Dynamic scheduling using CPU oversubscription in the ALICE Grid

Marta Bertran Ferrer[1,*], *Costin* Grigoras[1,**], and *Rosa M.* Badia[2,***]

[1]CERN, Esplanade des Particules 1, 1211 Geneva 23, Switzerland
[2]Barcelona Supercomputing Center, Plaça Eusebi Güell, 1-3, 08034 Barcelona, Spain

Abstract. The ALICE Grid is designed to perform a realtime comprehensive monitoring of both jobs and execution nodes in order to maintain a continuous and consistent status of the Grid infrastructure. An extensive database of historical data is available and is periodically analyzed to tune the workflows and data management to optimal performance levels. This data, when evaluated in real time, has the power to trigger decisions for efficient resource management of the currently running payloads, for example to enable the execution of a higher volume of work per unit of time. In this article, we consider scenarios in which, through constant interaction with the monitoring agents, a dynamic adaptation of the running workflows is performed. The target resources are memory and CPU with the objective of using them in their entirety and ensuring optimal utilization fairness between executing jobs.
Grid resources are heterogeneous and of different generations, which means that some of them have better hardware characteristics than the minimum required to execute ALICE jobs. Our middleware, JAliEn, works on the basis of having at least 2 GB of RAM allocated per core (allowing up to 8 GB of virtual memory when including swap). Many of the worker nodes have higher memory per core ratios than these basic limits and in terms of available memory they therefore have free resources to accommodate extra jobs. The running jobs may have different behaviors and unequal resource usages depending on their nature. For example, analysis tasks are I/O bound while Monte-Carlo tasks are CPU intensive. Running additional jobs with complementary resource usage patterns on a worker node has a great potential to increase its total efficiency. This paper presents the methodology to exploit the different resource usage profiles by oversubscribing the worker nodes with extra jobs taking into account their CPU resource usage levels and memory capacity.

1 Introduction

The Grid's fleet of worker nodes is widely heterogeneous. So are its batch scheduling and memory allocation policies on the running workflows. Since its inception, the ALICE Grid has always had as a constraining factor the allocation of 2 GB of memory per CPU core, going up to 8 GB when including swap. For jobs with larger memory requirements, there

*e-mail: marta.bertran.ferrer@cern.ch
**e-mail: costin.grigoras@cern.ch
***e-mail: rosa.m.badia@bsc.es

is no guarantee that sites have enough resources to be allocated, and they may eventually be killed by the batch system when the set limits are exceeded. The evolution of computing systems has brought with it an increase in memory capacity. In advanced generations, the minimum memory requirements are far exceeded, resulting in significantly more memory per CPU core.

One of the main objectives of the ALICE Grid is the optimization of the work done per unit of time. That is, the maximisation of the job turnaround, executing the highest amount of workload per unit of time on all the computational resources we have at our disposal. Many site batch queues allocate to ALICE jobs computing slots of a certain limited amount of resources. Grid site computing nodes run parallel workflows of different origins, different natures and different resource usage patterns. The nodes must distribute their computing power and memory capacity among the applications in execution, allocating portions to the execution of our jobs. Because of this limited allocation, when there is more memory available than the initial assumptions, idle resources are left unused. This causes a great detriment to the usage efficiency of computer systems, thus revealing the limitations of their resource allocation methodology among the running workflows.

To avoid this waste of resources, we propose the scenario in which whole nodes are allocated to the execution of our workflows. For a correct management of the resources of whole nodes, it is necessary that the running software makes use of resource partitioning and allocation techniques between the executed processes. This resource allocation methodology is commonly used in supercomputers. As the ALICE Grid already uses some HPC centres for job execution, it has incorporated in its middleware framework, JAliEn, a mechanism that is capable of auto-detecting the resources of the nodes and distributing them among the running jobs in whole node allocation scenarios. The allocation of a whole node brings with it multiple advantages for the scheduling of ALICE jobs, since by knowing the behaviour and execution patterns of the active workflows, a more efficient resource utilisation can be promoted. ALICE payloads behave differently depending on their nature, with different CPU, I/O and memory usage patterns. I/O constrained jobs tend not to use the full computing power of the allocated CPU cores. For this reason, the execution of additional tasks with complementary resource usage patterns has great potential to trade off these inefficiencies.

