Event Index — an LHCb Event Search System

2015 J. Phys.: Conf. Ser. 664 032019

(http://iopscience.iop.org/1742-6596/664/3/032019)

Journal of Physics: Conference Series **664** (2015) 032019 doi:10.1088/1742-6596/664/3/032019

Event Index – an LHCb Event Search System

A Ustyuzhanin [1,2,6,7], A Artemov[3,5], N Kazeev [1,2,4] and A Redkin[3]

[1] LHCb collaboration, Geneva, Switzerland
[2] Yandex School of Data Analysis, Moscow, Russia
[3] Yandex Data Factory, Moscow, Russia
[4] Moscow Institute of Physics and Technology, Moscow, Russia
[5] Moscow State University, Moscow, Russia
[6] Kurchatov Institute, Moscow, Russia
[7] National Research University Higher School of Economics (HSE)

E-mail: kazeevn@yandex-team.ru

Abstract. During LHC Run 1, the LHCb experiment recorded around 10^{11} collision events. This paper describes Event Index — an event search system. Its primary function is to quickly select subsets of events from a combination of conditions, such as the estimated decay channel or number of hits in a subdetector. Event Index is essentially Apache Lucene [1] optimized for read-only indexes distributed over independent shards on independent nodes.

1. Introduction

The LHCb experiment records millions of proton collision events every second. Most of them are not needed for further analysis and are discarded by a sophisticated multi-layer trigger system [2]. What is left amounts to 10^{11} events in Run 1. Before physics analysis takes place, the number of events is further reduced by a factor of around 10. This "stripping" process takes place after the full reconstruction of the events, and produces a set of a dozen "streams" of the analysis dataset. [3]. Those streams contain candidate events for different processes — identified by "stripping lines." Events that passed the stripping process are indexed by Event Index.

Along the stripping lines some other information is indexed — global activity counters (such as total number of tracks and hits in individual subdetectors), logical file names (LFNs) on the GRID, and run conditions database tags.

2. Architecture

Event Index consists of four primary parts: backend, which hosts the indexes and processes the queries; frontend, which interacts with the user; the GRID collector for downloading events from the GRID; and an indexer for compiling the indexes. Their relationship is expressed on the figure 1.

2.1. Backend

The principle component that stores events and handles queries is a 7-node cluster. Each node hosts several shards. A shard is an Apache Lucene index. Indexes are build from .root files using MapReduce with events being evenly distributed between the nodes.

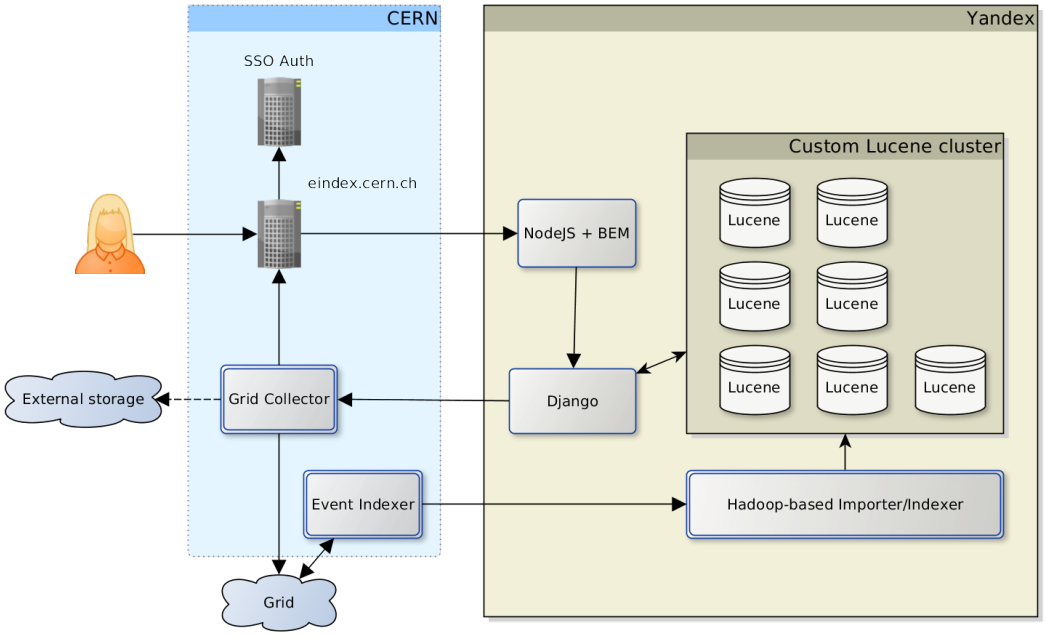

Figure 1. Event Index architecture

Events are represented in backend in a problem-agnostic generic format. Thus Event Index can be used on new datasets with minimal modification.

