Reducing the Human-in-the-Loop Component of the Scheduling of Large HTC Workloads

Frédéric Azevedo, Luc Gombert, Frédéric Suter

Reducing the Human-in-the-Loop Component of the Scheduling of Large HTC Workloads

Frédéric Azevedo, Luc Gombert, and Frédéric Suter

IN2P3 Computing Center / CNRS, Lyon-Villeurbanne, France
firstname.lastname@cc.in2p3.fr

Abstract. A common characteristic to major physics experiments is an ever increasing need of computing resources to process experimental data and generate simulated data. The IN2P3 Computing Center provides its 2,500 users with about 35,000 cores and processes millions of jobs every month. This workload is composed of a vast majority of sequential jobs that corresponds to Monte-Carlo simulations and related analysis made on data produced on the Large Hadron Collider at CERN.
To schedule such a workload under specific constraints, the CC-IN2P3 relied for 20 years on an in-house job and resource management system complemented by an operation team who can directly act on the decisions made by the job scheduler and modify them. This system has been replaced in 2011 but legacy rules of thumb remained. Combined to other rules motivated by production constraints, they may act against the job scheduler optimizations and force the operators to apply more corrective actions than they should.
In this experience report from a production system, we describe the decisions made since the end of 2016 to either transfer some of the actions done by operators to the job scheduler or make these actions become unnecessary. The physical partitioning of resources in distinct pools has been replaced by a logical partionning that leverages scheduling queues. Then some historical constraints, such as quotas, have been relaxed. For instance, the number of concurrent jobs from a given user group allowed to access a specific resource, e.g., a storage subsystem, has been progressively increased. Finally, the computation of the fair-share by the job scheduler has been modified to be less detrimental to small groups whose jobs have a low priority. The preliminary but promising results coming from these modifications constitute the beginning of a long-term activity to change the operation procedures applied to the computing infrastructure of the IN2P3 Computing Center.

1 Introduction

In the field of high-energy and astroparticle physics, detectors, satellites, telescopes, and numerical simulations of physical processes produce massive amounts of data. The comparison of these experimental and simulated data allows physicists to validate or disprove theories and led to major scientific discoveries over the last decade. For instance, in 2012, the ATLAS [10] and CMS [11] experiments running on the Large Hadron Collider (LHC) at CERN, both observed a

new particle which is consistent with the Higgs boson predicted by the Standard Model. These observations confirmed a theory of the origin of mass of subatomic particles which was awarded the Nobel Prize in physics in 2013. In 2016, the LIGO and VIRGO scientific collaborations announced the first observation of gravitational waves [13] which confirmed the last remaining unproven prediction of general relativity.

The next decade will see the beginning of major projects that will allow astroparticle physicists to address the most pressing questions about the structure and evolution of the universe and the objects in it. From 2022, the Large Synoptic Survey Telescope (LSST) will conduct a 10-year survey of the sky to produce the largest catalog of celestial objects ever built while the Euclid spatial telescope aims at drawing a 3D map of hundreds of millions galaxies from 2020.

A common characteristic to all these physics experiments is an ever increasing need of computing and storage resources to process and store experimental data and generate simulated data. Moreover, the sheer amount of data produced by physics experiments enforces the distribution of data and computations across a worldwide federation of computing centers.

The Computing Center of the National Institute of Nuclear Physics and Particle Physics (CC-IN2P3) [12] is one of the largest academic computing centers in France. It provides its more than 2,500 users from 80 scientific collaborations with about 35,000 cores and 340PB of storage. The reliability and high availability of the CC-IN2P3 allows it to achieve an utilization of these resources above 90%. In particular, the CC-IN2P3 is one of the twelve Tier-1 centers in the Worldwide LHC Computing Grid (WLCG) engaged in the primary processing of the data produced by the LHC and one of the only four centers that provide storage and processing resources for all four experiments installed on the accelerator.

This participation in the WLCG strongly influences the organization and the operation of the computing at the CC-IN2P3. It also defines and shapes the workload that is executed. Indeed, the four LHC experiments alone have used up to 75% of the allocated resources. In 2018, they represented 58% of the allocations, as the needs expressed by other experiments have been increasing.

The main characteristic of the CC-IN2P3's workload is that it is a High Throughput Computing (HTC) workload composed of a vast majority of sequential jobs. It mainly corresponds to Monte-Carlo simulation jobs and related data analysis made on the data produced at the LHC. We observed an increasing share of multi-core jobs (i.e., using several cores within the limits of a single node) in the workload over the last three years. Most of these multi-core jobs also run Monte-Carlo simulations but allow physicists to share some libraries and data structure and thus reduce the memory footprint. Finally, HPC jobs (e.g., using MPI or GPUs) are executed on a distinct set of nodes, which only represents a small fraction of the overall computing capacity of the CC-IN2P3.

The requirements of the main experiments running at the CC-IN2P3 influence the performance metrics the job scheduling system has to optimize. Indeed, the resource allocation procedure differs from that of traditional HPC centers

where scientific collaborations usually submit a research proposal which includes a request for an *envelope of cores.hours* to use during a limited time period. For a Tier-1 center of the WLCG such as the CC-IN2P3, resource requests are expressed as pledges for a given *computing power* expressed in HS06 [9]. The computing center is then committed to provide enough resources to answer to those pledges. Moreover, the accounting is made with regard to the actual CPU usage of a job rather than on its duration. This specific allocation procedure has been extended beyond the four LHC experiments to all the groups computing at the CC-IN2P3. The main objectives for the batch scheduling system are thus to ensure a fair sharing [4] of the resources according to the different pledges and to guarantee that all the pledges are respected. These objectives are translated into priorities and quotas assigned to jobs and user groups.

Scheduling also has to take into account the data-driven nature of the executed jobs. Several experiments running at CC-IN2P3 make a heavy use of the different storage subsystems the center provides (e.g., GPFS, HPSS, iRODS). To prevent the saturation of a storage subsystem and a failure which could have a cascading impact on the execution of the workload, additional conservative quotas are assigned to groups to limit the number of concurrent running jobs.

To schedule such a peculiar workload with regard to the aforementioned constraints and objectives, the CC-IN2P3 developed and maintained its own in-house job and resource management system for nearly 20 years. The development of the *Batch Queuing System* (BQS) started in 1992 and was initially based on NASA's *Network Queuing System* (NQS). This system has been tailored to suit the specific needs of the computing center and its major users. For instance, it was possible to "program" the scheduler to meet the production objectives expressed by the different experiments. Moreover, the respect of a fair sharing of the resources among the scientific groups and accounting mechanisms to ensure the respect of the pledges were part of the initial design.

The job and resource management system is complemented by a team dedicated to the operation of the computing infrastructure that adds an important "human-in-the-loop" component to the scheduling of the workload. Indeed, the role of the operators is not limited to reacting to incidents related to either resources or jobs. They can also directly act on the decisions made by the job scheduler and modify them. For instance, an operator can manually boost or lower the priority of a job/user/group or change the allocations of resources to a given queue in a proactive way.

The decision to stop the development of BQS was taken in 2011. It had become too costly in terms of human resources over the years. Since then, the job and resource management system of the CC-IN2P3 is Univa Grid Engine [14]. However, some legacy rules of thumb from the operation of BQS remained and add to the different rules motivated by the constraints on the hardware and software resources and the respect of pledges. This accumulation of rules sometimes acts against the job scheduler optimizations and forces the operators to apply more corrective actions than they should.

