Performance optimisations for distributed analysis in ALICE

L Betev[1], A Gheata[1], M Gheata[2], C Grigoras[1] and P Hristov[1]
[1] European Organization for Nuclear Research (CERN) - Geneva, Switzerland
[2] Institute for Space Sciences – Bucharest, Romania

E-mail: andrei.gheata@cern.ch

Abstract. Performance is a critical issue in a production system accommodating hundreds of analysis users. Compared to a local session, distributed analysis is exposed to services and network latencies, remote data access and heterogeneous computing infrastructure, creating a more complex performance and efficiency optimization matrix. During the last 2 years, ALICE analysis shifted from a fast development phase to the more mature and stable code. At the same time, the frameworks and tools for deployment, monitoring and management of large productions have evolved considerably too. The ALICE Grid production system is currently used by a fair share of organized and individual user analysis, consuming up to 30% or the available resources and ranging from fully I/O-bound analysis code to CPU intensive correlations or resonances studies. While the intrinsic analysis performance is unlikely to improve by a large factor during the LHC long shutdown (LS1), the overall efficiency of the system has still to be improved by an important factor to satisfy the analysis needs. We have instrumented all analysis jobs with "sensors" collecting comprehensive monitoring information on the job running conditions and performance in order to identify bottlenecks in the data processing flow. This data are collected by the *MonALISa*-based ALICE Grid monitoring system and are used to steer and improve the job submission and management policy, to identify operational problems in real time and to perform automatic corrective actions. In parallel with an upgrade of our production system we are aiming for low level improvements related to data format, data management and merging of results to allow for a better performing ALICE analysis.

1. Introduction

The ALICE analysis workflow requires processing of large data sets with complex event structure in a distributed environment. This justifies sharing the input data between analysis tasks using a ‘read once - process many times’ policy. Such approach requires de-serializing the full event information to satisfy the large diversity of analysis needs, putting a non-negligible burden on the I/O. From a data processing perspective, the individual tasks are optimized for data locality in a quasi-democratic distributed system. This implies that jobs are being scheduled such that most pull data via LAN, not excluding however a small share of remote files accessed using *xrootd* [1]. The remote reading can pose a serious limitation for the less CPU-intensive analyses, which can become I/O bound. The behaviour becomes pathological in case of individual user analysis, leading to a lower overall ALICE analysis jobs efficiency. This paper describes the recent developments carried out in the core offline team to optimize the ALICE distributed analysis and identify and redress as much as possible the sources of inefficiency. The reason for this work is triggered by today’s analysis needs in ALICE, projected to year 2018 through the requirements of the Run3 of LHC.

We have attempted as a first step to quantify the impact of the various sources of inefficiency to our existing data flow. This analysis is quite general and applicable for other distributed analysis systems as it allows understanding what elements can be optimized and to which extent. As a second step we have instrumented the analysis jobs with real-time monitoring, allowing identifying throughput limitations depending on resource configuration and its state, for example storage server load, site configuration or analysis job type. Correlating this information with expected values collected in a test facility allows taking corrective actions, described in the following chapters. The "hunt" for efficiency in analysis reveals both strong and weak features in our analysis model and gives a general approach for analysis optimization in a distributed environment.

2. Figures from today and goals for tomorrow

During the LHC *Run1*, ALICE was producing in average about *4 PBytes* of data suitable for analysis per month. In the same month, the data was processed about five times using about one third of the total available ALICE Grid [2] resources. During Run3, the RAW data reconstruction will be done on a dedicated online facility - the O^2 [3]. By assuming a constant Grid capacity growth of about 20% per year and its fair share between analysis and simulation we estimate an increase in analysis power by a factor of 5 by 2018. On the other hand, during Run3, the event rate is expected to increase by a factor of about 50, which are several factors above the projected processing over acquisition ratio. This problem is applicable for all LHC experiments and calls for detailed reviews of the analysis workflows to improve the overall efficiency.

To collect the necessary data for the efficiency studies, we decided to implement "sensors" into the analysis framework, which send information to a centralized monitoring system. The first objective of the analysis of the collected monitoring data was to identify bottlenecks in the workflow, allowing implementing different methods of data access and throughput optimization. This implies reducing the time to solution and at the same time increasing the efficiency by reducing the overheads in different stages of the distributed analysis workflow. In addition a simplification of the event structure and making low-level optimizations like caching or vectorisation is necessary.

