries to maximize the potential of individual machines by means of native code compilation. However, the existing LHCb software stack, implemented in C++, is not reasonably migratable to another platform, given the extensiveness of the code base alone. To this end, any approach, that wants to commit to at least some practical applicability on the LHCb event retrieval problem, has to make the integration of existing software stacks possible.

Generally, this can be done by providing efficient evaluation strategies. Stepping outside of the stripping perspective and positioning the "retrieval" problem into a database context, the general schema to obtain events from the *global* (raw-) event store E can be illustrated in terms of SQL by involving a conjunction of *expensive predicates* (or "user defined functions") p_i:

$$\texttt{SELECT * FROM E WHERE } \bigwedge_i \mathtt{p}_i$$

Actual physics analyses involve various types of predicates. They are defined in terms of properties of reconstructed decays or particles, instead of plain (*raw-event*) attributes that are typically expected in database queries.

Clearly, most of the effort arises within those functions, that reach outside of the scope of relational algebra (SQL). These "black boxes" are inherently hard al with by means of "traditional" database technology.

Hellerstein and Stonebraker [5] try to correctly take into account the cost of expensive predicates when optimizing query plans via operator reordering. *User defined functions* as well as *sub-query predicates* are sometimes incorrectly assessed as "zero-time operations“ by database optimizers. Due to the simplistic relational structure of the above mentioned expression, general purpose query plan optimization techniques are unlikely to offer improvements. In contrast, our approach is going to reduce the total number of evaluations when answering user queries.

Joglekar et al. [6] try to reduce the number of explicit evaluations of expensive predicate functions. In exchange for a decrease in query accuracy, correlations of a function's result and the value of a variable can be abused to decide, if the evaluation of the predicate is required or skippable. However, the required correlation estimation is only feasible for low cardinality attributes that rarely occur in the physics context.

Declarative languages, such as SQL or XQuery, were developed to offer enhanced expressivity, enable specific optimizations and enjoy widespread usage. The importance of the declarative property of big data analytics-centric languages is supported by the works of Fegaras [4] (MRQL) and Alexandrov et al. [2] (Emma).

As SQL has roots in linear algebra (and tuple relational calculus), MRQL and Emma have *monoid homomorphisms* and *monad comprehensions* (respectively) as their formal foundation. However, these languages address a (still) very broad domain of queries and databases, omitting possible optimization potential that can not be detected in a more general context. In our work, we are going to identify query patterns that are specific to the domain of interest (LHCb event analysis, in this case). Utilizing formal systems, such as the ones used by SQL, MRQL or Emma, provides the

[3]The LHCb data format allows tree-based seeking of single raw-events via identifier [8].

associated tools and insights as well as an evironment to reason about queries and specify transformation rules for optimization.

Given that every user has different requirements and is interested in different types of data, a specialized execution strategy can optimize individual requests and thereby further improve overall performance.

Building a new and customized DSL is costly and requires knowledge both in the application domain and programming language design [11]. DSL-compiler frameworks, such as Delite [11], improve the creation of new languages by providing abstract means to integrate domain specific optimizations, implicit parallelism and (native) code generation. The insights and tools provided by this type of framework can prove useful to retrieve information from user requests, generate synopsis queries and improve the overall productivity of analysts. We suggest additional ideas in this direction in section 5.3.

In [10], the authors provide a formal definition and query processing strategies for semantic caching. Semantic caching, in contrast to page-based caching, utilizes a semantic representation of the cache content in order to find items that can totally (or partially) answer a given query. We believe that this idea can be advantageous in our setting, as queries in the LHCb context share considerable "overlap". This stems from the fact that certain (sub-)decays are "contained" in multiple decays, requiring the same computations. In fact, the concept of shared computations is already (manually) implemented in the current LHCb software stack. Detecting and abusing this overlap automatically will therefore be beneficial.

5. ADDRESSING OPEN CHALLENGES

The precise requirements on the synopsis content are yet to be defined. So far, predicates were handpicked according to their selectivity for a selected sample query. Involved attributes were included into the synopsis and all values determined by an initial stripping pass. First benchmarks are promising [8], but we need to generalize the findings to provide performance indications for a range of (unseen) queries. Also, inherent challenges of distributed many-user, high-volume data processing need to be addressed.