In this experience report from a production system, we describe the decisions made since the end of 2016 to transfer some the actions done by operators to the job scheduling system or simply make these actions become unnecessary. The objective is to improve the job scheduling decisions, especially for small user groups and optimize the resource utilization, while minimizing the "human-in-the-loop" component in such decisions Three complementary modifications have already been implemented which deal with: (i) the partitioning of resources; (ii) the quotas assigned to the different user groups; and (iii) the computation of the fair-share by the job scheduler.

The remaining of this paper is organized as follows. First, we describe how large HTC workloads are processed at the CC-IN2P3 in Sect. 2 by detailing its computing infrastructure and scheduling and resource allocation procedures and characterizing the executed workload. Then, in Sect. 3, we motivate, present, and illustrate the benefits, be they an optimization of the scheduling and/or a reduction of the operation costs, for each of the three proposed modifications. Section 4 concludes this experience report and details future work directions.

2 Scheduling Large HTC Workloads at CC-IN2P3

2.1 Organization and Management of the Computing Infrastructure

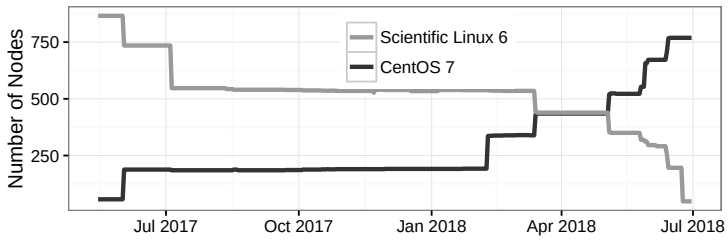
As mentioned in the introduction, the CC-IN2P3 provides its users with about 35,000 *virtual* cores (i.e., hyper-threading is activated on physical cores). More precisely, and at the time of writing of this article, this computing infrastructure is made of 816 nodes whose characteristics are given in Table 1.

Due to the recent upgrade of the default Operating System from Scientific Linux 6 to CentOS 7, this set of nodes is currently split into two distinct partitions of respectively 768 and 48 nodes. This allows the user groups that chose to not migrate their codes to get access to some resources. Nodes were progressively moved from one partition to the other from June 2017 to June 2018 as shown by Fig. 1. This variation in the size of the partitions and the set of user groups allowed to access each partition had an impact on several aspects of the scheduling process. We will highlight some of the consequences of this migration in the characterization of the workload given in Sect. 2.3.

Table 1: Characteristics of the nodes in the CC-IN2P3's computing farm.

Model	**#Nodes**	**#vCores / Node**	**#vCores**
Intel Xeon E5-2650 v4 @ 2.20GHz	232	48	11,136
Intel Xeon Silver 4114 @ 2.20GHz	240	40	9,600
Intel Xeon E5-2680 v2 @ 2.80GHz	149	40	5,960
Intel Xeon E5-2680 v3 @ 2.50GHz	123	48	5,904
Intel Xeon E5-2670 0 @ 2.60GHz	72	32	2,304
Total	**816**		**34,904**

Fig. 1: Transition from the Scientific Linux 6 Operating System to CentOS 7.



In addition to these nodes that are dedicated to the execution of the HTC workload, the CC-IN2P3 also offers resources for parallel (512 cores without hyper-threading in 16 nodes), GPU-based (32 NVIDIA K80 GPUs and 128 cores in 8 nodes), and large memory (a node with 40 cores and 1.5TB of memory) jobs, and five nodes dedicated to interactive jobs.

These computing resources are managed by Univa Grid Engine (UGE v8.4.4) whose scheduling algorithm is an implementation of the *Fair Share Scheduler* first described in [5]. Its principle is to assign priorities to all the unscheduled jobs to determine their order of execution. These priorities derive from three fundamental policies. The first policy is related to the *entitlement* of a job to access resources. It relies on the implementation of the *Share Tree* policy which defines this entitlement of a job according to the previous resource usage of a user/project/group. The administrators of the systems first define a total number of *tickets* which basically corresponds to a virtualized view of the complete set of resources managed by the system. This total number of tickets is then distributed among groups (and then among sub-projects, users, and eventually jobs). In the configuration used at CC-IN2P3, the different shares are proportional to the resource pledges expressed by the different user groups.

When a given group A does not use its allocated share, pending jobs of other groups are allowed to use the corresponding resources. The group with the least accumulated past usage has the highest priority in that case. However, when group A starts to submit jobs again, a compensation mechanism is triggered to allow this group to reach back its target share. Two parameters control the behavior of this policy, illustrated by Fig. 2. The *half-life* specifies how UGE forgets about the past usage of a given group. This parameter thus acts on the selection of groups allowed to benefit of the resources left unused by another group. The second parameter is the *compensation factor* that limits how fast a group will reach back its target share. The higher the value, the more reactive to variations in the workload the system will be. The current values used at CC-IN2P3 are 2,160 for the half-life and 2 for the compensation factor.

The second policy implemented by UGE corresponds to the expression of the urgency of a job and defines some weights in the computation of the priority of the job. This urgency is decomposed in three components. First, the closer a job is to its deadline (if one has been specified at submission time, which is not the

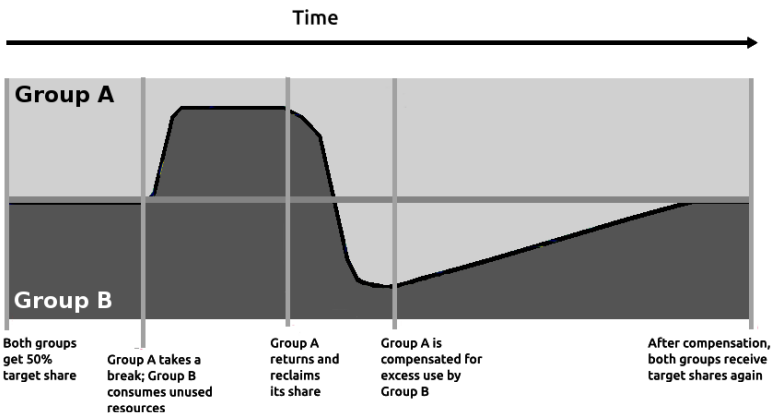


Fig. 2: Example of the Share-tree policy applied to two user groups allowed to use a half of the resources each.

case at CC-IN2P3), the higher its priority will be. Second, the priority will also increase along with the waiting time of the job in the scheduling queue. Finally, a higher priority will be given to jobs that request expensive resources. For instance, at CC-IN2P3, resources are organized in queues whose characteristics are given in Table 2. These queues mainly differs by the maximum duration, both in terms of wallclock and CPU times, of the jobs, the available memory and scratch disk space, and the type of jobs, i.e., sequential or multi-core. In the current configuration, a greater weight is given to the multi-core queues, which are thus seen as more "expensive resources" than sequential queues.

The sum of the maximum numbers of virtual cores that each queue can use is more than 2.86 times the actual number of available cores. This guarantees the highest possible utilization of the resources but also prevents the saturation of queues. Any type of job thus has a good chance to access resources, hence increasing the quality of service experienced by the users, who are not advised to specify a queue on submission. They are encouraged to express job requirements instead and let the scheduling system select the most appropriate queue.

