Hadoop and friends - first experience at CERN with a new platform for high throughput analysis steps

D Duellmann, K Surdy, L Menichetti and R Toebbicke
CERN, 1211 Geneva 23, Switzerland.
E-mail: dirk.duellmann@cern.ch, kacper.surdy@cern.ch, luca.menichetti@cern.ch, rainer.toebbicke@cern.ch

Abstract. The statistical analysis of infrastructure metrics comes with several specific challenges, including the fairly large volume of unstructured metrics from a large set of independent data sources. Hadoop and Spark provide an ideal environment in particular for the first steps of skimming rapidly through hundreds of TB of low relevance data to find and extract the much smaller data volume that is relevant for statistical analysis and modelling. This presentation will describe the new Hadoop service at CERN and the use of several of its components for high throughput data aggregation and ad-hoc pattern searches. We will describe the hardware setup used, the service structure with a small set of decoupled clusters and the first experience with co-hosting different applications and performing software upgrades. We will further detail the common infrastructure used for data extraction and preparation from continuous monitoring and database input sources.

1. Introduction
The quantitative analysis of computing infrastructure metrics is recently receiving increasing attention at CERN and other High Energy Physics sites. In an environment of constant budget and at the same time diminishing yearly improvements from Moore’s and Kryder’s laws the science community is forced to follow all workflow optimisation options with similar rigor (and often also similar methods) as applied to physics analysis.

The IT department has therefore setup a working group with the goal to understand the science workflows involving the CERN computing center (eg file transfers in the LHC grid, accuracy of CPU benchmarks) on a more quantitative level.

The scope of this working group includes the medium to long term metric analysis using statistical and machine learning methods aiming to go beyond isolated time series analysis performed in traditional monitoring systems. More details about this analysis can be found in [8]. The working group consists of experts from most significant IT services and works in close collaborations with experiment computing experts. Some of the development reported in this contribution have been performed in collaboration with infrastructure experts from the Bhabha Research Centre (BARC, Mumbai).

The metrics taken into account originate from many different subsystems in the CERN computing centre. Table 1 gives some overview of the data sources and volumes.
Table 1. Example metric sources and volumes

| Metric Source | Volume | Description                      |
|---------------|--------|----------------------------------|
| lemon         | 78 TB  | box level monitoring metrics.    |
| squid         | 110 GB | http cache access metrics        |
| openstack     | 12 TB  | agile infrastructure             |
| syslog        | 23 TB  | unstructured box logs            |
| eos           | 12 TB  | file access metrics              |
| castor        | 55 TB  | tape archive access              |
| LANdb         | O(100) MB | hostname, ip, hypervisor, location |
| perfsonar     | O(10) GB | network link status              |
| exp. dashboards | 1 TB  | experiment job summaries         |
| exp. file popularity | 200 GB | experiment user data access      |
| batch         | 500 GB | batch accounting and queue config |
| hardware specs | 100 MB | hardware details including CPU and disk ratings |

Despite the relatively modest total data amount one is confronted with a number of significant analysis challenges due to the often poor data design and data quality of the input data. The system described here attempts to offload the involved analysts from routine tasks like data collection, filtering and reformatting, so that the limited human effort that is available for data analysis can be applied effectively.

2. Components
A variety of components has been developed by the open source community to interact with Hadoop ecosystem. The Hadoop service at CERN provides a selection of them. The selection has been made to satisfy most of the usage scenarios and users' requests while keeping the maintenance efforts in balance. The core components of the Hadoop service are HDFS, YARN, MapReduce, Pig, Spark, HBase, Sqoop.

2.1. Hadoop architecture
This section describes the roles and typical usage of the components.

**HDFS** is the main, and often only, file system used by the applications. It provides the base distributed storage layer with necessary data replication and controls access right to the files and directories.

**YARN** manages computing resources, i.e. CPUs and memory, on the clusters and schedules jobs. More about resource management can be found in the later part of this contribution.

**MapReduce** is a framework where data processing applications are defined as a sequence of map and reduce tasks. It uses YARN to allocate resources for task executions.

**Pig** is a tool that generates a chain of MapReduce tasks based on operations defined in a high level scripting language.