5.1 Creating the synopsis

To offer performance and correctness, even for new queries, some general qualities of the synopsis can be declared:

- *Applicability* - The synopsis has to contain attributes that are relevant for upcoming user query, otherwise we do not gain any advantage.
- *Correctness* - To prevent "false-negatives", no event that is actualy relevant for a user request should be rejected by the synopsis scan.
- *Selectivity* - To more-then-amortize the additional cost of a synopsis scan, a sufficiently large number of events needs to be rejected, preventing their expensive reconstruction.

Note that events, that are selected but in fact uninteresting ("false-positives"), are permissible although undesired. They are expected to be rejected by the second step and just deteriorate selectivity and therefore performance, which is less crucial then correctness.

To serve an adequate amount of user request topics, a sufficiently broad selection of synopsis attributes has to cover most frequent analyses. As a first approach, the stripping line formulations, even though being inherently "vague", involve many commonly used attributes, because the complete set is designed to cover a wide range of analysis subjects at the LHCb. A stripping line is formulated in multiple consecutive steps that define specific reconstruction procedures. Those steps also include filter statements on properties determined during this procedure. Also, there is a considerable *overlap* between the lines, as several steps are frequently shared. Many particles and even some decays are involved in multiple analyses and usually involve the same or just slightly different predicates. Currently, investigating criteria to assess attributes and their (combined) selective power for upcoming queries is important, as it represents the next logic step towards a systematic creation of an event synopsis.

A successful preselection strategy has to offer a selectivity comparable to stripping lines. Given such a selective synopsis, the overall number of events can be reduced to a manageable portion and enables the user to perform the reconstruction in a reasonable amount of time. Note that a "local restripping" is already done by analysts in practice on manually selected stripping line results in order to refine the event selection according to individual user requirements.

5.2 Result Caching

Depending on the size of the synopsis, distributing redundant copies (or only relevant columns to cover a single topic) enables the execution of the preselection independently for different work groups, possibly even single users. This way, we are able to shift the "expensive query predicate" problem further towards a data serving problem, where only actually interesting (but unprocessed) events have to be efficiently handed over to many users.

However, the selected data might be resident in different sites that are geographically dispersed (as it is the case for CERN/LHCb) and/or busy, introducing latencies. Non-uniform data access (over the data-sites) increases contention in both network and local resources. Note that this issues also arises in settings, where queries do not involve (network-)communication-intensive algorithms, such as the LHCb data analysis.

Adequate caching mechanisms can greatly improve the ability to serve data by adaptively holding frequently requested (sets of) events, greatly reducing data transfer volumes and serving latencies. Also, with knowledge about the data (-dependencies), events could be cached speculatively. For example: Decays that appear to be very similar to the desired decay are sometimes explicitly fetched to exclude them properly from the analysis.

5.3 Query Specification Interface for Physics Analyses

Developing a dedicated interface for LHCb physics analysis queries, such as a DSL, offers several benefits for this project. In addition to the general advantage of improved ease-of-use for analysts, such a language can have performance critical implications by guiding query plan optimization:

- Declarative formulation in higher-level semantics enables the user to specify his intent while relieving him from being familiar with implementation details.

- Automatic generation of preselection queries, that can be evaluated on the synopsis. This enables the user to specify queries without knowing the synopsis schema.

- Identification of new synopsis attributes by determining overlap or ”similarity“ between queries. This information could serve as a foundation to adaptively add or remove synopsis attributes, based on common query ”topics“. Performance/selectivity of upcoming query could be improved by iteratively replacing or adding information to the synopsis.

- Selectivity approximation of user requests with precalculated statistics, such as value distributions and correlations of synopsis attributes. Offering a mechanism to estimate the reduction rate of a query beforehand allows quick rejection of infeasible queries, solely based on statistics and more importantly: without starting actual computations or data transfer requests.

- Improve (opportunistic) caching. Events that only closely fail to match predicates (and other ”similarities”, such as the ones mentioned above) could become relevant and therefore proof beneficial to have in a cache, especially in an interactive test-and-see query refinement scenario. Also, result sets, that superset other results and are contained in a cache, can be used to answer particular (upcoming) queries (see *cache-answerability* [10]).

Multiple of these aspects rely on the ability to analyse the structure of queries. Automatic, non-trivial optimizations, such as filter push downs, require the system to reason about (possibly higher order) operations and their arguments, which is hard to do with application programming interfaces (APIs). Having a well-defined DSL, backed by a suitable, formal foundation, allows the definition of abstract transformation rules and eases cost-based optimization [4]. Also, answering formally motivated questions, such as cache-answerability, requires the means to infer implication- and satisfiability properties of queries [10]. Furthermore, disconnecting the interface from the execution platform offers the ability to keep the same interface, if the backend gets replaced. This is particularly useful for large projects that are expected to operate over many years.