ALICE uses from the beginning of the data taking a common analysis framework, described in detail in previous papers [4,5]. This brings many advantages to users and allows for more efficient organized analysis. Some of these advantages are:

- Having common interfaces hiding data diversity and backend complexity;
- Freedom to run user analysis on all resources in a transparent way;
- Uniform analysis model and style covering the full workflow;
- Powerful job control and traceability;
- Sharing input data for many analysis in a train-like fashion, with an increased quota for organized analysis

A downside of the approach was to have quite "generous" event information in the general analysis data formats to accommodate any type of analysis, leading to a bigger event size and reducing the analysis throughput and efficiency. The organized "train" model [5] also implies a larger impact of individual errors on the turnaround time and overall efficiency in distributed analysis. We also observed bad usage patterns of the framework in user analysis, which trigger in many cases inefficient use of resources.

3. Low level view on analysis efficiency

Trying to quantify efficiency in all phases of an analysis job, we end-up with a simple classification shown in figure 1. The very first stage is reading the event data from either a network stream or the local disk. The time spent for this I/O operation is proportional to the event size and with one over the streaming throughput, having as overhead the rate of I/O operations performed times the latency (or round trip time). While a fast network or a local SSD can help fetching the data faster into memory, it is very important to minimize the rate of I/O requests, especially for WAN streams. For ALICE analysis, the data streaming is steered by the ROOT framework [6] and the way to achieve this is to activate tree caching. This will be discussed based on a simple example in this section. In figure 1, the reading and writing repeats after each event to fetch the next event and possibly to write some tree output.

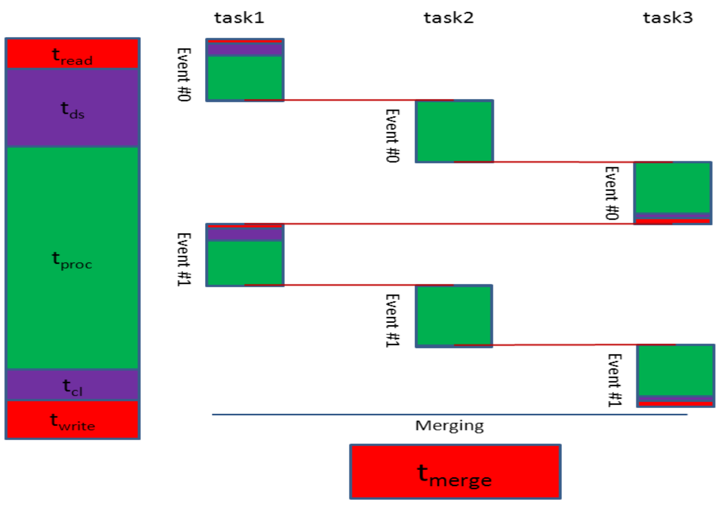


Figure 1. Processing phases for ALICE analysis trains. The total time spent per event is shared by I/O (t_{read}+t_{write}), de-serialization of the buffers into memory objects (t_{ds}), processing (t_{proc}) and event clean-up (t_{cl}). These phases repeat sequentially for each processed event in case asynchronous tree prefetching is not activated in ROOT. The merging time for the analysis outputs of different jobs has to be accounted in addition, as well as the specific distributed processing overheads.

In a second stage, the event is de-serialized by ROOT from memory buffers. While this operation is accounted as CPU time by the OS, it represents in fact hidden I/O and has to be considered as such in the overall budget. Again, this time depends on the event size, but also on its complexity (i.e. number of branches that have to be de-serialized). In case of ALICE AOD format, we have recently measured that simply flattening the event structure into a single level *ntuple* while keeping the same information as in the original data structure (which uses composition in an object-oriented approach) reduces by a factor of two the deserialization time and highly improves the *ROOT* file compression factor. Table 1 presents some simple measurements for an empty analysis train, which is just reading all events in a file in different conditions. This illustrates the throughput limits to which the analysis can become I/O bound due to either reading latency or deserialization.

Table 1. Throughput limits when reading ALICE AOD data in different conditions. While in the local case the limit comes from de-serializing data, for WAN reading without caching the I/O requests is typically bound by the round trip time. A heavy load on the disk servers can additionally worsen the situation. The CPU cycles contributing to the apparent efficiency are just used for deserialization and put an upper limit to the achievable throughput.