This paper presents a study of the CPU and memory resource utilisation levels of the Grid worker nodes, which reveals that they could accommodate additional job execution slots if more customised scheduling policies were adopted. It is also presented how the Grid middleware framework has been extended with the functionality of job oversubscription through the constant monitoring of the CPU and memory resources used in whole node scheduling scenarios. It is structured as follows: Section 2 gives an overview of oversubscription strategies used in different computing environments. In Section 3, the current Grid landscape is presented, with a survey of worker node resource utilisation levels for assessing the feasibility of oversubscription policies. Section 4 details the implementation of the workflow. Finally, Section 5 presents a study of its deployment in a production environment.

2 State of the Art

A common practice employed to reduce unused computational resources is oversubscription. Unused resources are to be found given the frequent disparity between the amount of resources that workloads request and how much they actually use during execution [1]. Oversubscribing a resource is understood as offering more than its nominal capacity under the assumption that the share allocated to other workflows will not be fully utilised [2]. This mechanism makes it possible to launch speculative or opportunistic tasks, thereby improving

the levels of utilisation with the ultimate goal of reducing resource idleness while ensuring performance guarantees.

Oversubscribing an environment comes with risks, such as CPU overload or memory limit overruns, that can degrade the performance of running applications and even lead to outages and crashes. Consequently, this can complicate guaranteeing strict service-level agreements (SLAs) [3]. To counteract the effects of CPU overload, running workloads can borrow resources allocated to others running concurrently, be cancelled or preempted, or be migrated to other physical machines with enough available resources [4] [5]. Potential problems are resource mismatches, due to speculative job submission and fluctuations in system utilisation levels, and Quality of Service (QoS) degradation due to increased execution latency [1].

In cloud environments it is a common practice to establish consolidation strategies to minimise the number of servers used and reduce resource fragmentation [6]. For this purpose, it is recommended that physical machines simultaneously host several Virtual Machines (VMs). Their strategic placement is a key factor in minimising resource allocation. To meet performance guarantees, the standard practice is to provision VMs with redundant resources, which results in low levels of resource utilisation. To prevent that, oversubscription strategies can be used to reduce the resource entitlements of each VM according to its QoS requirements [1]. In addition, the implementation of oversubscription practices can lower power consumption and hardware costs [2].

In the literature we can find studies of oversubscription of resources of different nature such as memory [7], whose overloading is particularly devastating to application performance, bandwidth [8] [9], guaranteeing efficient network utilization, and CPU processors [5]. A differentiating factor in oversubscription policies is that they need not result in resource competition, but in resource sharing guaranteeing performance isolation [6].

3 Oversubscription workflow

3.1 Current situation on the Grid

The ALICE Grid constituent sites and systems are largely diverse. The performance characteristics of the individual constituent systems differ due to their architectures and the generations of their components. In addition, access to storage is also a limiting factor when executing tasks requiring input data. This causes job execution performance levels to differ between sites, and also between the constituent worker nodes.

A wide variety of jobs of different nature are executed on the Grid. One of the main differences between jobs of different types is their pattern of resource usage. In this way, the efficiency of a machine's resource usage will also be determined by the nature of the jobs being executed. An example would be I/O bound jobs such as analysis jobs, which perform processing of data obtained from physical events. The processing of data requires network access and therefore the efficiency of the jobs is constrained by the speed at which the data is accessed, which makes strategic placement a key factor in minimising resource allocation.