6. ROADMAP

A synopsis that supports efficient, scan based event selection for new queries promises major performance benefits. But many open challenges exist. Currently, it is not obvious what kind of information should form the synopsis and how the choice will impact performance in a more general sense. Therefore, we first need to confirm the ideas introduced in [8] by developing means to systematically extract “frequent” synopsis attributes. Getting a better understanding on the implications of attribute choice and their impact on performance and applicability is necessary to support a sufficiently wide range of accurate queries. Also, alternatives to “carefully selected reconstruction attributes” as the synopsis’s constituents should be explored.

The idea to incorporate domain specific knowledge for optimization lends itself to interesting concepts in data processing, where general purpose techniques fail to offer significant gains. Developing new and universal schemes of applying domain knowledge for performance allows the transfer of our findings to other situations. Extracting information from user queries in order to derive a execution strategy is our first step into this direction.

After having a precise and selective mechanism in place, the “data serving” aspect of the problem will offer multiple incentives for future research.

7. ACKNOWLEDGMENTS

This work has been supported by the German Ministry of Education and Research (BMBF), project Industrial Data Science, and by Deutsche Forschungsgemeinschaft (DFG), Collaborative Research Center SFB 876, project C5.

8. REFERENCES

[1] Apache Drill - Schema-free SQL for Hadoop, NoSQL and Cloud Storage. https://drill.apache.org/. Accessed: 2019-03-03.

[2] A. Alexandrov, A. Kunft, A. Katsifodimos, F. Schüler, L. Thamsen, O. Kao, T. Herb, and V. Markl. Implicit parallelism through deep language embedding. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD ’15, pages 47–61, New York, NY, USA, 2015. ACM.

[3] G. M. Essertel, R. Y. Tahboub, J. M. Decker, K. J. Brown, K. Olukotun, and T. Rompf. Flare: Native compilation for heterogeneous workloads in apache spark. *CoRR*, abs/1703.08219, 2017.

[4] L. Fegaras. An algebra for distributed big data analytics. 2017.

[5] J. M. Hellerstein and M. Stonebraker. Predicate migration: Optimizing queries with expensive predicates. pages 267–276, 1993.

[6] M. Joglekar, H. Garcia-Molina, A. Parameswaran, and C. Re. Exploiting correlations for expensive predicate evaluation. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD ’15, pages 1183–1198, New York, NY, USA, 2015. ACM.

[7] V. Khachatryan et al. Observation of the rare $B_s^0 \to \mu^+\mu^-$ decay from the combined analysis of CMS and LHCb data. *Nature*, 522:68–72, 2015.

[8] M. Kußmann, M. Berens, U. Eitschberger, A. Kilic, T. Lindemann, F. Meier, R. Niet, M. Schellenberg, H. Stevens, J. Wishahi, B. Spaan, and J. Teubner. Delorean: A storage layer to analyze physical data at scale. In B. Mitschang, D. Nicklas, F. Leymann, H. Schöning, M. Herschel, J. Teubner, T. Härder, O. Kopp, and M. Wieland, editors, *Datenbanksysteme für Business, Technologie und Web (BTW 2017)*, pages 413–422. Gesellschaft für Informatik, Bonn, 2017.

[9] J. M. Phillips. Coresets and sketches. *CoRR*, abs/1601.00617, 2016.

[10] Q. Ren, M. H. Dunham, and V. Kumar. Semantic caching and query processing. *IEEE Trans. on Knowl. and Data Eng.*, 15(1):192–210, Jan. 2003.

[11] A. K. Sujeeth, K. J. Brown, H. Lee, T. Rompf, H. Chafi, M. Odersky, and K. Olukotun. Delite: A compiler architecture for performance-oriented embedded domain-specific languages. *ACM Trans. Embed. Comput. Syst.*, 13(4s):134:1–134:25, Apr. 2014.

[12] C. The LHCb Collaboration. Upgrade Software and Computing. Technical Report CERN-LHCC-2018-007. LHCB-TDR-017, CERN, Geneva, Mar 2018.

[13] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, VLDB '98, pages 194–205, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.