Device throughput and access time		Job throughput	Apparent job efficiency
Local spinning:	50 MB/s, 13 ms	5.93 MB/s	86.5%
Local SSD:	266 MB/s, 0.2 ms	6.83 MB/s	94.1%
WAN:	7.4 MB/s, 63 ms RTT		
No load on the remote disk server		0.46 MB/s	13.4%
Heavy load on the disk server		0.083MB/s	2.5%

While reading the ALICE AOD from the local disk, the tests reveal an I/O bound behaviour. Most of the job time in this case is taken by de-serialisation, which limits the throughput to a maximum of about 7 MB/sec for the test machine. To be noted that this limit depends linearly on the CPU speed of the machine, fact observed directly on the I/O bound jobs being run at different computing centres.

The third phase of processing is the actual task execution, which needs to be maximized with respect to the I/O. This can be done in ALICE analysis by chaining many analyses in a single train. While this can achieve the goal of reading once and processing many times, it introduces extra memory needs and pushes the time to analysis completion. A compromise is reached by combining I/O and CPU intensive tasks in a train, thus optimizing the overall CPU/IO performance.
Some additional analysis overheads consist in writing the output data and cleaning-up of the event transient data structures (like list of tracks) in between reading events. In the ALICE case this represents less than 1% of the total job time and will not be discussed here. What represents a real bottleneck in getting the final results for distributed analysis is the merging of all partial results produced due to the splitting of the input data to several sub-jobs. The current solution adopted in ALICE analysis and possible further improvements of this algorithm will be discussed in the following section.

We haven't discussed the impact of the job and data management on the distributed analysis performance. This does not affect the job efficiency, but can increase significantly the time to completion. Different types of overheads can be triggered by unevenness of the sub-job completion time, delays in the jobs submission due to resource load or failed job resubmissions, thus generating long completion tails. The merging procedure introduces the necessity of additional synchronization steps as described in the following section.

Finally, analysis efficiency is also a function of the analysis algorithm quality. Monitoring performance on train-by-train basis provides information on the code quality too. We have observed also efficiency improvements in case of using more compact data sets like analysis object data, or when reading only partially the input event information. To spot the sources of analysis inefficiency, we have to consider all these details and measure the effect of each on the overall budget.

4. Overview of operations 2012/2013

Considering the overall Grid activity for this period, the approximate share of resources used by the main ALICE distributed computation task was 60% simulation, 10% raw data reconstruction and 30% data analysis. From the 30%, the share between organized and individual user analysis was evenly split. The time profile of the individual user analysis efficiency with more than 200 users is shown in figure 2.

The average efficiency of 26% for analysis jobs in 2013 is contributing to the total ALICE job efficiency with a weight of 20%. This averages among user test and production jobs, most often non-optimized. These numbers are also influenced by the fact that most ALICE physics working group's analyses are now executed in organized central trains.