3.1.1 Grid site resource utilization analysis

A survey of the Grid worker nodes has been undertaken to study their CPU and memory utilisation levels. This study aims to analyse the feasibility of adopting oversubscription policies for hosts using whole node submission (see Section 4.1). Over a 72-hour period, all worker nodes reporting values to the ALICE monitoring system MonALISA have been surveyed. These nodes could potentially be running other third-party workloads that we are

not aware of, but given that ALICE workloads are also heterogeneous, the analyzed scenarios are considered as a representative sample for this case study.

One of the main observations has been that the vast majority (more than 76%) have spare memory resources not used by the running jobs, which are expected to use up to 2 GB of RAM/core (allowed up to 8 GB/core when including swap). That is, their memory per core ratio is higher than the amount required. This constraint is checked by the respective batch queue at each site, at values adapted to the physical resources on each node. In a whole node scheduling situation then, as the jobs are limited to these levels of memory usage, idle resources will be left unused by any of the running workflows.

Another analysis performed consisted of a scan of the levels of memory and CPU usage in the Grid worker nodes, continuously sampling the amount of idle resources to evaluate the potential additional slots. This is done through the accounting of resources that remain idle during a certain time window and could be used for the execution of additional workflows without being preempted. The variation of this time window will have a direct impact on the available resources reported. If the period is shorter, more idle resources will potentially remain available. However, as we increase the study window, they will be more prone to be used by other workflows due to workload fluctuations. As far as idle CPU cores are concerned, we see that the number of nodes that maintain a certain amount of cores available decreases as the amount of cores is increased. That means that many more nodes having one or two free cores are observed compared to the number of nodes which have for instance ten free cores. The same holds for available memory. As the new experiment software for LHC Run 3 [10] deploys multi-core jobs, although legacy single-core jobs are still executed, this study has been performed considering extra slots equipped with 8 CPU cores. Given that each core is assigned 2GB of RAM, a total allocation of 16 GB per slot is assumed.

3.1.2 Potential for additional execution resources

We have done a study that illustrates the potential extra slots we could have on the Grid as a whole, taking into account the time that resources are kept idle on worker nodes. For this purpose, the thresholds for starting and killing payloads were established. In this study the job slots are considered to be assigned eight cores and an idle nine-core boundary was established to account for a new slot: eight cores for the execution itself and one left idle for system operations. A temporary grace interval has been set to decide when to start a new job, thus waiting for the amount of idle cores to stabilise, and to decide when to kill a payload in case the running machine becomes saturated. In order to determine these values, the optimal settings were analysed to find a balance between the preemption rate and the added pressure on the ALICE job management database. The resulting optimal configuration was achieved for an hour-long interval during which cores need to remain idle before starting a new job, and fifteen-minutes-long interval during which the machine exhibits constant saturation before it gets preempted. Since we set an interval of one hour as a temporal decision threshold, this time will always be lost, the idle cores not being used by any payload.

From the collected data, the resulting usable eight-core CPU hours, defined as those idle-time intervals lasting 8 hours or more, have been studied. Fig. 1 shows the number of nodes with 9+ idle cores when ascending usable time intervals. It is worth noting that a large part of the nodes keep the resources idle for the whole measure period, as seen in the highest peak. We find an additional amount of 251k usable eight-core CPU hours that turned out to contain 27.5k extra eight-hour full slots. If we consider the case where all the hosts included in our study had been used in whole node regime for the execution of ALICE workflows, they would have provided in 72 hours a total of 2.5M eight-core CPU hours. From these figures it can be concluded that the amount of additional usable CPU hours represents a 10.2% of the

total, all of which are currently not used, but could be through the adoption of the presented oversubscription mechanism.

On the other hand, node oversubscription also brings with it a caveat: the potential preemption of eight-core jobs due to machine saturation. As seen in Fig. 2, in the measured 72-hour interval, a total of 18.8k eight-core jobs would have been killed due to node CPU usage exceeding the threshold for more than 15 minutes. Jobs running in this regime would run opportunistically, which unavoidably would result in added pressure on the ALICE job management database given the higher preemption rate. Nonetheless, it is considered manageable for the workflow of the experiment.

