RAVEN - A RANDOM ACCESS, VISUALIZATION AND EXPLORATION NETWORK FOR THE ANALYSIS OF PETABYTE SIZED DATA SETS

Nikolai Gagunashvili[1], **Hans Kristjan Gudmundsson, Nicola Whitehead**
University of Akureyri, Akureyri, Iceland
nikolai@unak.is, hkg@unak.is, nicolaw@unak.is

Markward Britsch, Michael Schmelling
Max-Planck-Institute for Nuclear Physics, Heidelberg, Germany
markward@mpi-hd.mpg.de, michael.schmelling@mpi-hd.mpg.de

Helmut Neukirchen
University of Iceland, Reykjavik, Iceland
helmut@hi.is

Abstract

The analysis and visualization of the LHC data is a good example of human interaction with petabytes of inhomogeneous data. A proposal is presented, addressing both physics analysis and information technology, to develop a novel distributed analysis infrastructure which is scalable to allow real time random access to and interaction with peta-bytes of data. The proposed hardware basis is a network of intelligent "CSR"-units, which combine Computing, data Storage and Routing functionalities. At the software level the project would develop efficient protocols for broadcasting information, data distribution and information collection upon such a network, together with a middleware layer for data processing, client applications for data visualization and an interface for the management of the system.

1 Introduction

During the construction of the CERN Large Hadron Collider (LHC)[1] it was realized that the analysis of the data produced by the LHC experiments requires a computing infrastructure which goes far beyond the capabilities of a single computing center, and which since then has been built up in the framework of the Worldwide LHC Computing Grid [3, 4].

Despite the fact that many new concepts regarding data distribution and sharing of computing load have been implemented, the computing models for the analysis of the LHC data are still very close to the approach by earlier generation particle physics experiments. They focus on filtering the huge initial data sets to small samples which are relevant for particular physics question, which then are handled locally by the physicist doing the analysis (see e.g. [5].

While making efficient use of limited resources, this scheme has some obvious shortcomings.

- At a given time direct access is possible to only a small fraction of the total event sample. This reduced sample also has to serve to define and check the selection criteria for the selection jobs. As a consequence the selection may be biased or inefficient.

[1] corresponding author

- The time constant for full access to the data is given by the frequency of the selection runs which go through the complete data set. Programming errors or missed deadlines for code submission can easily result in months of delay for the affected analyses.
- High statistics measurements, i.e. analysis which use information from more than a small fraction of all events, are not feasible. The same holds for finding exceptional rare events which are not caught by selection criteria based on prior expectations.

What is needed is a framework which allows random access on petabyte-size datasets. It should have a scalable architecture which allows to go to real time information retrieval from the entire data set. The initial use case of this infrastructure will be faster and more efficient access to the data. Beyond that, however, also novel ways of interacting with the data and new ways of data visualization will evolve.

2 Requirements

In particle physics the basic units which make up a data set are so-called "events". At the LHC this is the information recorded from a single bunch crossing of the two proton beams. At LHCb [2], for example, with a typical size of 50 kB per event and 2×10^{10} events recorded per year, the annual data volume amounts to O(1) PB. The analysis of the data is conceptually simple in the sense that all events are equivalent, i.e. at the event-level it parallelizes trivially. Also the information content of a single event has a relatively simple structure, consisting of lists of instances of a few basic elements such as "tracks", "calorimeter clusters" or "vertices" which contain the measured information about the final state particles created in a high energy collision. Different events will differ in the number of those objects and the relations between them.

The basic mode of data analysis in particle physics is characterized by two steps. In the first step the data set is scrutinized for events containing a specific signature. Events with this signature then are analyzed in detail, either by extracting some characteristic information or by iterating the selection process with additional criteria.

It is evident that depending on the selection criteria the size of the event sample used in a specific analysis can vary by many orders of magnitude. On the other hand, the maximum communication bandwidth available to return information back to the user will essentially by fixed, i.e. the interaction between user and the full data set must be such that the network traffic stays below a certain limit.