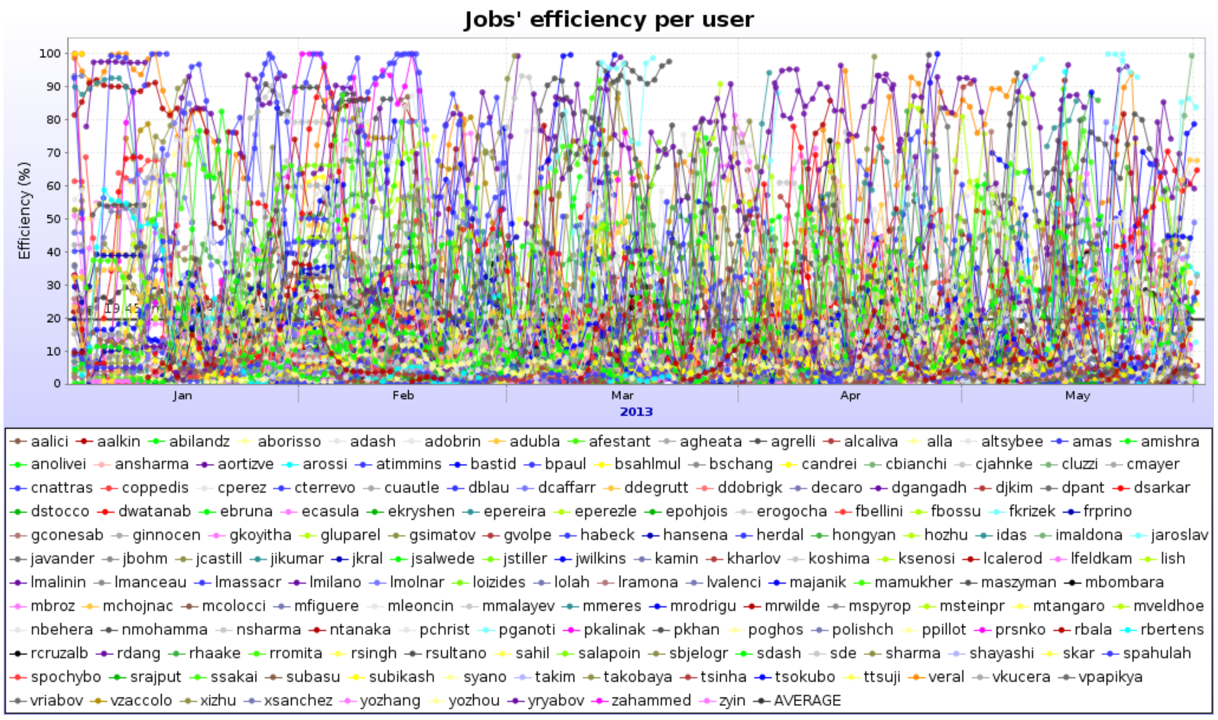
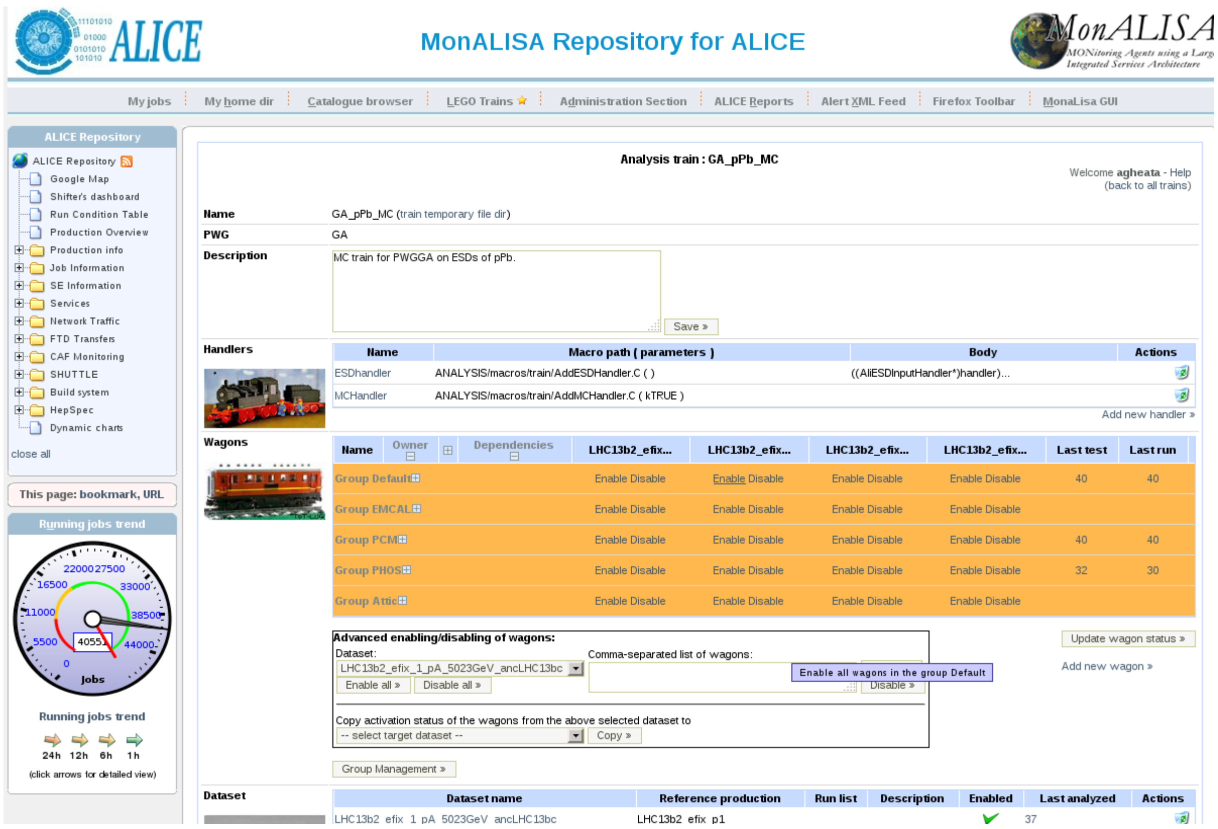


Figure 2. Time profile of individual user analysis efficiency for 5 months of operation in 2013. The average job efficiency was less than 20% for this period, spanning all possible values. This shows that such analysis mode is far from being optimal.