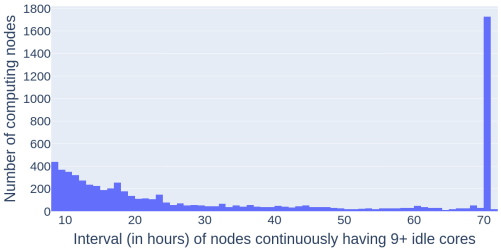

Figure 1. Distribution of the length of potential extra free slots on worker nodes from all Grid sites.

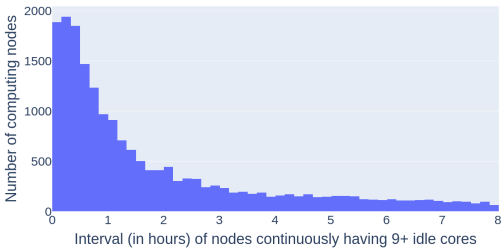

Figure 2. Execution time length distribution of potentially preempted slots within worker nodes from all Grid sites.

4 Implementing oversubscription on the ALICE Grid

4.1 Whole node allocation and partitioning

The executing sites can allocate resources to the ALICE Grid jobs in different granularities. However, this study considers the allocation of a subset of the resources of the executing machines, and the allocation of whole nodes, i.e. all the resources that make up a worker node. The allocation of a whole node brings with it a great potential for JAliEn, the ALICE Grid middleware framework, to perform a tailored allocation of the jobs to be executed. Given the prior knowledge of the run patterns of the different types of jobs running on the Grid, the selection of the jobs executed in parallel can be optimised to maximise the utilisation levels of the machines.

The methodology followed for job allocation is based on late binding. The worker nodes send match requests to the ALICE Central Services in which they announce their capabilities, such as available memory, remaining non-allocated cores, their residual lifetime or other limitations imposed on the site. In the Central Services the allocation of the jobs is performed, whose descriptions are sent in response to the received requests. Once the execution of the jobs is finished, the resources they were occupying are freed and returned to the central pool for future allocations. The management of all the resources of a node allows its custom partitioning among the jobs in execution, without setting restrictions or rigid allocations. Having this ability to distribute resources according to the time of need makes it possible to tune how many jobs are executed in parallel.

4.1.1 Resource usage monitoring

Each of the sites that make up the ALICE Grid has a monitoring agent that maintains a communication channel with the Central Services through which it periodically sends monitoring

information on the executing payloads and the resources used. This information can be the key to make scheduling decisions that enhance the usage patterns of the computing resources in order to achieve high levels of efficiency. In terms of monitoring the worker nodes, this includes CPU, memory, disk, and network usage rates. From this information, JAliEn is able to sketch a detailed description of the machine.

4.2 The oversubscription workflow

In whole node scheduling scenarios we want to use the available resources in the most efficient way. Having a prior knowledge of the usage patterns of the jobs that are regularly executed on the Grid allows us to make predictions of their behaviour. More precisely, it is known that the analysis tasks are I/O intensive and that they do not use their allocated CPU slots at all times. Given this situation, idle portions of CPU can be gathered and used to execute computing-intensive jobs. Among the main payload types executed on the Grid, Monte-Carlo simulations are best suited to the desired job profile. As executing CPU-intensive tasks does not add pressure on I/O, no competition will be added for the latter resources.

The node oversubscription is a configurable flag in the ALICE central (LDAP) configuration of the target site. The amount of cores to consider for the oversubscription pool is determined based on the assessment of their resource ratios. When scheduling the Job Agents that manage the execution of the payloads, the memory per CPU core ratio is computed and it is decided whether cores can be oversubscribed. For this to be possible, it must be guaranteed that this ratio is higher than 2 GB of RAM per core and 8 GB when also considering swap. The amount of CPU cores available for oversubscription is calculated as follows:

$$\max(0, \min(\mathrm{RAM}/2, (\mathrm{RAM} + \mathrm{swap})/8) - \mathrm{physicalCores}) \tag{1}$$