Table 2: Upper bound on resources usage of the different batch scheduling queues.

Queue	**CPU time (in hours)**	**walltime (in hours)**	**memory (in GB)**	**disk space (in GB)**	**#vCores**
huge	72	86	10	110	10,337
long	48	58	4	30	31,040
longlasting	168	192	4	30	3,996
mc_huge	72	86	8	30	9,336
mc_long	48	58	3.6	30	34,752
mc_longlasting	202	226	3	30	20,416

The configuration of these queues also illustrates the operational priorities of the CC-IN2P3. We can see that the `mc_*` queues dedicated to multi-core jobs are allowed to access much more cores than the queues reserved to sequential jobs. This confirms the higher priority given to multi-core jobs which now represent a large fraction of the CPU consumption as the characterization of the workload given in Sect. 2.3 will show. We also note that jobs are not really distinguished by their execution time in this configuration. Indeed, the `huge` and `mc_huge` queues are primarily intended to jobs that need more memory or disk space. Moreover, the access to the `longlasting` queues is limited to certain user groups. Then, the bulk of the workload is directed to the `long` and `mc_long` queues. The rationale is to simplify the computation of the fair-share by the job scheduler whose respect is the main operational objective. However, this also calls for good estimations of job duration. Bad estimate can have two drawbacks. First, it may be harmful to the scheduler as it has to cope with important discrepancies between the "estimated" and actual duration of the jobs. Second, short jobs submitted by small groups whose priority is low with regard to the global fair-share policy may be severely delayed. Indeed, their short duration is not translated in an increase of their priority. As we will see in Sect. 2.3, estimations are not always provided or automatically defined and are usually far for being accurate.

The last component in the computation of job priorities by UGE is the capacity for users to manually specify a POSIX priority at submission which only acts as another weighting factor in the formula used to determine the overall scheduling priority of the job.

In addition to these policies implemented by the job and resource management system, a last configuration parameter has a strong influence on scheduling. This is the definition of limitations as *Resource Quota Sets* (RQS). These limitations are expressed as a maximal number of virtual cores (or slots in the UGE terminology) that can access a given hardware of software resource and thus be allowed to enter the system. They are applied at two levels. Global limitations are applied to all groups and jobs indifferently. Such limits are classically used to define resource pools (e.g., depending on the operating system running), prevent the saturation of a storage or database service, or are related to the number of available license tokens for commercial software.

In the specific configuration of the CC-IN2P3 system, extra RQS are applied to groups to either limit the number of concurrent jobs or the number of jobs accessing a given resource. The former is used as a way to enforce the respect of the resource pledges expressed by each group by averaging their estimated consumption over each quarter of the year. The latter corresponds to the implementation of a conservative approach to further prevent the saturation of sensitive storage subsystems such as a shared parallel file system.

2.2 Resource Allocation Procedure

Every year in September, the representative of each of the scientific collaborations using the CC-IN2P3 is asked to pledge resources for the next year. Each group provides an estimation of its needs in terms of computing and storage on

each of the available subsystems. While the large collaborations such as those of the LHC experiments have a well defined and planned definition of their requirements at a worldwide scale, smaller groups usually define their needs from their consumption of the previous year with an empirically estimated delta.

The accuracy of these pledges is critical for two reasons. First, the sum of the expressed requirements, combined to the available budget, define, after an arbitration process, the purchase of new hardware to ensure that the CC-IN2P3 can fulfill its primary mission and serve all the experiments. This is another major difference with traditional HPC centers that buy and host the biggest affordable supercomputer and then arbitrate the demands with regard to the capacity of this machine. Second, the pledges define the Resource Quota Sets applied to each group and thus have a direct impact on job scheduling.

The allocation of computing resources works as follows. Each group expresses its pledge as an amount of work to be done during each quarter of the year. This amount is given in *Normalized HS06.hours*, a unit coming from the High Energy Physics community. It corresponds to the normalization of the results of the HS06 benchmark [9] on the different types of nodes in the computing infrastructure to take node heterogeneity into account multiplied by a number of hours.

The accumulation of all the required numbers of HS06.hours defines the computing power the CC-IN2P3 has to deliver. Once arbitration has been done, the respective share of this total number that has to be allocated to each group is computed. Then this share is converted into a number of virtual cores needed to process this amount of work in a year. Finally, this number of virtual cores defines a consumption objective used by the job scheduler to compute its fair-share.

2.3 Characterization of the Workload

We analyzed the workload processed at CC-IN2P3 over ten weeks from March 29, 2018 to June 12, 2018. This corresponds to the period between two scheduled maintenance shutdowns of the computing center. Job submissions are blocked a day before the maintenance in order to drain the scheduling queues while jobs are progressively allowed into the system after the maintenance to prevent stress on the storage subsystems.

We extracted and combined information on jobs from three tables of the Accounting and Reporting Console (ARCo) provided by Univa Grid Engine:

- The `sge_job` table contains basic information about jobs such as the job id, the user who submitted it and its group, or the submission date;
- The `sge_job_usage` table contains information about the resource usage made by a job, including its beginning and end date, the number of cores on which the job has been executed, its memory consumption, or its exit status;
- The `sge_job_request` table stores key-value pairs that correspond to the different resource requests made by jobs, such as the requested number of cores or memory, an estimation of the walltime, the specification of a given queue, or the need for a specific storage subsystem.

We performed a first processing of these database extracts to solve two major issues related to the way UGE stores information into ARCo. First, ARCo creates several entries for jobs whose execution spans over more than a day. For instance, for a job starting on April 25th at 5 PM and ending on April 26th at 11 AM, two entries will be created, one for April 25th from 5 PM to 12 AM and one for April 26th from 12AM to 11 AM. Moreover, only the last entry will log the total runtime of the job, the other ones leaving that field to 0. We thus had to merge all the entries for such long jobs into a single one. Second, the CPU consumption is not stored in seconds but in *normalized HS06*, the unit used for resource pledges. A normalization factor is applied that depends on the node onto which the job has been executed to take into account the heterogeneity of the computing nodes. This factor ranges from 9.7 to 11.3. For each job in the workload, we found out the normalization factor that was applied to the actual CPU consumption and converted the value back to seconds. Finally, we kept the jobs that start before (resp. end after) but end (resp. begin) within the considered period. This allows us to have a complete view of the workload over the 10 weeks. However, these jobs may be excluded for some of the specific analyses we present in the remaining of this section.

We developed a Python script to convert this information to the Standard Workload Format (SWF) [1] used by the Parallel Workloads Archive [3]. This format describes a job by 18 fields. While the conversion was straightforward for some of these fields, e.g., job number, submission, start, end and wait times, or number of allocated processors, others required more thinking and work.

The memory consumption of the jobs is logged by ARCo as two complementary metrics: the integral memory usage expressed in GB.CPU.seconds, i.e., the average memory consumption of the job, and the *maximum Resident Set Size* (RSS). We decided to keep the latter as exceeding the maximum amount of memory allowed by the configuration of a queue would cause the failure of the job. Consequently, we also used the RSS as the requested amount of memory, even though jobs also expressed requirements related to Virtual Memory.