The benefits of running organized analysis come mostly from the fact that many analyses share the same data, which gives a large I/O reduction compared to running them individually. In addition, the CPU efficiency behaves like the best of the components. One of the downsides of organized analysis is the need to test thoroughly the performance of each individual component. This problem was addressed by a new LEGO-like system, allowing to assemble large analysis trains from individual components in an automated way [7]. The system performs individual component checks before any GRID job submission, preventing running any faulty analysis code. In addition, the LEGO frontend is based on a easy to use web interface and allows both users to register their analysis and interesting datasets, and administrators to group and schedule analysis trains according the working group priorities.

Figure 3 presents the typical interface and components of the LEGO framework. To be noted that this framework brought several improvements in both performance and code stability.


MonALISA Repository for ALICE

My jobs | My home dir | Catalogue browser | LEGO Trains | Administration Section | ALICE Reports | Alert XML Feed | Firefox Toolbar | MonaLisa GUI

ALICE Repository
- ALICE Repository
  - Google Map
  - Shifter's dashboard
  - Run Condition Table
  - Production Overview
  - Production info
  - Job Information
  - SE Information
  - Services
  - Network Traffic
  - FTD Transfers
  - CAF Monitoring
  - SHUTTLE
  - Build system
  - HepSpec
  - Dynamic charts

close all

This page: bookmark, URL

Running jobs trend

Running jobs trend
24h 12h 6h 1h
(click arrows for detailed view)

Analysis train : GA_pPb_MC

Welcome agheata - Help
(back to all trains)

Name	GA_pPb_MC (train temporary file dir)
PWG	GA
Description	MC train for PWGGA on ESDs of pPb.

Save »

Handlers

Name	Macro path (parameters)	Body	Actions
ESDhandler	ANALYSIS/macros/train/AddESDHandler.C ( )	((AliESDInputHandler*)handler)...	
MCHandler	ANALYSIS/macros/train/AddMCHandler.C (kTRUE)		

Add new handler »

Wagons

Name	Owner		Dependencies	LHC13b2_efix...	LHC13b2_efix...	LHC13b2_efix...	LHC13b2_efix...	Last test	Last run
Group Default				Enable Disable	Enable Disable	Enable Disable	Enable Disable	40	40
Group EMCAL				Enable Disable	Enable Disable	Enable Disable	Enable Disable		
Group PCM				Enable Disable	Enable Disable	Enable Disable	Enable Disable	40	40
Group PHOS				Enable Disable	Enable Disable	Enable Disable	Enable Disable	32	30
Group Attic				Enable Disable	Enable Disable	Enable Disable	Enable Disable		

Advanced enabling/disabling of wagons:
Dataset: LHC13b2_efix_1_pA_5023GeV_ancLHC13bc
Enable all » Disable all »
Comma-separated list of wagons:
Enable all wagons in the group Default
Disable »
Copy activation status of the wagons from the above selected dataset to
-- select target dataset -- Copy »

Update wagon status »
Add new wagon »

Group Management »

Dataset

Dataset name	Reference production	Run list	Description	Enabled	Last analyzed	Actions
LHC13b2_efix_1_pA_5023GeV_ancLHC13bc	LHC13b2_efix_p1			✓	37	

ALICE
MonALISA
MONitoring Agents using a Large Integrated Services Architecture
22000
27500
16500
33000
11000
38500
5500
44000
0
40551
Jobs

Figure 3. The ALICE LEGO framework interface, allowing registering analysis modules, datasets and composing trains. Each train is tested component by component on a dedicated facility, and then executed in the ALICE GRID. Every stage in this process is monitored using the MonALISA system [8].