Event Index is optimized for read-only indexes on a static hardware configuration. Cluster expansion is still possible and can be accomplished in two ways. First, if both new data and new nodes are available, the data can be indexed on these nodes without changes to the existing structure. This approach may be suboptimal, as the best performance is only achieved when the data is evenly distributed among the nodes. Second, if only nodes are added, we must either redistribute the existing shards between nodes or reindex the dataset to include them into the cluster. Index splitting is possible but constitutes a highly experimental [4] procedure with computational costs similar to that of reindexing.

Requests are handled by a Java application as follows. Any node can become a master node by virtue of initiating a request.

- Search request: A master node receives a query, sends it to all the nodes, each in turn sends it to its shards, shards run the query, and cache the resulting bitset.
- Partial search results retrieval: A master node receives a query, asks all the nodes for the results counts, determines the nodes to send the request to. Nodes receiving the following request do the same with shards. The master node then gathers the responses and forwards them to the user.
- Field value aggregation: A master node receives a query, sends it to all the nodes, each in turn sends it to its shards, each shard aggregates the field values from the matching events. The master node aggregates the results and returns them to the user.
- Histogram calculation: A master node receives a query, sends it to all the nodes, each in turn sends it to its shards, each shard counts unique values of the requested fields, and returns them to the master node, which computes the resulting histogram.

Queries are transformed into Lucene Filters using a simple top-down parser for context-free grammar. It consists of two parts: the tokenizer and the parser itself. The tokenizer transforms

a query string into a list of tokens (=, ! =, >=, <=, (,),AND, OR, HAS) and values. The parser uses the list to build the solution tree, using prefix notation to handle parentheses and substituting HAS and AND for missing unary and binary operators. It then converts the tree into a Lucene Filter.