The most problematic field in this conversion to the SWF format was the *time requested* by jobs. Indeed, the run time of a job is classically computed as the difference between its end and start times. However, at submission time, users are encouraged to express their requests not as a hard or soft walltime limit (using the `h_rt` or `s_rt` flags) but as a hard or soft CPU time limit (using the the `h_cpu` or `s_cpu` flags). Again, this is dictated by the plegding and accounting procedures that use HS06 as main metric, and thus an efficient CPU consumption as both an objective and a performance indicator. To reflect this peculiar way of expressing the maximal duration of a job, we fill the *Requested Time* field of the SWF with the *CPU Time* requested by users.

Information about users and groups has been anonymized by converting them to integer values. The configuration of UGE allows administrators to distinguish different *projects* within a user group. For instance, for the ATLAS collaboration, the Monte-Carlo simulation jobs do not belong to the same project as the data analysis jobs. As the induced workloads may differ a lot from one project to

another within a single user group, we decided to reflect this differences in the produced SWF file by using the *project* information rather than the *group* one.

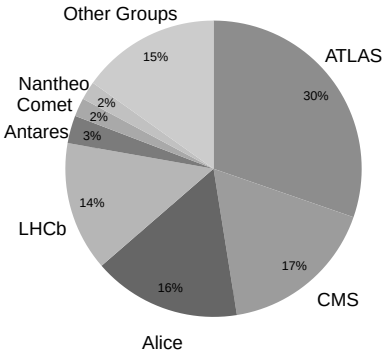
For the fields related to the queue and partition to which a job has been scheduled, we associate two conversion tables to the SWF logs. There are six queues, described in Tab. 2, and two partitions defined by the running Operating System, whose boundaries evolved over the considered period as shown in Fig. 1. Finally, three fields (i.e., *Executable (Application) Number*, *Preceding Job Number*, and *Think Time from Preceding Job*) remain undefined as the corresponding information was not logged by UGE.

The resulting SWF file is composed of 7,607,154 individual jobs. Almost 90% of these jobs (i.e., 6,824,118 jobs) are sequential. However, if we consider the cumulative CPU consumption of jobs, we observe a different distribution. Multi-core jobs represent about 40% of the CPU consumption. More precisely, 96.1% of these multi-core jobs run on eight cores and are submitted by only two users groups: ATLAS (73.3%) and CMS (22.8%).

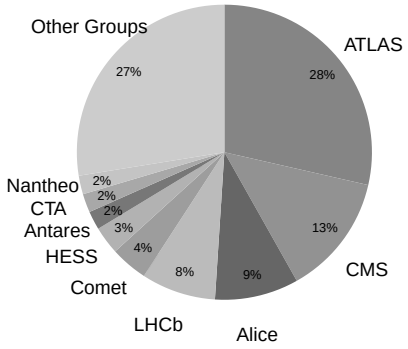
We start our analysis by comparing the CPU usage made by the different user groups to their pledges. This is a way to measure how well the job scheduler allocates resources to groups with the objective of respecting the fair-share that derives from these pledges. Figure 3a shows the observed distribution of the CPU usage over 10 weeks while Figure 3b shows how the pledges for computing power were distributed among the different user groups for 2018. We only distinguish the groups whose shares are greater than 2% of the total CPU consumption in these graphs for the sake of readability. The "Other Experiments" section aggregates the usages and pledges of the more than 70 other user groups.

Fig. 3: Distribution of the computing resource consumptions (a) and requests (b) among the different scientific collaborations using the CC-IN2P3.

(a) Usage from 03/29/18 to 06/12/18



(b) Pledges for 2018



The fact that most of the user groups with the largest pledges are also the biggest consumers of CPU resources confirms the overall respect of the fair-share by the batch scheduling system. On both graphs, the four LHC experiments (ATLAS, CMS, Alice, and LHCb) have the biggest shares, which explains why the whole resource allocation procedure and the performance objectives assigned to the job scheduler are driven by the demands of these groups. However, those graphs also show large discrepancies between pledges and usages that can be explained by the combination of several factors. First, the submission patterns of the different groups vary over the year in periods of high or low activity. Another monitoring tool used by the operation team shows that the two groups that are among the top pledgers but not among the top users (i.e., CTA and HESS) did not submit enough jobs to meet their objectives for the second quarter of the year. Conversely, the Alice, CMS, and LHCb groups increased their submissions with regard to their objectives and took advantage of these unused shares. Second, the upgrade of the default OS and the moving boundaries of the partitions shown by Fig. 1 were beneficial to some groups that were among the first to migrate. They were thus able to access a less crowded set of nodes while others groups had to compete for a smaller amount of CPU resources. However, we consider that this behavior is related to a transient yet impacting event and should not have appeared so clearly with a single stable partition.

An important characteristic of the considered workload is that almost half the jobs (47.55% or 3,615,225 jobs) are submitted through a *grid middleware stack* while the other half (52.45% or 3,991,929 jobs) is submitted by actual users directly to the batch systems. Figure 4 shows the *daily* (left) and *weekly* (right) arrival rates for these *Grid* and *Batch* jobs.

We clearly observe two different submission patterns. While the batch job arrival rate follows a traditional "working hours and business days" pattern, grid jobs are submitted at an almost constant rate. We also note that the average number of submissions per hour over the considered period is very similar for both types of jobs (respectively 1956 and 2160 jobs per hour).

Fig. 4: Daily (left) and weekly (right) arrival rates for Grid and Batch jobs. For the daily arrival rate, the gray area depicts the typical working hours.

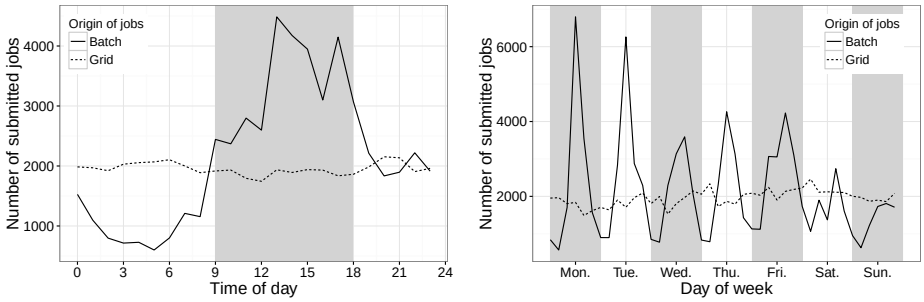
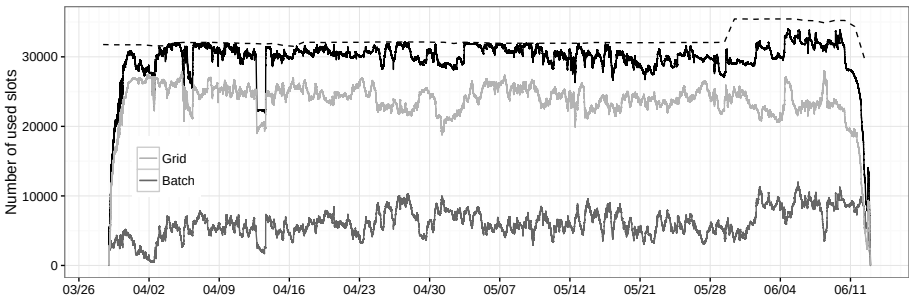


Fig. 5: Utilization of the resources, in terms of slots, over the considered period. The dashed black line indicates the total number of available slots, the solid black line represents the overall utilization and the other two lines show which part of this utilization respectively comes from grid and batch jobs.