The LEGO framework allowed for a detailed monitoring of the performance, both for sub-job efficiency and global execution of all involved stages: testing, job scheduling, execution and merging. During train operation, we have observed large inefficiency spikes for a large set of jobs that otherwise were behaving correctly in local tests. This behaviour was correlated to the fact that the job splitting algorithm managed only partially to assign local files to analysis jobs, so the faulty behaviour was assigned to the fact that few ROOT files were accessed through WAN. Indeed, ROOT issues a large number of read requests and not caching them leads to extremely expensive overheads related to latency to the remote disk server. Adding tree-caching support to the framework improved with a factor of more than two the overall analysis jobs efficiency, without having to modify the dispatching model.

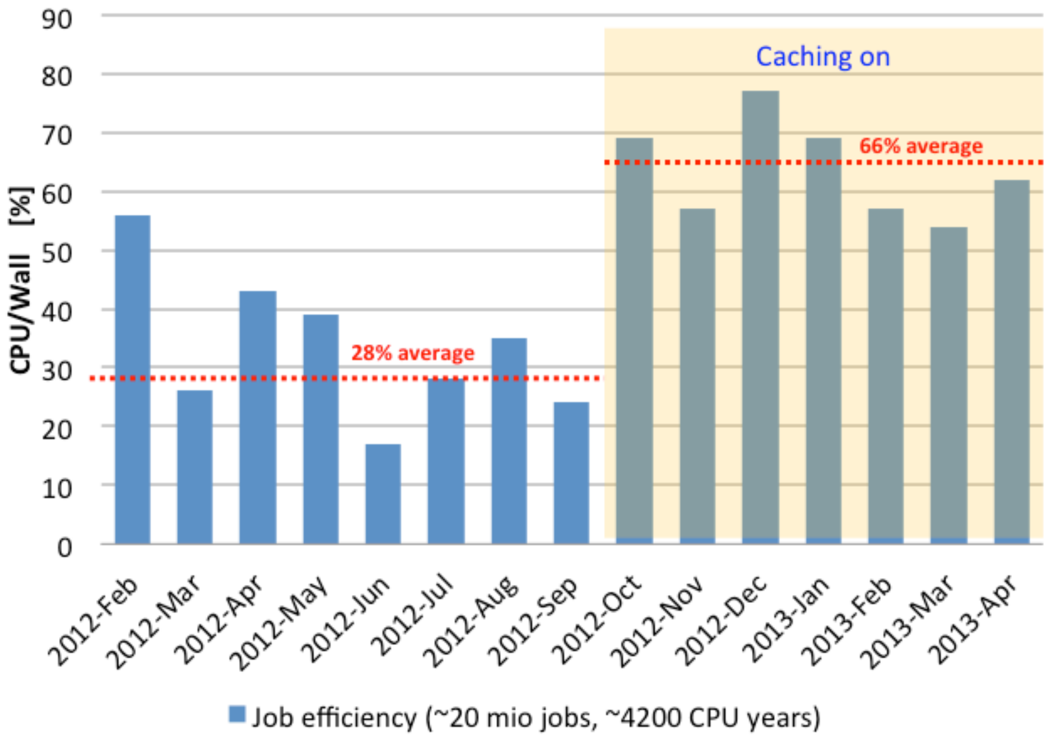


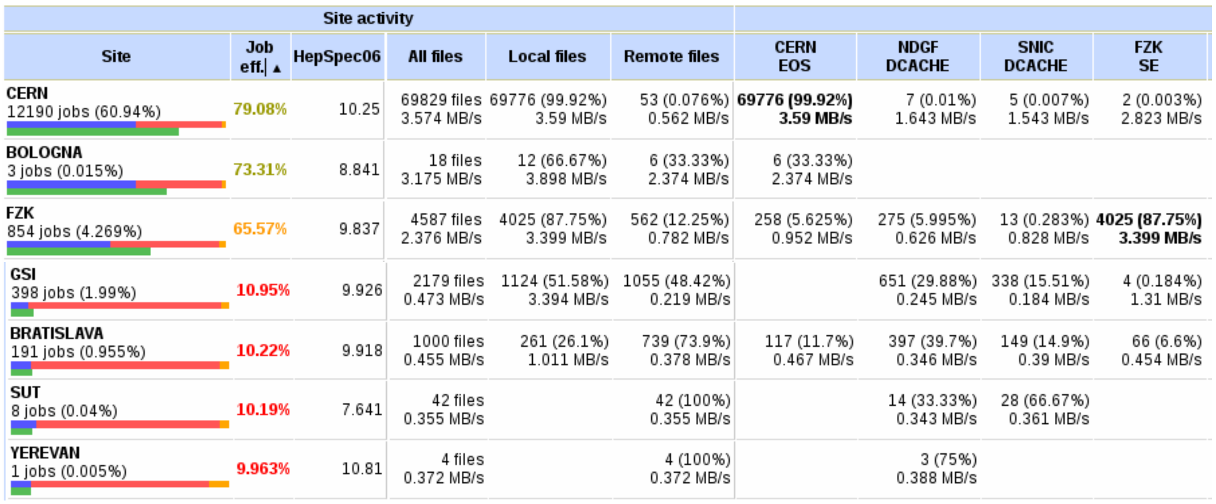
Figure 4. Overall efficiency improvement for ALICE organized analysis due to the introduction of *TTreeCache* support.