**Spark** is an increasingly popular data analysis engine used on our clusters. It offers interfaces for multiple programming languages and can be used for interactive applications, batch data processing, and streaming jobs. In our configuration a Spark application submits jobs to the cluster resource manager (YARN) and usually operates on data stored on HDFS. More about Spark APIs and possible integration with Notebooks can be found below.

**HBase** is a key-value database that runs on top of HDFS. It is used when low access-latency is preferred over high throughput. Its daemons run on cluster nodes independently from YARN.
While it is an important part of the service and can be accessed with Spark, it is less commonly used in the current data analysis workflows. **Sqoop** is a tool used for transferring data from relational databases to Hadoop. Typically, Sqoop jobs are scheduled to perform a data import.

2.2. **Ensuring service level**
Currently the Hadoop service is regarded as “production” nevertheless not highly critical. Short, scheduled, downtimes are considered to be acceptable by the users. The redundancy is ensured at the datanode level, any failure of up to two machines does not affect the service. The namenodes remain the potential single points of failures. A total loss of a namenode would mean a temporary downtime of the service and a loss of the changes made to the file system over the last tens of minutes. Work is currently being done to improve the service by moving to a High Availability setup, which allows continuous operation even in case of namenode failure.

2.3. **Versioning and provisioning**
Cloudera Hadoop Distribution (CDH) is used on all the clusters and all component versions are kept synchronised with the CDH release, however different clusters may have different CDH versions deployed. All configuration management is done using Puppet. With no High Availability support in place, software updates are disruptive, i.e. cause downtime, and need to be scheduled. A shift to less intrusive rolling updates is planned for the near future.

3. **Infrastructure setup**
The CERN Hadoop service offers several Hadoop clusters for a variety of applications. Separating applications across multiple clusters was mainly motivated by the desire to avoid potential interference between users and is required to ensure appropriate isolation level of different applications.

3.1. **Hardware setup**
Every cluster is equipped with two nodes dedicated to perform the role of namenode, HBase master and YARN resource manager. The other (15–40) data nodes feature reasonable amount of main memory (64GB), typically 32 CPU cores and a storage capacity of several tens of terabytes. This equipment was not specifically ordered for the Hadoop service, but shares its specifications with the hardware used for other storage services. The disc drives are used in native mode, i.e. without RAID, erase coding or similar techniques, relying solely on HDFS for load management and resilience.

4. **Resource management**
The service is offered to multiple users who do not necessarily work together in the same team therefore ways of coordinating activities are needed. The main mean for limiting application resource usage within a cluster is by setting up disk usage quota and prioritising resources allocations in YARN. The separation in multiple cluster allows for additional isolation between groups of users where those methods are not effective.

4.1. **Isolation with separate clusters**
There are two major reasons for separating groups of users in different clusters. The first one is to satisfy conflicting software update requirements. Users developing new applications are usually interested in having access to the newest software versions, while those running stable production applications prefer avoiding interruptions and changes. The other aspect relates to the shared resources that are not under the control of the resource manager, namely the load
put on the namenode. An application which needs a high rate of metadata operations on the file system stresses the namenode and may affect the whole system.

4.2. Dynamic resource pooling

**YARN.** Fair Scheduler with preemption is used to manage computing resources on the cluster. Dedicated queues are used for different types of applications. Critical applications have designated high priority queues with guaranteed resources while other applications compete for the remaining resources. Any job, even with low priority, can potentially use all the available resources. Within the same queue YARN will try to assure fair resource allocations among the jobs. Between different queues preemption policy will guarantee resources for those with higher priority.

**HDFS.** Quota on the space usage and the file count are set for individual users and for applications. Most users keep their initial conservative quotas and request larger ones on a per application basis. The need for quotas on space usage is a direct consequence of storage capacity limitations. Quota on file counts are required to limit the memory consumption of the HDFS daemons.

4.3. Monitoring

Monitoring of the cluster relies on a few separate systems. Hardware and operating system statuses and, in the event of problems, alerts are sent to Lemon [5]. Ganglia is used for graphical resource usage representation, including CPU and memory utilisation and other performance metrics. Additional information about a particular job can be obtained using the monitoring pages provided by Hadoop. Additionally, probes checking high level functionalities run periodically to send alerts and to update the CERN Service Status Board.

5. Data Preprocessing

Hadoop users may run batch or streaming jobs that are producing or collecting data from external sources. Also programs from outside the cluster may use the Hadoop API to generate additional data. The main data producer in the Hadoop clusters is the CERN IT Monitoring team [1] that is adopting a combination of these techniques but mainly write data using agents from external nodes. After the ingestion, data are preprocessed and reorganized before being served to analysts.