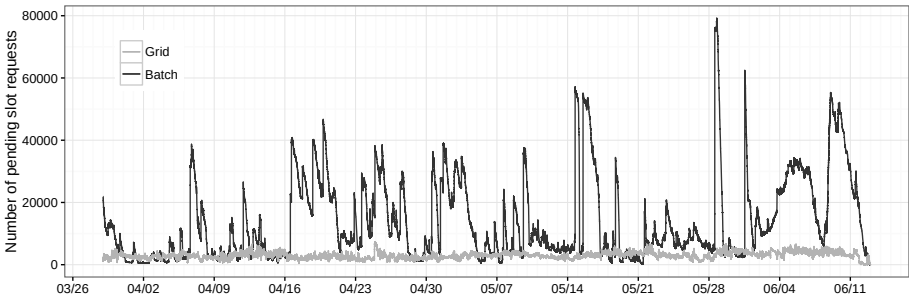
Then, we study what is the utilization of the computing resources induced by these submissions. Figure 5 shows how many virtual cores (or slots) are simultaneously used. The dashed black line indicates the total number of available slots which evolved over the considered period. The solid black line represents the overall utilization, while the other two lines show which part of this utilization respectively comes from grid and batch jobs.

We observe that one of the major operational objectives of the CC-IN2P3 is met as the utilization is generally well over 90%. One noticeable exception is around April 13th when a change in the configuration of UGE prevented jobs to be scheduled. We also see that while there are more jobs submitted by local users than coming through the grid, the number of slots respectively used by these two types of jobs shows a different distribution. About 80% of the overall utilization in terms of slots comes from grid jobs. This important difference can be explained by the fact that the vast majority of the multi-core jobs executed on CC-IN2P3's resources and most of the jobs related to the four LHC experiments come from the grid.

An interesting thing to note is that when less slots are used by grid jobs, more are used by batch jobs to reach a close to maximum overall utilization. This could mean that the jobs and resource management system makes a good job at balancing the resource allocations between the two categories of jobs. However, if we consider the number of pending slot requests, i.e., the sum of the numbers of slots requested by jobs waiting in the queues, we observe a less ideal situation shown by Fig. 6.

We observe that there are much less pending requests for grid jobs than for batch jobs. Over the whole period, around 3,000 slots are requested by grid jobs that have to wait for resources to be available while this average is of nearly 14,000 for batch jobs. The difference between the maximum number of pending slot requests is even more glaring: 7,400 for grid jobs and 79,200 for batch jobs.

Fig. 6: Evolution of the number of pending slot requests over the considered period with regard to the job category.



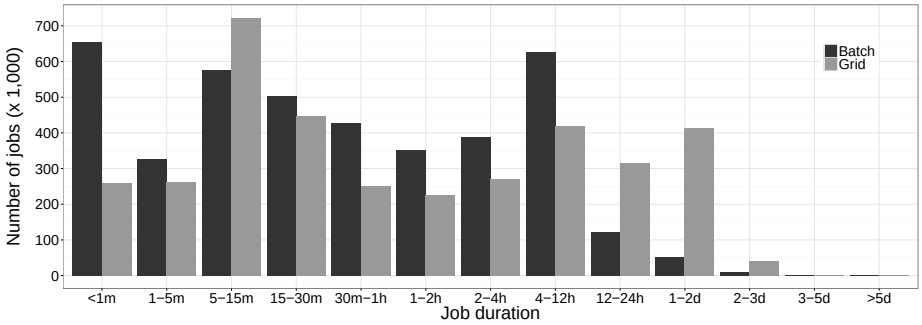
More importantly, we observe three periods (April 16-20, May 3-4, and June 4-7) where this number of pending slot requests for batch jobs remains well above 15,000 for more than two days.

Several factors can explain the results shown in Fig. 6. First, the differences in submission patterns illustrated in Fig. 4 indicate that a high number of batch jobs can be submitted at certain periods and needs to be absorbed by the system. To some extent, we can observe peaks in Fig. 6 at the beginning of each week that match the higher submission rates for Mondays and Tuesday shown by Fig. 4 (right). Conversely, the almost constant submission rate of grid jobs can be straightforwardly translated in an almost constant number of slot requests and explain the little variations we observe for this category of jobs. Second, grid jobs are usually not directly submitted to the batch system. The middleware stacks used by the different experiments typically include another layer that control the submission rate according to the observed status of the queue and the number of running jobs. A third potential cause is that grid jobs are submitted by groups with the largest shares, hence the highest priorities. Then, batch jobs submitted by smaller groups tend to have lower priorities, experience more delays in their scheduling, and thus tend to accumulate in the scheduling queues. However, these reasons do not fully explain why the observed peaks in the number of pending slot requests take so much time to be absorbed by the system. To understand that, we have to look at the duration of the jobs.

Figure 7 shows the distribution of job duration and distinguishes grid jobs from batch jobs. In the studied log, 84% of the submitted jobs are in the `long` or `mc_long` queues whose limits are set to two days of CPU time. 4% are "long lasting" jobs that can consume for up to 7 days of CPU time, while the remaining 12% correspond to "huge" jobs which are limited to three days of CPU time.

As nearly 58% of the studied workload is composed of jobs that run for less than an hour, we created more intervals for these very short jobs. We observe two major peaks at more than 600,000 jobs in the less than a minute interval for batch jobs and in the five to fifteen minutes for grid jobs. For grid jobs, the

Fig. 7: Distribution of the number of jobs according to their run time from March 29, 2018 to June 12, 2018.



explanation of this large amount of very short jobs pertains to the use of *pilot jobs* to create a steady resource pool. When a pilot starts but realizes that it has no jobs to execute, it ends itself after a short time, i.e., up to 15 minutes. For batch jobs, failed jobs accounts for nearly 30% of the jobs that end in less than a minute. Other potential causes need to be investigated but the bulk of these very short jobs was submitted by only five identified users groups.

At the other end of the range of job duration, we can see that about 52,000 jobs (0.67%) have a duration greater than two days, which is the upper bound of the `long` queue. However, more than 41,000 of these jobs were submitted in the `long` queue. The reason is that these jobs do not fully use the CPU and can then run beyond the limit of 48h of CPU time up to 58h hours, which is the upper bound of this queue with regard to execution time. However, the vast majority of the jobs that were submitted to the `huge` and `longlasting` queues last for less than a day. This indicates that groups with an access to these reserved queues submitted jobs that should have gone in the regular `long` queue.

This analysis pleads for a redefinition of the scheduling queues, and the pools of resources they can access to better take the characteristics of the workload into account. For instance, a classical rule is to assign to a queue a number of resources that is inversely proportional to the duration of the jobs submitted in this queue [7]. The rationale is that the system can afford to concurrently execute a large number of short jobs, as they will release the resources soon. Conversely, the number of long running jobs has to be controlled to prevent large delays for shorter jobs caused by the lack of available resources. The importance of the configuration of the queues on the quality of the produced schedules has been outlined in [7,6]. However, such a work still has to be done at CC-IN2P3 and will require to measure and balance the effects of adding more queues to the system with regard to the respect of the fair sharing of the resources, hence the respect of the pledges made by the user groups. Having a precise estimation of the duration of a job at submission time is key to the success of such a redefinition of the scheduling queues.