The current situation is that despite introducing tree caching, the individual user analysis efficiency is still low. To improve the overall Grid efficiency, moving ½ of the individual user jobs to organized analysis would result in an immediate efficiency gain of 7%. Another observation is that the time to solution is almost 3 times faster for compact analysis formats for I/O bound analysis, which encourages the migration to AOD-based analysis. Our analysis led to classifying the main actions to perform to alleviate the I/O problem, which is quite important for ALICE analysis:

- Caching I/O queries and prefetching. Recent measurements indicate extra gains of 10-15% when activating data prefetching in ROOT.
- Reducing the analyzed event size, having to assess the price of cutting down generality and multiplying the number of datasets.
- Selective disabling branches can alleviate I/O cost by a factor of 5, at the price of reducing the size of the analysis trains to just few modules.
- Encouraging custom filtering and skimming procedures in organized analysis can substantially reduce the input information requirements in particular cases. We distinguish the case of rare signal analysis requiring a small fraction of input events, or analysis requesting a small fraction of the available event information.

5. Instrumenting analysis performance

We have implemented low-level "sensors" in the analysis manager to collect detailed performance information at job level. The sensors collect timing information per analysis session, measuring the throughput and efficiency for every input file of the job. In addition, they collect information on the data sources, worker node identity or timing per analysis module within a train. The information is being read, processed and summarized by the monitoring system, allowing us to detect pathologies in the data flow, study data type effects and investigate possible flaws in train components. A typical summary output of a single train processing is given in Figure 5.



Site activity									
Site	Job eff.▲	HepSpec06	All files	Local files	Remote files	CERN EOS	NDGF DCACHE	SNIC DCACHE	FZK SE
CERN 12190 jobs (60.94%)	**79.08%**	10.25	69829 files 3.574 MB/s	69776 (99.92%) 3.59 MB/s	53 (0.076%) 0.562 MB/s	**69776 (99.92%) 3.59 MB/s**	7 (0.01%) 1.643 MB/s	5 (0.007%) 1.543 MB/s	2 (0.003%) 2.823 MB/s
BOLOGNA 3 jobs (0.015%)	**73.31%**	8.841	18 files 3.175 MB/s	12 (66.67%) 3.898 MB/s	6 (33.33%) 2.374 MB/s	6 (33.33%) 2.374 MB/s			
FZK 854 jobs (4.269%)	**65.57%**	9.837	4587 files 2.376 MB/s	4025 (87.75%) 3.399 MB/s	562 (12.25%) 0.782 MB/s	258 (5.625%) 0.952 MB/s	275 (5.995%) 0.626 MB/s	13 (0.283%) 0.828 MB/s	**4025 (87.75%) 3.399 MB/s**
GSI 398 jobs (1.99%)	**10.95%**	9.926	2179 files 0.473 MB/s	1124 (51.58%) 3.394 MB/s	1055 (48.42%) 0.219 MB/s		651 (29.88%) 0.245 MB/s	338 (15.51%) 0.184 MB/s	4 (0.184%) 1.31 MB/s
BRATISLAVA 191 jobs (0.955%)	**10.22%**	9.918	1000 files 0.455 MB/s	261 (26.1%) 1.011 MB/s	739 (73.9%) 0.378 MB/s	117 (11.7%) 0.467 MB/s	397 (39.7%) 0.346 MB/s	149 (14.9%) 0.39 MB/s	66 (6.6%) 0.454 MB/s
SUT 8 jobs (0.04%)	**10.19%**	7.641	42 files 0.355 MB/s		42 (100%) 0.355 MB/s		14 (33.33%) 0.343 MB/s	28 (66.67%) 0.361 MB/s	
YEREVAN 1 jobs (0.005%)	**9.963%**	10.81	4 files 0.372 MB/s		4 (100%) 0.372 MB/s		3 (75%) 0.388 MB/s		

Figure 5. Job efficiency and throughput are highly dependent on the data locality (one can check the percentages of "local files" and "remote files"). "Pathologies" can be easily monitored with this tool, pointing to transient operational problems, excessive load of resources or site misconfiguration problems.