The CPU utilisation levels of the worker nodes concerned are continuously monitored. The monitoring agent keeps track of the idle CPU and uses it to make scheduling decisions. When it detects that the idle CPU levels are above a certain threshold and a minimum of 2 GB of free memory is available, the opportunity to run a job in oversubscription mode is triggered. A grace period is established to ensure that these values are not ephemeral and, if conditions permit, a request is sent to Central Services advertising the resources in the oversubscription pool. In this request, the nature of the job to be received is also detailed, in this case Monte-Carlo simulations, to promote the complementary resource patterns. The job to be executed is set to a low priority to minimise interference with the jobs that are being executed using the resources in the regular pool.

Since we need to have all the machine's resources available to enable oversubscription decisions, one of the necessary conditions is to have whole node allocations configured on the sites. Several sites on the Grid are configured in this way, showing good results in terms of resource utilisation efficiency.

4.3 Rescheduling policies for the oversubscribed jobs

Continuous monitoring of CPU and memory utilisation also serves to detect states where the machine becomes saturated and jobs running in the regular resource pool are negatively affected. When alarming utilisation levels are detected, oversubscribed jobs are preempted. The preemption of oversubscribed jobs and their rescheduling differs from the regular rescheduling of failed jobs, for which the resubmission counter is permitted to increase until it reaches a certain threshold. In the case of an oversubscribed job that is interrupted due to lack of available resources, the job's resubmission counter is not incremented, thus the job is rescheduled

without penalty. Another policy applied with regard to job preemption is the consideration of its running time to decide the order in which such jobs are cancelled. Priority is given to continue executing the workflows that have been running the longest and will therefore have the highest probability to complete, while preempting those with a shorter lifetime.

5 Deployment of the oversubscription workflow in a production environment

The presented implementation has been evaluated by deploying the oversubscription-enabled framework on machines with different hardware characteristics. As described in Section 4, the maximum number of jobs executed in oversubscribed mode is given by the underlying system architecture. The results obtained on two different machines will be used to illustrate the added power of oversubscription. In both cases, the amount of oversubscribed cores is bounded by their swap capacity.

- Machine 1: Server with 64 cores Intel Xeon Gold 6226R at 2.90 GHz with 500 GB of RAM and 8 GB of swap. Using Eq. 1, 3 cores were added to the oversubscription pool.
- Machine 2: Server with 32 cores Intel Xeon Gold 6244 at 3.60GHz with 756 GB of RAM and 8 GB of swap. Oversubscription pool of 11 cores.

One-day-long slots have been run on both machines, comparing regular and oversubscription-enabled job executions. Single-core analysis jobs with the same input data have been executed on both machines and both scenarios, homogenising the testbed for a consistent evaluation of results. As many tasks as CPU cores of the systems have been executed in parallel. As oversubscribed jobs, single-core CPU-intensive MonteCarlo simulations have been run. The metrics defined to evaluate the results are, on the one hand, the efficiency of the analysis jobs (regular executions) and, on the other hand, the average efficiency of CPU resource utilisation of the machine over the one-day-long slot. The machine's CPU utilisation is computed as the sum of individual jobs' CPU times over the 24h interval.

The obtained results are as follows:

- **Machine 1**

- Regular job average execution efficiency in *non-oversubscribed* regime: 55.1%
- Regular job average execution efficiency in *oversubscribed* regime: 53.5%
- Machine utilization efficiency ratio (oversubscribed / non-oversubscribed): 1.03

- **Machine 2**

CPU usage of this machine in both scenarios is shown in Fig. 3 and 4.

- Regular job average execution efficiency in *non-oversubscribed* regime: 63.3%
- Regular job average execution efficiency in *oversubscribed* regime: 58.2%
- Machine utilization efficiency ratio (oversubscribed / non-oversubscribed): 1.28

From the conducted experiments and the metrics obtained, we can draw some conclusions. First, the increase in efficiency observed in the machines is consistent with the amount of extra jobs executed. Moreover, the drops in job efficiency are also coherent between the two machines. As a main result of this study, we highlight the pronounced increase in machine utilisation levels due to oversubscription, with the largest improvement of 28%, observed on the machine with the highest number of oversubscribed jobs.