5.1. Data ingestion

In HDFS, independently from the technique used, data are written in chunks at periodic time intervals. In case of streaming data, most of the clients are accumulating events before flushing them into HDFS. Currently the biggest data producers are Flume clients; they write small files into temporary folders. It is important to change temporary destination folder (tmp) for each Flume collector once per day at least, to consolidate what has been accumulated during the day in a reduced number of files with a size that fits an optimal number of blocks per file (usually 3 or 4 for uncompressed textual file format, around 1 GB size). Therefore, during analysis, the programmer has to treat the tmp and already aggregated folders differently, so that the two can be combined to process up-to-date data.

Currently 6 PB of storage are available across all clusters with nearly 70% of occupied HDFS space. The growth rate is estimated to be stable below 4 TB per day. How to reduce this size and to sort data properly is still under development together with the monitoring team. The main idea is to adopt different approaches in the way files are written and aggregated, in the first place employing Spark (instead of Pig to run the tmp folder aggregation) and considering different formats and compressions. At the moment the majority of data are written as they
come from data sources through the transport layer. The monitoring team has adopted JSON for convenience and as it is still the most used format for serialization in network communication.

5.2. Data reorganisation
The reason why JSON is still preponderantly present is because it doesn’t put any constraint on the schema definition: a JSON file (Hadoop style, one JSON per row) can be parsed even if its lines have totally different schemas. This gives the possibility to implement all data collectors with the same logic and relieve data producers of the obligation to define a data structure for each source stored. As a consequence raw data present in HDFS may look like sylogs, rather unstructured or with metrics embedded in strings as nested objects. Before stepping into the analysis phase, a lengthy sequence of additional steps is necessary to skim data: removing corrupted lines, parsing or converting values, correcting malformed or dropping unnecessary columns, merging multiple attributes into one value, filtering lines based on time or metric types. Unfortunately these steps cannot be automatised for all datasets and may vary even for one dataset in time. After a second phase where data can be further filtered and enriched using external information (e.g. IP address resolution, geo-location), processed datasets can be finally written into HDFS or used to produce further aggregations and summaries. In any case, once the desired result is obtained, it is better to store it instead of repeating the time consuming procedure on raw data. The resulting dataset will usually be stored in daily folders, regardless of the original raw structure. Having daily instead of monthly folders allows to run lighter conversion jobs, to have fine grain backups and to process only days of interest. For each subsystem we provide in HDFS one or more datasets with different shape serving different analysis purposes. Possible formats are: CSV and JSON (textual formats, general purpose, handy to be exported), Parquet (columnar based format, useful to filter lines efficiently, high compression), Avro (serialization standard, binary encoded, fast full scans).

6. Analysis
Information available, stored in HDFS, are beyond time series data and belong to different computing sub-systems across the CERN infrastructure or within the WLCG [6]. These don’t use Hadoop merely as long term storage: although many of them may already have solutions to run analysis and monitoring (ElasticSearch, Oracle), there are situations where they require to complement their tasks using Hadoop. As platform, Hadoop supports more flexible analytics, enabling the possibility to join and combine data from different sources and process big data volumes. Infrastructure analytics is a complex activity and may include the evaluation of several computing sub-systems at once, depending on what component is under investigation. This is one of the main reasons to motivate the usage of Hadoop as common base repository: to cross-correlate and combine information between projects, enabling the possibility to quantitatively understand the computing involving the CERN infrastructures. The goal of the analytics platform is to reduce the time between data acquisition and analysis, keeping the focus on understanding the data instead of spending time with data manipulation.