The quantities of interest in a typical particle physics analysis are probabilities or probability density functions for a certain process or configuration to occur. Numerical estimates are obtained by means of histograms, i.e. simple counters for how often a certain condition is observed. The analysis framework thus must be able to handle this kind of cumulative information, which even for very large event samples reduces to a limited set of numbers.

In addition to cumulative information from many or even all events, the system must be able to transmit some or all information from a few selected events. This is of particular relevance for very rare types for final states, such as for example events with a candidate Higgs decay or other exotic processes and which require an in depth analysis of single events.

The combination of the two access modes becomes particularly relevant in the context of interactive searches for special event types starting from the full data set. Here powerful visualization

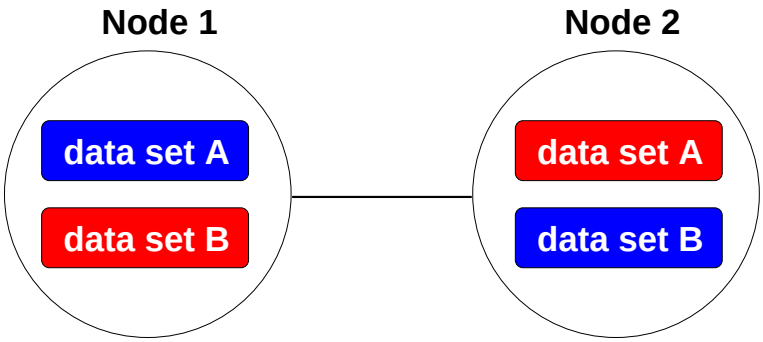

Figure 1: Simple of example of redundant data storage on two nodes. If both nodes are present, the analysis starts in parallel on different subsets. Node 1 will start on data set A, node 2 will start with B. If one node is unavailable, either because it's down or busy with another task, then the other node will process the entire data set.

tools and user interfaces are required, which provide an intuitive representation of the properties of the event set, together with the possibility of interactive select-and-zoom schemes to focus on certain candidates.

3 Design Aspects

The requirements outlined above suggest a design similar to that of a biological brain: a dense network of many "simple" nodes combining data storage, processing and the routing of information flow. For the use case of particle physics, each node would store a small fraction of the total event sample, have the possibility to run an analysis task on those events and route information back to the user having submitted the analysis query. In the following these nodes will be referred to as Computing-Storage-Routing (CSR) units, which at the hardware level are standard commodity CPUs. With an appropriate middleware-layer a network of such CSR-units will then constitute a RAVEN system.

One important aspect of RAVEN is redundant and encrypted data storage. While encryption should simply ensure the confidentiality of the data also in case that public computing resources are used, redundant storage assures that the entire data set can still be processed even if some nodes becomes unavailable. A simple sketch how duplication of data between two nodes can serve these purposes is shown in Fig. 1.

For a particular analysis or visualization task, instructions would be broadcast to all CPUs. These instructions will then be executed on the local event samples, and the information retrieved from those events routed back to the user.

As discussed before, with respect to the information that is returned one has to distinguish between cumulative data, and per-event data. Since all data have to go back to a single node, per-event data should either be of only limited volume per event or should be transmitted for only a subset of all events. Cumulative data on the other hand, such as histograms, flowing back through the network can be accumulated on-the-fly such that the total amount of information transmitted over the network stays comparatively small, even for very large event samples. Figure 2 illustrates the case.