Fig. 8: Cumulative distribution functions (CDF) of actual and estimated job CPU times for batch and grid jobs.

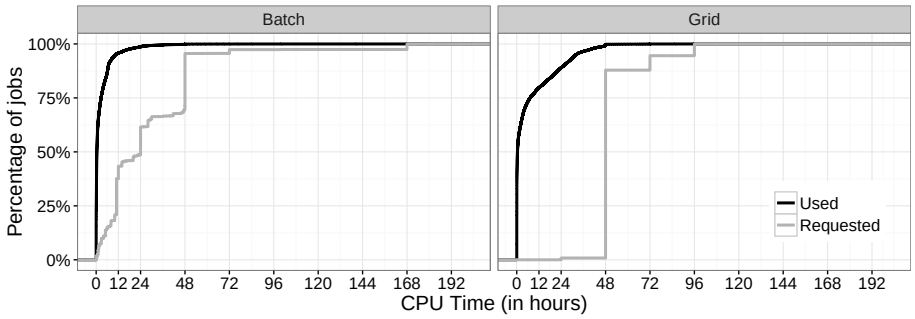
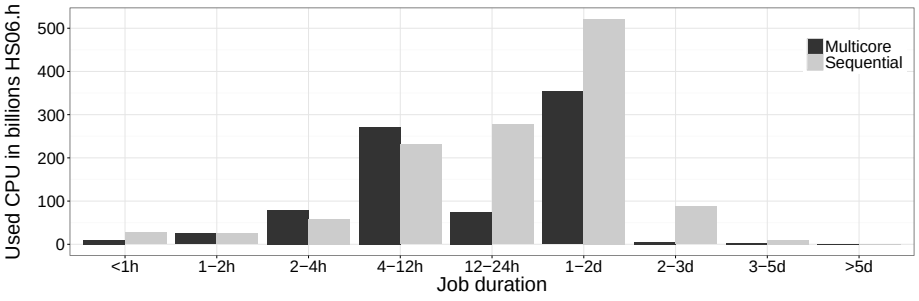


Figure 8 shows that the estimations of the CPU time consumption associated to the jobs, when specified, are far from accurate. For batch jobs, many different estimations are provided by users, most of them being either straightforward (e.g., 12, 24, or 48 hours) or arbitrary (e.g., 20,000 seconds or 47 hours) values. For grid jobs the situation is even worse. The small set soft of CPU time requests shown by the steps in the grey line of the right panel of Fig. 8 are automatically added by one component of the grid middleware stack and correspond to the limits of the queues. These requests are thus not related to the profiles of the jobs at all. Improving these estimations will imply to better tune the configuration of the grid middleware component and to better inform and form the users about the consequences of the provided estimations on the scheduling of their jobs.

We end this characterization of this 10-week workload executed at CC-IN2P3 by considering the distribution of the CPU utilization with regard to the duration of the jobs. Figure 9 shows that jobs running in less than 24h represent an important part of the system utilization which is not reflected in the configuration of queues. This also confirms the impact of multi-core jobs which only account for 10% of the submissions but 40% of the CPU consumption.

Fig. 9: Distribution of the CPU usage by job run time.



3 Reducing the Human-in-the-Loop Component

In this section, we motivate and explain the modifications made to the configuration of the job and resource management system during the year 2017. However, this experience report does not include any quantification of the benefits of these modifications. The main objective was to reduce the burden put on operators by improving the decisions made by the batch scheduling system, hence automating some of their daily interventions. Such interventions were not tracked before the modifications. The evaluation of the gain would thus have been subjective and difficult to quantify.

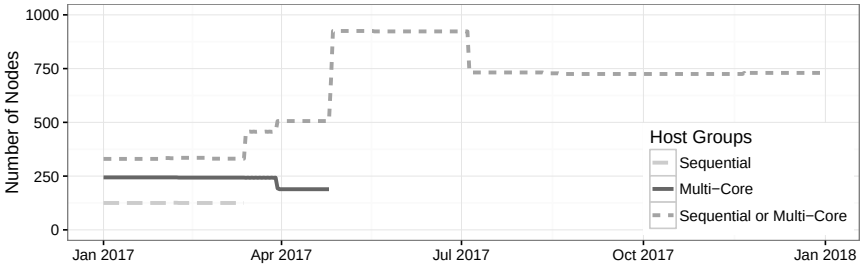
3.1 From Physical to Logical Resource Partitioning

Despite the efforts made while purchasing new hardware to keep the computing infrastructure as homogeneous as possible, nodes used to differ a lot in terms of CPU power and amount of memory from one model to another. A direct consequence of this heterogeneity was that some nodes were more suited than others to the execution of the 8-core jobs that require more memory. As mentioned in the previous section, almost all of these jobs are submitted by the two main experiments (in terms of resource allocation and consumption) running at the CC-IN2P3. They are thus considered of the highest priority.

Before March 2017, the computing infrastructure was physically partitioned in three *host groups* as shown in Fig. 10. One was dedicated to sequential jobs (125 nodes), another to multi-core jobs (245 nodes), and the third and largest one (330 nodes) accepted the execution of both sequential and multi-core jobs.

The primary motivation of such a partitioning was to guarantee that the high priority multi-core jobs can start without having to wait for the completion of several sequential jobs. A secondary motivation was to keep the capacity to allow sequential jobs to run on these nodes dedicated to multi-core jobs when they become idle. The major physics collaborations such as ATLAS or CMS usually execute an important part of their computations during planned *campaigns*.

Fig. 10: Transition from a physical to a logical node partitioning in 2017. The variation of the number of nodes from May to July corresponds to the period between the reception of new nodes and the decommission of old hardware.



They are also able to coordinate the use of multiple computing centers at a continental scale to distribute the load. Then, there can be variations in the job submission pattern, and periods of lower load could be exploited by other user groups. However, this management of distinct resource pools had a high operational cost. For instance, if a decrease in the submission of multi-core jobs by the ATLAS experiment was detected by the operators, they first had to check with the dedicated support to determine if this behavior was expected and know the duration of the lower load period. Then, nodes were manually reassigned to the mutualized host group to keep them utilized. A safety margin was kept in case the submission rate of ATLAS starts to increase earlier than excepted. This safety margin could be exploited by other groups running multi-core jobs such as CMS, but as the estimated end of the low load period got closer, more stringent limitations had to be manually applied to these groups.

With the end of Moore's Law, the node heterogeneity tends to disappear. New processors have more cores but there are no important clock rate deltas from one generation to another anymore. The historical physical partitioning of the resources is thus no longer justified. To simplify the management of the computing infrastructure, it has been replaced by a logical partitioning. In other words, the existing host groups have been merged, as shown by Fig. 10, and the distinction between sequential and multi-core jobs is now handled by the queues presented in Table 2. This change is almost transparent for the users (who are not supposed to specify a queue) as the job scheduler can automatically assign multi-core jobs to one of the `mc_*` queues. However, this is an important change from the operational point of view. Indeed, operators no longer have to manually specify the boundaries of the resource pools based on experience and rough estimations of the foreseen evolution of the submission patterns. This burden is now transferred to the job scheduling system which has been designed to adapt its decisions according to the respective filling of the queues.