Considering heterogeneous datasets adds complexity during analysis. The goal is to join information coming from different sources in a defined time window using certain criteria. Having data partitioned in HDFS as daily folders helps to avoid reading unnecessary information. The file format may have significant performance impact, depending on the detailed row and column selectivity of an analysis problem. Hence we provide several alternative format choices. Finally, after the join, a smaller subset is ready to be analysed. This kind of operation is often carried out using Spark.
6.1. Spark
The lack of constraints regarding data formats and schemas, may lead to disruptive situations for analysts that have to learn beforehand how to deal with this degree of freedom, ending up studying how to use a variety of new technologies and formats. A way to bypass this issue is to adopt Spark from early analysis stages. In addition to performance considerations that will favour Spark compared to plain Hadoop MapReduce [12], Spark has lately expanded its audience in the landscape of parallel data analysis tools due to its simplicity, allowing users to write complex operations using a high-level API. It binds most popular analysis languages: Python, R, Scala, Java and offers an unifying stack where the general purpose core [9] contains multiple components specialized in different workloads (i.e. SparkSQL core, machine learning and streaming libraries). All primitives can hence be expanded with other functions provided by these components, giving the user full control over the parallel execution process and many alternatives to enrich the final result. What makes Spark different, compared to traditional MapReduce jobs, is its execution plan that during the computation tends to cache data in memory and to reuse them as much as possible before accessing the disk for another IO operation. The typical use case for Spark is to have a job that starting from unstructured data creates a more compact dataset with defined schema, format and compression (a Spark program that converts JSON into Parquet contains 3 lines). Simple Pig jobs running data aggregations can be reimplemented in a more efficient Spark job, which is less resource demanding. In the last year, the ratio of Spark jobs in the Hadoop Service has increased reaching 35% of total applications and is expected to outgrow soon the ratio of PIG jobs.

6.2. Web notebooks
Popular among data scientists, web notebooks are web applications that allow users to execute code and commands within a web page. The output displayed can be a text or a plot, making it a useful tool for data inspection and visualization. The most popular is Jupyter [11], which runs a Python kernel. PySpark (the python version of Spark) can easily replace that kernel powering the notebook with a job running in the cluster. Another web notebook is Apache Zeppelin [4], an open source product developed in Scala, Java and Javascript natively supporting Spark. At CERN, a project called SWAN [13] is already providing web notebooks as a service; although they don’t support yet PySpark notebooks running in cluster mode, a collaboration between the two teams (Hadoop and SWAN admins) is already in progress to make this feature available in the near future. Currently users asking for Jupyter or Zeppelin (with Spark running in cluster mode) have to create a machine in Openstack running ad-hoc Puppet modules created for this purpose. To run a Spark job it is necessary to create a JAR or a script and submit it using a special command, while web notebooks simplify this process and can be a really powerful platform for learning and testing. Additionally, for every instruction issued in a notebook the output will persist in the web page; this is convenient to store and carry around results of a computation without a running application.

6.3. Examples
Lemon data. In collaboration with BARC, we have setup an analytics repository with combined metrics from different subsystems (collected by Lemon). Spark is used to implement a periodic processing pipeline that extracts relevant metrics and desired statistical aggregates from Lemon raw data, stored as nested JSON objects. Each of them represent a metric created by a sensor process running in a machine or VM. Depending on the kind of metric, each message may have different frequency and size; however in HDFS all metrics belonging to the same subsystem are stored together in monthly folder as big text files. Role of the Spark job is to sift among all messages only those of interest and perform an aggregation. Currently around 100 TB of 1000 kind of metrics are stored. Filtering out those not relevant for analysis is the first and
the most time demanding step. A large fraction of metrics is designed to record operations and to help during issue tracking. Some metrics are useful for time series visualization but less so for quantitative analysis. Before the aggregation it is necessary to correct accuracy errors, wrong units or corrupted measurements. Later, once the aggregation has run by cluster or group of VMs, statistically relevant summaries are generated (e.g. hourly statistics, standard deviation, min/max). Data is stored in JSON back into HDFS and ready for further analysis. Currently Spark jobs are running weekly generating statistics for 20 “cloud” project metrics and CERN’s batch infrastructure. Examples are host statistics, average CPU load metrics (one of the most frequent). The processing requires 16 minutes with 50 executors (1 GB of memory each) for a monthly input volume of 1.3 TB.

**EOS logs.** Logs containing file transactions, read and write operations, from EOS archive system. Processing and analysing these data is fundamental to evaluate the use of the EOS storage service. A monthly folder can reach 1 TB size and it is challenging to select lines based on few criteria (one day or one machine) since no schema is provided inside the JSON transport layer (a string with custom key-value format): the entire log line has to be parsed and evaluated. This task is particular inefficient because it is necessary to go through all lines when selecting just few of them. Another issue is that in some cases we find duplicated messages. To overcome these problems, a Spark job is running daily to convert the monthly folder of raw data into daily gzip compressed Parquet folders. An example, for Alice EOS 2016-08 raw logs size is 1.4 TB, meanwhile the Parquet version amounts to 93.1 GB. This 90% reduction of size can be explained: discarding fields that are not of interest (10 out of 44, on average accounting for 3% of total log length), duplicates deletion (around 25% of total records) and the rest is Parquet RLE within columns and GZ compression.