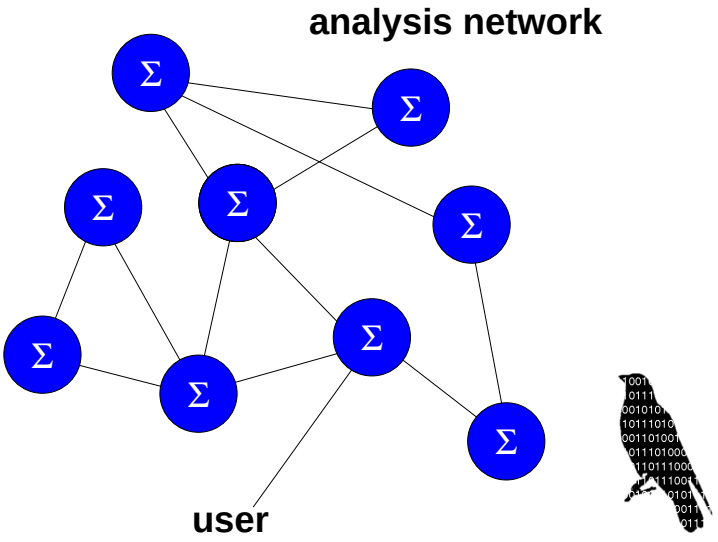

Figure 2: Sketch of a RAVEN network. Histogram data, for example, produced by the analysis jobs on the different CSR-units are routed back to the node connected to the user, and updated on-the-fly on the way back. The links show which nodes are aware of their neighbors, i.e. the network topology routing and data distribution have to deal with.

4 Implementation Aspects

A central feature of the design of a RAVEN system is its scalability, which almost automatically comes from the fact the different events are independent and thus can be spread over as many CSR-units as are available. Scalability allows to develop RAVEN on a small test system and later expand the working system to the size required for a particular application, possibly also taking advantage of cloud-computing infrastructures.

A particular implementation dealing with 1 PB of data spread over 10^4 CSR-units would correspond to 100 GB per node. Assuming a processing speed of 100 MB/s, which seems possible today, the data set could be processed within a quarter of an hour. A test system should typically have one percent of the capacity of the 1 PB system.

One problem that has to be addressed for RAVEN is the creation of ad-hoc routing and communication topologies for a given analysis query, which is used both to distribute the query to all nodes and to collect the results of the analysis. Here one challenge also arises from the fact that arbitrary next-neighbor topologies must be allowed. Furthermore, since many analyses will only access subsets of the full data set, the system should be able to process multiple queries simultaneously.

Another issue is the distribution of data, analysis code and actual query of a specific analysis. The biggest challenge is the distribution of the full data set. Here different data items have to go to different nodes, which in view of the total data volume that has to be distributed is a highly non-trivial task. The data distribution scheme also should take care of the redundant storage scheme. Finally it would be desirable that a RAVEN system automatically detects new CSR-units joining the system and migrates part of the data to the new resources.

While the distribution of the full data set will happen only rarely, updates of the analysis code will be more frequent, though still rare compared to analysis queries. The latter two can be distributed via a broadcast mechanism. The splitting into analysis code and query is motivated

by the goal to minimize the network traffic. Instead of distributing the full analysis code, which for a typical LHC experiments amounts to O(1) GB, with each query, a middleware approach is foreseen. Here the (in general machine dependent) analysis code forms a software layer on top of the operating system, which provides a machine independent high level language to perform the actual physics analysis.

While the mapping of the classical analysis models based on histograms or n-tuples on a RAVEN infrastructure is relatively straightforward, the system calls for novel approaches to exploit its real-time capabilities in new visualization tools for the interaction of a human being with petabytes of data. A simple example would be an interactive interface on a parallel-coordinates [6] presentation of multivariate event data, which allows to select outlier events.

The performance of the system can be optimized by making sure that events falling into the same class with respect to a specific selection are distributed as evenly as possible. An analysis query addressing only that subset then will harness a large number of CPU simultaneously and finish with minimal time. Providing analysis jobs with the possibility to tag events as belonging to a certain class, one could also envisage a system which is able to automatically migrate data between nodes in order to minimize access times.

Another level of optimization would be to store event related information which is created by a specific analysis for further use. Information that should be kept in a persistent store can either be specified by the user, or selected automatically, e.g. storing by default all information that is determined with computational cost above a certain threshold.

