Supervised job preemption methodology for controlled memory consumption of jobs running in the ALICE Grid

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

CERN[1], Barcelona Supercomputing Center[2]

E-mail: marta.bertran.ferrer@cern.ch

Abstract. The Grid resources of the ALICE experiment range considerably in terms of memory capacity, CPU cores, and resource management. Memory allocation for scheduled jobs depends on the hardware constraints of the executing machines, system configurations, and batch system policies. The ALICE O2 software framework introduces multi-core tasks where deployed processes share resources. To accommodate these new use cases, most Grid sites provide ALICE with multi-core slots of customisable number of cores. The Grid middleware manages the resources within a slot, sub-partitioning and distributing them among allocated jobs. This allows for parallel execution of jobs with different requirements within the same resource-sharing slot. From the scheduling system's perspective, this job set is treated as a single unit for resource usage accounting. Overconsumption by any job can lead to the entire slot being killed, terminating all co-executing ALICE jobs. To prevent this and promote job completion with reasonable resource usage, the Grid middleware should implement targeted preemption of top-consuming jobs when overall consumption approaches the system's termination threshold.

This article analyzes site resource limiting procedures, including termination policies and memory thresholds, and the design of the ALICE Grid middleware framework's methodology for targeted preemption. Preemption decisions are made in real time, considering various factors of running payloads, weighted according to experiment priorities, to maximize efficiency and successful task completion.

1 Introduction

The Grid of the ALICE experiment executes tasks with different resource requirements and use patterns. These tasks perform reconstruction, analysis or simulation of physics events for which they tend to require large amounts of memory. The data acquisition of the experiment in Run 3 is trigger-less, collecting data in the form of continuous data-frames [1]. In order to avoid multiple allocations of such large memory segments, the O2 framework allocates shared segments which many processes map and use. This keeps the memory per CPU core under control but still can reach or pass above the WLCG [2] standard of 2GB of memory per core.

The heterogeneous Grid resources show disparate behaviours in the treatment of jobs whose memory usage exceeds the initial assumptions. Beyond the WLCG standard, if the executing nodes have more memory resources available, they can allow the running jobs to allocate more. This heterogeneity of sites, both in terms of hardware resources, configurations and node utilisation levels, makes the memory resources to which jobs have access neither predetermined nor very predictable, as factors such as the workload of the machines tend to fluctuate over

time. One of the options for dealing with site allocation constraints is the requesting of a larger amount of cores than strictly needed, which leads to a proportional raise in the allocated memory. However, this increase in the requested cores might entail a lower usage efficiency on the allocated CPU cores.

Grid worker nodes have limited memory resources for two main reasons. The first of them is the physical hardware characteristics of the machines. On memory-bound systems, the provisioning of extra memory resources to the tasks in need might be unfeasible. Secondly, sites can also apply configurations that explicitly limit the allocated memory per core. This constraining can be done in different ways, and at different levels. From the batch queue configuration, limits can be applied that encompass the full slot. The ALICE Grid middleware (JAliEn) can, on top of that, divide the resources of this slot between the co-running workflows, therefore the limits are applied to the whole set of jobs executed in a multi-core slot. Consequently, independent jobs running in a shared slot may get killed when co-executed with memory over-consuming tasks, their overall consumption being higher than the allocated memory by the slot.

This article is organised as follows: Section 2 presents how other HEP experiments manage controlled memory usage on Grid resources. Section 3 introduces the current situation in the ALICE Grid in terms of job memory utilisation and the site policies for memory management. Section 4 describes the structure and operation of the module developed for job orchestration in high memory pressure scenarios. Upcoming work is introduced in Section 5.

2 State of the art

The memory profiling of applications working with particle physics data is complex. This is mainly because the events processed in the different tasks are widely diverse, containing a varying number of particles which results in different memory requirements [3]. Furthermore, an accurate setting of the needed resources requires careful benchmarking by using estimates that are large enough to avoid frequent failures in heterogeneous execution scenarios, but not so large that might decrease resource usage efficiency through being partially left unused [4]. In addition, the late-binding strategies used in Grid environments requires up-front requesting of the required resources, which can result in allocation inefficiencies.

The ATLAS experiment describes in [3] its strategy for a refined and finer-grained brokering by approximating the memory needs of jobs with similar requirements. The initial estimate is made with a subset of jobs, and is fine-tuned as more tasks are executed.