5.1. Merging strategy

Merging the analysis jobs outputs is a general problem for analysis and can introduce large overheads. Having a single merging job per master job does not scale in execution time and memory requirements in the general case. We have adopted a different solution to alleviate the linear scaling with the number of files and to profit by the intrinsic parallelism offered by a distributed system. We organize merging in stages, with a limited number of files per job in a given stage. In this process we profit from the standard scheduler policies by providing a standard collection of output files to be merged at the end of the analysis jobs. The staged merging exhibits a log(N) scalability, where N is the number of stages (maximum 5). In Figure 6 we present job profiles provided by MonALISA system, which include the merging stages. Using this merging policy, we observe that the overhead is much less than the total job time.

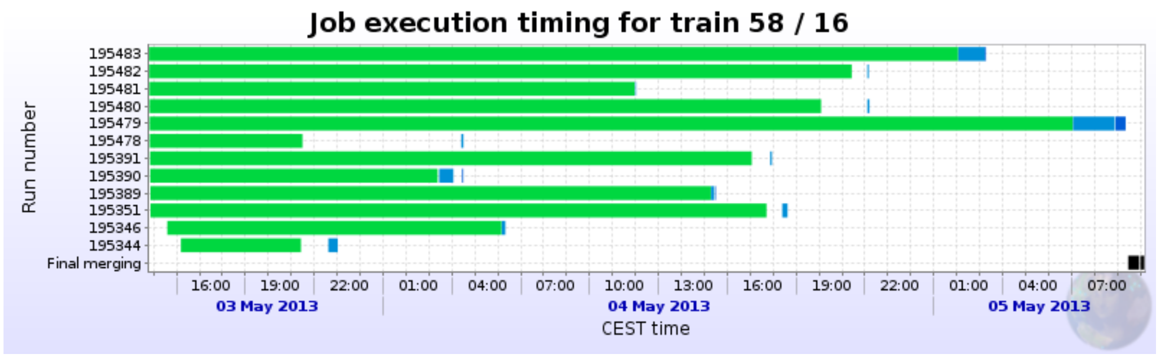


Figure 6. Typical ALICE analysis job time profile. The green bars represent the data processing time, the blue bars represent the merging stages per data set (in this case a run), while the final merging stage is the black bar starting as soon as the last merging job per run is completed.

6. Conclusions

The paper describes the current efforts to improve the performance of the ALICE distributed analysis jobs. The investigation method is quite general in its principles and aims to understand the limitations and possible bottlenecks of the current framework and computational fabric. We are able to monitor job efficiency information and throughput in relation with several factors: the input file location, input data type or execution site. The analysis of the collected monitoring data allowed for finding solutions to improve the overall efficiency in ALICE organized analysis by a large factor, and represent invaluable lessons for the evolution our analysis data formats and distributed processing patterns. Such studies are triggered by the future computing requirements in the LHC Run3, to be addressed within the ALICE O^2 project [3]. Many of these requirements are common for all LHC experiments. The evolution of the ALICE analysis framework toward a better performing system is continuing and the progress will be reported in future communications.

7. References

[1] The XROOTD project, xrootd.org

[2] Bagnasco S *et al* 2008 *J.Phys.:Conf.Ser.* **119** 062012

[3] The ALICE O^2 project, alicematters.web.cern.ch/?q=ALICE_02project

[4] Gheata A 2008 ALICE analysis framework *Proc. XII Int. Workshop on Adv. Comp. and Analysis Techniques in Phys. Research (Erice) PoS(ACAT08)* p028

[5] Gheata A 2012 Making distributed ALICE analysis simple using the GRID plug-in *J. Phys.: Conf. Ser.* **368** 012014

[6] The ROOT project, root.cern.ch

[7] The ALICE LEGO train system, https://alimonitor.cern.ch/trains/

[8] Legrand I *et al* 2009 *Computing Physics Communications* **180** Issue 12 December 2009 pp 2472-2498