3.2 Simplification of the Access Rule and Quota Mechanisms

The origin of the definition of *resources* and the application of quotas on these resources goes back to the use of the BQS job and resource management system. The term "resource" first encompassed job related parameters such as the required operating system, the needed amount of memory, the maximum CPU time, or the queue in which to place the job. This definition was rapidly extended to cover the different services offered by the CC-IN2P3 that a job could need.

A motivating example for such a definition of resources and quotas is the case of a job that need to fetch data from a distributed storage system S_1, store the produced results on another storage system S_2, and also needs to access a relational database D during its execution. Such a job can succeed if and only if all of the three dependencies on external services S_1, S_2, and D can be satisfied. Specifying the needed resources at submission time allows the job scheduler to delay a job if one or more services are not available (e.g., because of an incident or a temporary saturation) to prevent an unavoidable failure of the job.

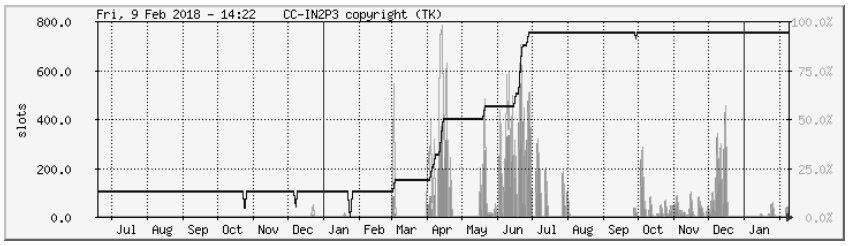
This mechanism has then evolved into a two-level quota mechanism. The first level defines global limits on the number of concurrent jobs that can access a given resource without regard to the submitter. Such limits allow to prevent the different storage subsystems or database services to be overloaded, ensure that the number of license tokens for a commercial software is not exceeded, or define physical pools with different versions of the operating system when an upgrade in underway. The second level specifies quotas for {resource, group} couples. The rationale is to be able to block or limit the access of a given group to a specific service more easily. For instance, if a group has filled its allocated disk space on a storage subsystem, jobs that could write more data will be rejected until more space has been granted or cleaning has been made. This also allows operators to easily drain the use of a resource for maintenance operations or incident recovery by blocking all the jobs that expressed a dependency on that resource.

Over the years, this appealing way to ensure a fine regulation of the job submission and to optimize the utilization of the computing and storage infrastructures became a very complex set of rules, thresholds, and locks to control the maximum number of jobs per user, group, machine, or service. The multiplication of resource definitions, hence the accumulation of limits for a given user group not only slows down the scheduling rounds as the job scheduler has to check everything, but also makes it sometimes difficult to understand why some jobs cannot enter the system. For instance, a restrictive limit may have been applied at some point, and for a good reason, to a certain {resource, group} couple and not been reconsidered afterwards. Then, sometimes months later, users complain that their jobs do not run for what they consider as no good reason, because of this persisting but forgotten limit.

At the end of 2016 the decision has been made to simplify the access rule and quota mechanisms. The main objective was to reduce the number of declared *resources* to only keep a minimal set of essential requirements that jobs have to express. The CC-IN2P3 being a production center, such changes have to be done carefully to prevent any major disruption of the activity. The chosen solution was to progressively relax the quota associated to a {resource, group} when it becomes a bottleneck while ensuring that the associated resource can cope with the increase. Figure 11 illustrates this action. Eventually, when the limit is high enough, it obviously becomes meaningless and can thus be safely removed. This was especially done for quotas related to the storage subsystem.

A similar method has been applied to the maximum number of jobs of a group that can run simultaneously. Such RQS were defined as a way to enforce the fair-share and to be able to rapidly react to an unwanted overconsumption of the resources by a given group. However, this kind of limit was especially harmful to small user groups whose computing needs correspond to short bursts of a large number of jobs every once in a while, for instance just before a deadline for the submission of an article. In such a case, users had to open an issue on the user support ticketing system to explain that they would like to see more of their jobs running. Then, the operation team would grant this exception by manually relaxing the quota and/or boosting the priority of jobs.

Fig. 11: Relaxation of a per-group RQS on a storage subsystem. The black line indicates the maximum number of *slots* (i.e., virtual cores) currently available to the group. The grey part corresponds to the number of used slots.



After a few months of operation, we can conclude that letting the job scheduler deal with submission bursts without any human intervention is a success. The concerned groups reduced their time to solution without harming other groups. This also reduces the load of both the operation and support teams who have less tickets to handle. However, some limits have to be kept for certain groups whose jobs have a specific greedy behavior or are highly sensitive to the accessibility of the storage subsystems.

3.3 Extending the Fair-Share History Window

The last important modification made to the configuration of the job scheduler is related to the implementation of the fair sharing of resources. The basic principle of a fair-share allocation is, for each user/group, to assign a priority to jobs that is inversely proportional to the usage of the resources by this user/group over a sliding time frame. The rationale is very simple: if a group already computed a lot, it has to make room for another that did not. Then, this group will compute less (and see its priority increase) while the other computes more (and its priority decreases). A key configuration parameter of such an algorithm is the size of the time frame over which to compute the resource usage.

Until the end of 2016, the size of this history window was set to 24 hours. As for many other parameters, this value was motivated by the predominance of the LHC-related jobs in the workload and the commitment to fulfill the pledges for these experiments made by the CC-IN2P3. Such a short time window was one of the levers to ensure a good reactivity of the system when the largest groups, i.e., ATLAS or CMS, started to submit jobs after a period of inactivity. These jobs got the highest possible priority and were scheduled immediately.

The main drawback of this strategy is that the jobs submitted by groups with much lower priorities suffered from large delays. From the point of view of the users belonging to these groups, the fair-share was felt as particularly unfair. To circumvent this issue, the operation team developed several mechanisms to ensure that smaller groups were not disadvantaged. For instance, they develop a script that ensured a minimal number of running jobs per user by modifying the priorities of some jobs to force the system to schedule them earlier. Another

technique was to specify additional resource definitions (as explained in the previous section) dedicated to these groups. Adding a requirement on this "special" resource at submission time allowed the jobs to bypass the fair-share mechanism completely and start almost immediately.

The proposed solution to ensure a fair sharing of the resources for *all* the user groups, be they small or big consumers, without having to bypass or modify the decisions made by the job scheduler was to increase the time frame used to determine the priorities. To prevent any harmful disruption of the production due to this change, we decided to progressively and empirically increase this value. It was first set to 15 days in January 2017, then to 30 days in March, and finally to 90 days in June 2017. After each modification, the operation team monitored its impact on the production. While benefits for small groups and no loss of the quality of service for the largest groups were observed, the time frame was increased. A more principled process based on the particular setting of the CC-IN2P3 would obviously have to be found. However, this would require a parametric impact study combined to a thorough evaluation through simulation before being deployed in production that is part of our future work.

The second modification made to the priority determination mechanism is related to the metric used to measure the resource utilization. Historically, the priorities were based on the *used CPU time* because the pledges made by the experiments are expressed as a CPU consumption. This metric is thus used to determine if the computing infrastructure can satisfy all the pledges and for the accounting of the resource usage. However, it may also favor inefficient jobs, i.e., jobs that are unable to fully exploit the CPU. Let's consider two groups that submit one job of the same duration each, one using 100% of the CPU capacity and the other only 50%. Because of the chosen metric, the latter is seen as consuming less resources than the former over the same time period and will end up with a higher priority. Then the subsequent inefficient jobs from the second group will be scheduled earlier even though they "waste" CPU time.