In addition to the actual functionality of a RAVEN-system a management interface is required to monitor the performance of the system like load balancing, resource usage and network traffic, and which allows to add or remove nodes from the system.

5 Prior Work

Realization of the RAVEN project will benefit greatly from already existing knowledge in networking, middleware design, distributed data storage and computing. Projects which are in principle interesting from the point of view of RAVEN are for example `BitTorrent` [7, 7] for broadcasting information over a network, the `Apache Hadoop` [9] project addressing scalable, distributed computing, the `BOINC` [10, 11] framework for volunteer computing and grid computing, or the `xrootd` [12, 13] server for low latency high bandwidth data access in the `root` [14, 15] framework, which defines the de-facto standard for data analysis in particle physics. Additional input could come from the Grid-middleware developers e.g. `gLite` [16, 17] or the `Linux` community [18].

6 Summary

A proposal has been presented which targets the problem of high performance parallel analysis of the LHC data. The architecture of such a RAVEN system ensures full scalability with random access to the entire data set. Further key features of the system are redundant storage, on-the-fly accumulation of results and a rigorous middleware-approach to the data analysis plus the development of new tools for visualization and interaction with massive data set.

Acknowledgments

The authors would like to thank Torsten Antoni (GridKa), Stefan Heinzel (RZG), Andreas Heiss (GridKa), John Kennedy (RZG), Stefan Kluth (MPP), Wolfgang Müller (HITS) and Andreas Reuter (HITS) for useful discussions and constructive criticism concerning the RAVEN proposal.

References

[1] Evans L. and Bryant P. (editors): LHC Machine. JINST 3 (2008) S08001.

[2] The LHCb Collaboration, A. Augusto Alves Jt. et al.: The LHCb Detector at the LHC. JINST 3 (2008) S08005.

[3] The LHC Computing Grid: LCG Website, `http://lcg.web.cern.ch/lcg/`.

[4] Eck, C., et al.: LHC computing Grid: Technical Design Report. Version 1.06. LCG-TDR-001 CERN-LHCC-2005-024.

[5] The LHCb Collaboration, Antunes Nobrega R. et al.: LHCb Computing Technical Design Report. CERN/LHCC 2005-019.

[6] Inselberg, A.: The Plane with Parallel Coordinates. The Visual Computer **1** (1985) 69-91.

[7] BitTorrent, Inc.: BitTorrent Website, `http://www.bittorrent.com`.

[8] Cohen, B.: Incentives Build Robustness in BitTorrent. 1st Workshop on Economics of Peer-to-Peer Systems, University of California, Berkeley, CA, USA (2003).

[9] The Apache Software Foundation: Apache Hadoop Website `http://hadoop.apache.org/`.

[10] BOINC project: BOINC website, `http://boinc.berkeley.edu/`.

[11] Anderson, D. P.: BOINC: a system for public-resource computing and storage. Fifth IEEE/ACM International Workshop on Grid Computing 2004.

[12] XRootD project: Scalla/XRootD Website, `http://project-arda-dev.web.cern.ch/project-arda-dev/xrootd/site`.

[13] Hanushevsky A., Dorigo A. and Furano, F.: The Next Generation Root File Server. Proceedings of Computing in High Energy Physics (CHEP) 2004, Interlaken, Switzerland, 2004.

[14] The ROOT team: ROOT Website, `http://root.cern.ch/`.

[15] Antcheva I., et al.: ROOT – A C++ framework for petabyte data storage, statistical analysis and visualization. Computer Physics Communications **180** (2009) 2499–2512.

[16] gLite Open Collaboration: gLite Website, `http://glite.web.cern.ch`.

[17] Laure E., et al.: Programming the Grid with gLite. Computational Methods in Science and Technology **12** (2006) 33–45.

[18] The Linux Foundation Website, `http://www.linuxfoundation.org`.