2.2. Performance
Indexing 10^{10} events took three days and 0.5 Tb of hard drive space per node.

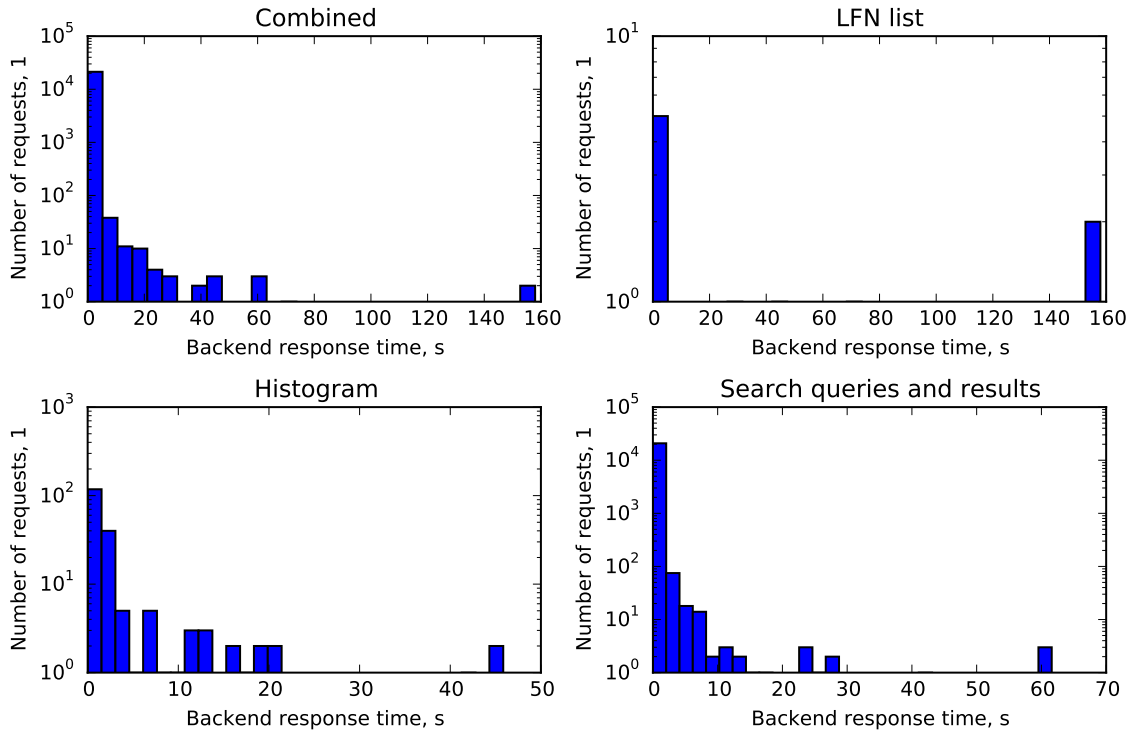

Figure 2. Backend response times for various request types. Data is taken from the live instance at https://eindex.cern.ch

The backend response times for various requests can be seen in Figure 2. This response was within 20 seconds for the majority of requests. The outliers are currently being investigated.

2.3. Frontend
All user interaction is done through the web interface, protected by CERN Single Sign-On [5]. Queries can either be typed manually or constructed with the help of an interactive wizard. Example searches:

- For a specific stripping line:
  "HAS StrippingB02D0D0KSLLBeauty2CharmLineDecision AND AND stripping=20r1"
- By file location:
  "lfn=LFN:/lhcb/LHCb/Collision11/CHARMTOBESWUM.DST/
  00022760/0002/00022760_00029252_1.CharmToBeSwum.dst AND stripping=20r1"
- Stripping line and nPVs value:
  "HAS StrippingB2D0KD2HHBeauty2CharmLineDecision AND stripping=21 AND nPVs> 4"

Event Index can compile a list of logical file names (LFN) containing the search results. If there are less than 1000 results, Event Index can fetch them from GRID as a .root file and display them in the web browser using Event Display [6]. Users can also plot histograms for the global activity counters.

2.4. The GRID collector
The GRID collector handles the .root file download requests. It resides on a dedicated server at CERN. It uses LHCb DIRAC [7] for retrieving event locations on the GRID. Then it launches parallel Gaudi[8] jobs for events retrieval and format conversion for Event Display. The source code is available on `https://gitlab.cern.ch/YSDA/grid_collector`.

3. Status
Event Index is deployed into production on `https://eindex.cern.ch/` [1]. Data from strippnigs 20, 20r1, 21, 21r1 is available.

4. Future works
We are currently studying the needs of different groups in LHCb to make Event Index a better tool. Plans include Python API, MC and turbo stream [9] indexing, and free form query processing.

5. Summary
Event Index allows selection of events and viewing of histograms of basic properties in a matter of seconds. This is much faster than the current use of GRID, which can take hours. Event Indexs core architecture will allow it to scale with data and be used for different datasets.

References
[1] https://lucene.apache.org/, Apache software foundation
[2] Head T, 2014, The LHCb trigger system, *JINST 9 C09015* doi:10.1088/1748-0221/9/09/C09015
[3] Charpentier P (on behalf ofthe LHCb Collaboration), The LHCb Computing Model and Real Data *Journal of Physics: Conference Series, vol. 331, num. 7, CHEP-2010*
[4] Lucene 5.1.0 misc API https://lucene.apache.org/core/5_1_0/misc/org/apache/lucene/index/IndexSplitter.html
[5] Ormancey E, 2007, CERN Single Sign On solution, *CHEP-2007*
[6] Langenbruch C, Couturier B, Frank M, A WebGL event display for LHCb: Status update, *5th LHCb Computing Workshop, May 18, 2015*
[7] Stagni F, et al, LHCbDirac: distributed computing in LHCb, *Journal of Physics: Conference Series, vol. 396, num. 3, CHEP-2012*
[8] Barrand G et al., GAUDI - A Software Architecture and Framework for building HEP Data Processing Applications, *Computer Physics Communications 140 (2001) 4555*
[9] Benson S, 2015, The LHCb Turbo Stream *CHEP-2015*

[1] accessible only to the members of LHCb collaboration