**PhEDEx transfer.** PhEDEx, the CMS transfer system uses a file catalogue. Data are stored in Oracle and imported daily to HDFS using Sqoop, storing catalog incremental updates and block replica table snapshots. Hence calculation of long statistics and aggregations are offloaded into Hadoop, avoiding to run long procedures in Oracle. Beside, keeping the snapshots for long term allows to reconstruct the transfers history even after this information has been removed from Oracle. A similar popularity analysis is performed on CMS XRootD data [10].

**Network Connection Study.** This study had the goal to optimise the utilisation of IPv4 networks at CERN and involved to calculate the maximum concurrent connection count on all subnets. The input was half a year of connection records stored in a few GB Oracle table. An Oracle SQL query, even optimized to run in parallel, took some 14 hours to complete. An easy way to improve the performance was to move data and run the same query in Hadoop. This was achieved by using Sqoop to export the table and Spark to run the SQL query. The final execution took only 15 minutes [7].

7. Conclusion and future development

In summary, the CERN analytics working group has in collaboration with colleagues from BARC, Mumbai implemented an analysis repository for computer center infrastructure metrics that allows to combine information from all major computing subsystems aggregated and stored by the CERN IT Monitoring team. Hadoop and Spark have been chosen to implement a scalable analysis pipeline, which extracts relevant raw metrics from a unstructured data lake and derives relevant statistical aggregates for further analysis. The analysis input data is provided in a variety of data formats, including the CPU and storage efficient “parquet” and “avro” formats. The resulting systems complements the parallel short term analysis pipeline based on Kibana/ElasticSearch and allows to extend analysis scalability and flexibility, available visualisation and processing options and last but not least data retention.
Aknowledgment
The authors would like to thank Bangalore Jagadeesh, Vineet Menon, Dheeraj Gupta and Ajay Kumar from the Bhabha Atomic Research Centre for their active contribution defining, testing and optimising the Spark based Lemon data aggregation.

References
[1] Aimar A, Karavakis E, Andrade P, Magnoni L. Unified monitoring architecture for it and grid services. CHEP, 2016.
[2] Apache Hadoop. http://hadoop.apache.org.
[3] Apache Spark. http://spark.apache.org.
[4] Apache Zeppelin. http://zeppelin.apache.org.
[5] Babik M, Fedorko I, Hook N, Lansdale T H, Lenkes D, Siket M, Waldron D. Lemon - lhc era monitoring for large-scale infrastructures. *Journal of Physics: Conference Series*, 331(5):052025, 2011.
[6] Bird I, Buncic P, Carminati F, Cattaneo M, Clarke P, Fisk I, Girone M, Harvey J, Kersevan B, Mato P, Mount R, Panzer-Steindel B. Update of the computing models of the wlcg and the lhc experiments. Technical Report CERN-LHCC-2014-014. LCG-TDR-002, Apr 2014.
[7] Canali L, Kothuri P, Baranowski Z. Integration of oracle and hadoop: hybrid databases affordable at scale. CHEP, 2016.
[8] Duellmann D, Nieke C. First results from a combined analysis of cern computing infrastructure metrics. CHEP, 2016.
[9] Karau H, Konwinski A, Wendell P, Zaharia M. *Learning Spark: Lightning-Fast Big Data Analysis*. O’Reilly Media, 2015.
[10] Meoni M, Boccali T, Magini N, Menichetti L, Giordano D. Xrootd popularity on hadoop clusters. CHEP, 2016.
[11] Project Jupyter. http://jupyter.org/.
[12] Shi J, Qiu Y, Minhas U, Jiao L, Wang C, Reinwald B, Özcan F. Clash of the Titans: MapReduce vs. Spark for Large Scale Data Analytics. *Proc. VLDB Endow.*, 8(13):2110–2121, September 2015.
[13] Saavedra E T. Swan: a service for web-based data analysis in the cloud. *CHEP*, 2016.