The historical way to address this issue was to add a new quota to limit the number of inefficient jobs running, hence adding more complexity to the scheduling. A simpler solution, adopted in September 2017, was to change the metric from *used CPU time* to *actual run time*. This simple modification solved the issue of inefficient jobs without adding an extra complexity to the system. For the other jobs the change is transparent.

4 Conclusion and Future Work

The IN2P3 Computing Center is the largest French academic High Throughput Computing center. Its primary mission is to answer the computing and storage needs of the major international scientific collaborations in the domains of high-energy and astroparticle physics. To manage the execution of millions of individual jobs every month on nearly 35,000 cores, the CC-IN2P3 relied for years on an in-house batch scheduling system, a complex set of admission rules, and quotas on hardware and software resources. However, the ever increasing sizes of

both the infrastructure and workload made the existing system too cumbersome to maintain and put a heavy load on the operation team.

In this experience report, we presented the specificity of the CC-IN2P3, characterized the large HTC workload executed on its resources, and show how complex its operation has become. Then we detailed the work engaged at the end of 2016 to transfer some of the actions done by operators to the job scheduling system with the objective to minimize the "human-in-the-loop" component in scheduling decisions. The proposed modifications that were recently implemented shows preliminary but promising results. However, the work presented in this paper is only the beginning of a long-term activity to change the operation procedures applied to the computing infrastructure of the CC-IN2P3.

Our future work thus includes several directions that we plan to follow. First, we will continue the analysis of the workload to further characterize the jobs and identify leads for improvement. One of the main objective will be to work on a better estimation of the execution time of the jobs. Ideally, we would like to encourage users to provide better estimations of the walltime at submission time as it is classically done on HPC systems. This should both help the job and resource management system to schedule jobs and the operation team to refine the definition of queues. Second, we plan to resort to simulation to assess the impact of potential modifications of the configurations of queues and quotas as proposed in [7,6]. Several tools are available to perform such a simulation study, like Alea [8] or Batsim [2]. The main obstacle is that the "Human-in-the-loop" component that we started to reduce makes it difficult to use logs obtained from the job scheduler and replay them under a different configuration. Indeed, the scheduling decisions taken solely by the job and resource management system may have been altered by operators, without being reflected in the logs. This may lead to interpretation biases. Further reducing these human interventions is thus an essential step in the optimization of the configuration of the job scheduling system. Third, we plan to gather user feedback in a few months to measure the impact of the proposed modifications as perceived by the users. This will give us a complementary point of view on the benefits of this work and may outline new and unforeseen modifications to make. Finally, we would like to give access to contextualized logs to the job scheduling research community. As mentioned before, this requires more work to reduce the human interventions and to be able to indicate in the logs when operators modified the decisions taken by the system. We believe that the large HTC workload processed at the CC-IN2P3 has specific characteristics, detailed in Sect. 2.3 which are very different of what can be found in the Parallel Workloads Archive for instance. This will constitute a new source of interesting problems to solve for the research community, whose feedback would benefit to the operation of the CC-IN2P3.

Acknowledgements

The authors would like to thank the members of the Operation and Applications teams of the CC-IN2P3 for their help in the preparation of this experience report.

References

1. Chapin, S., Cirne, W., Feitelson, D., Patton Jones, J., Leutenegger, S., Schwiegelshohn, U., Smith, W., Talby, D.: Benchmarks and Standards for the Evaluation of Parallel Job Schedulers. In: Proc. of the 5th Workshop on Job Scheduling Strategies for Parallel Processing. pp. 67–90. San Juan, Puerto Rico (Apr 1999). https://doi.org/10.1007/3-540-47954-6
2. Dutot, P.F., Mercier, M., Poquet, M., Richard, O.: Batsim: a Realistic Language-Independent Resources and Jobs Management Systems Simulator. In: Proc. of the 19th and 20th International Workshops on Job Scheduling Strategies for Parallel Processing – JSSPP 2015 and JSSPP 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 10353, pp. 178–197. Springer (2017). https://doi.org/10.1007/978-3-319-61756-5
3. Feitelson, D., Tsafrir, D., Krakov, D.: Experience with using the Parallel Workloads Archive. Journal of Parallel and Distributed Computing **74**(10), 2967–2982 (2014)
4. Jackson, D., Snell, Q., Clement, M.: Core Algorithms of the Maui Scheduler. In: Proc. of the 7th International Workshop on Job Scheduling Strategies for Parallel Processing JSSPP 2001, Revised Papers. Lecture Notes in Computer Science, vol. 2221, pp. 87–102. Springer (2001). https://doi.org/10.1007/3-540-45540-X
5. Kay, J., Lauder, P.: A Fair Share Scheduler. Communications of the ACM **31**(1), 44–55 (Jan 1988)
6. Klusáček, D., Tóth, Š.: On Interactions among Scheduling Policies: Finding Efficient Queue Setup Using High-Resolution Simulations. In: Silva, F., de Castro Dutra, I., Santos Costa, V. (eds.) Proc. of the 20th International Conference on Parallel Processing (Euro-Par 2014). Lecture Notes in Computer Science, vol. 8632, pp. 138–149. Springer (2014). https://doi.org/10.1007/978-3-319-09873-9
7. Klusáček, D., Tóth, Š., Podolníková, G.: Real-Life Experience with Major Reconfiguration of Job Scheduling System. In: Desai, N., Cirne, W. (eds.) Proc. of the 19th and 20th International Workshops on Job Scheduling Strategies for Parallel Processing – JSSPP 2015 and JSSPP 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 10353, pp. 83–101. Springer (2017). https://doi.org/10.1007/978-3-319-61756-5
8. Klusáček, D., Tóth, v., Podolníková, G.: Complex Job Scheduling Simulations with Alea 4. In: Proc. of the 9th EAI International Conference on Simulation Tools and Techniques (Simutools'16). pp. 124–129. ICST, Prague, Czech Republic (2016)
9. Michelotto, M., Alef, M., Iribarren, A., Meinhard, H., Wegner, P., Bly, M., Benelli, G., Brasolin, F., Degaudenzi, H., De Salvo, A., Gable, I., Hirstius, A., Hristov, P.: A comparison of HEP code with SPEC 1 benchmarks on multi-core worker nodes. Journal of Physics: Conference Series **219**(5), 052009 (2010)
10. The ATLAS collaboration: Observation of a New Particle in the Search for the Standard Model Higgs Boson with the ATLAS Detector at the LHC. Physics Letters B **716**(1), 1 – 29 (2012). https://doi.org/10.1016/j.physletb.2012.08.020
11. The CMS collaboration: Observation of a New Boson at a Mass of 125 GeV with the CMS Experiment at the LHC. Physics Letters B **716**(1), 30 – 61 (2012). https://doi.org/10.1016/j.physletb.2012.08.021
12. The IN2P3 /CNRS Computing Center: `http://cc.in2p3.fr/en/`
13. The LIGO Scientific Collaboration and Virgo Collaboration: Observation of Gravitational Waves from a Binary Black Hole Merger. Physical Review Letters **116**, 061102 (Feb 2016). https://doi.org/10.1103/PhysRevLett.116.061102
14. Univa Corporation: Grid Engine. `http://www.univa.com/products/`