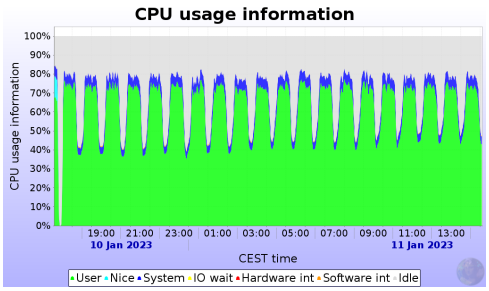

Figure 3. CPU usage of Machine 2 running with the regular framework.

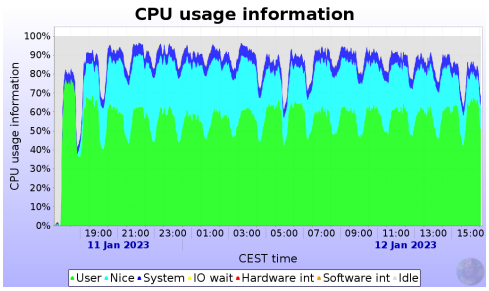

Figure 4. CPU usage of Machine 2 running with the oversubscription-enabled framework.

6 Conclusion

This paper presents a study of the utilisation levels of the machines that make up the ALICE Grid, which reveals that part of their resources remain idle for considerably long periods of time. Resource utilisation patterns of concurrent tasks of different natures allow utilisation efficiencies to be increased. With sufficient memory resources and only in cases where whole-node scheduling is enabled, oversubscription of CPU cores in the worker nodes has been found to lead to pronounced increases in utilisation efficiency. However, adding parallel workloads increases the pressure on resources, resulting in decreased individual job execution efficiency. Nonetheless, in Grid environments, where resource heterogeneity limits the enforcement of strict SLAs, the observed increase in machine utilisation compensates for the decrease in individual job execution efficiency.

References

[1] R. Yang, C. Hu, X. Sun, P. Garraghan, T. Wo, Z. Wen, H. Peng, J. Xu, C. Li, IEEE Transactions on Parallel and Distributed Systems **31** (2020)

[2] R. Householder, S. Arnold, R. Green, *On cloud-based oversubscription* (2014)

[3] R. Ghosh, V.K. Naik, *Biting Off Safely More Than You Can Chew: Predictive Analytics for Resource Over-Commit in IaaS Cloud*, in *IEEE 5th CLOUD* (2012)

[4] N. Jain, I. Menache, J.S. Naor, B. Shepherd, *Topology-aware vm migration in bandwidth oversubscribed datacenter networks*, in *International Colloquium on Automata, Languages, and Programming* (Springer, 2012)

[5] X. Zhang, Z.Y. Shae, S. Zheng, H. Jamjoom, *Virtual machine migration in an over-committed cloud*, in *IEEE Network Operations and Management Symposium* (2012)

[6] Y. Liu, *A Consolidation Strategy Supporting Resources Oversubscription in Cloud Computing*, in *IEEE 3rd CSCloud* (2016)

[7] D. Williams, H. Jamjoom, Y.H. Liu, H. Weatherspoon, *Overdriver: Handling memory overload in an oversubscribed cloud* (ACM New York, NY, USA, 2011), Vol. 46

[8] Z. Guo, J. Duan, Y. Yang, *Oversubscription Bounded Multicast Scheduling in Fat-Tree Data Center Networks*, in *IEEE 27th IPDPS* (2013)

[9] D. Breitgand, A. Epstein, *Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds*, in *IEEE INFOCOM* (2012)

[10] P. Buncic, M. Krzewicki, P. Vande Vyvre, *Technical Design Report for the Upgrade of the Online-Offline Computing System. CERN-LHCC-2015-006, ALICE-TDR-019* (2015)