Some of the approaches are based on using historical data from previous executions to predict resource usage [5]. In the IceCube experiment, machine learning models have been used to predict peak memory usage of its tasks with the goal of performing memory allocations that minimise resource wastage [4]. The task requirements set by user estimates are being tuned accordingly, improving the quality of memory allocations.

3 Current Grid status

Figure 1 shows the global maximum and average maximum job memory used per CPU core at the different Grid sites during a three-day period, in which a comparable Grid job mix is executed on all of them. Among sites, the node architectures, configurations and observed behaviours differ widely. As a safety net, JAliEn monitors the memory consumed per job and preempts its execution once it exceeds four times its guaranteed allocation, that is 8GB. Nevertheless this protection is often not enough, as Grid jobs fail due to memory allocation issues in the executing nodes, proving that their termination mechanisms act before our over-consuming job preemption workflow.

Grid sites mainly present two kinds of configurations:

- Sites with very sharp termination thresholds, such as *Vienna*. These sites impose constraints by defining hard limits, which are set to custom values and are mainly enforced through settings in the different batch systems (SLURM [6] and HTCondor [7]) or by using cgroups

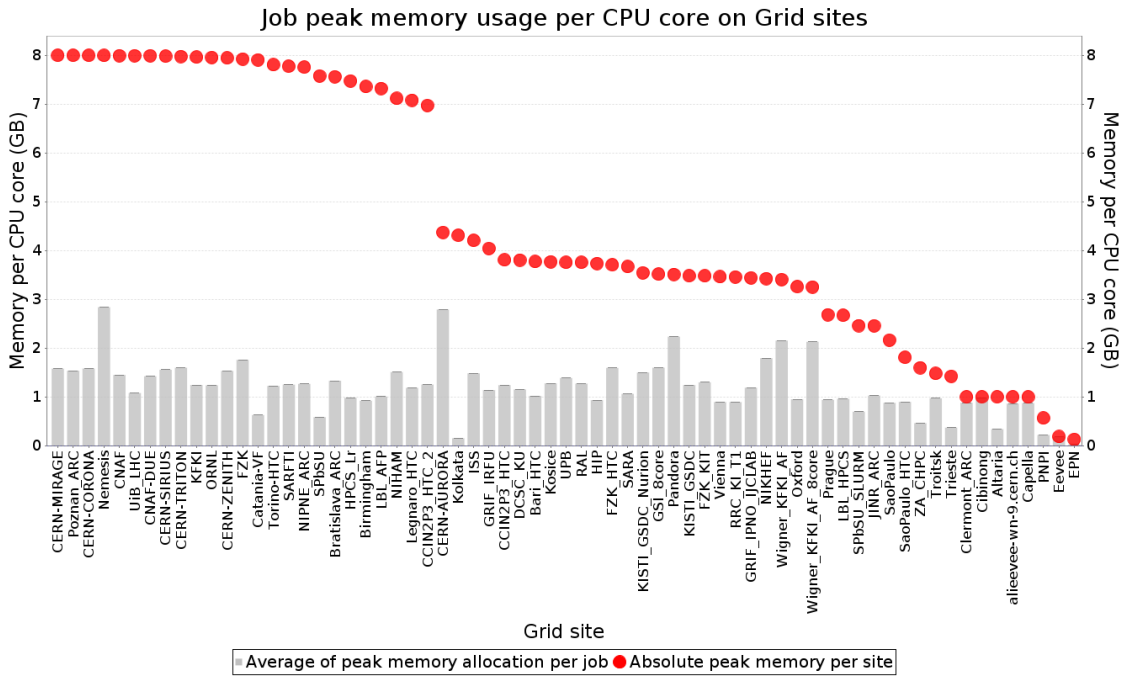

Figure 1: Average of executed jobs peak memories and absolute job peak memory utilisation per CPU core on the different Grid sites

[8]. In addition to the definition of static hard limits, sites can enforce policies defined through custom statements set in the batch queue configuration files such as the HTCondor ClassAd. The behaviour of these sites is highly predictable, and they are ideal candidates for controlled job preemption.

- Sites with soft limits, or no explicitly configured limit, whose termination thresholds vary greatly depending on the conditions of the execution environment, such as machine load or the amount of resources available to the system itself. Examples would be *LBL* and *ISS* sites. In these cases, the monitoring of the executing system conditions must be exhaustive, as this is what will trigger a potential slot preemption.

4 Memory controller module for controlled job preemption

A memory control module has been implemented in the ALICE Grid framework by means of which the memory limits configured in the slot are discovered, taking into account the multiple possible set-ups. This module constantly monitors the memory consumption of deployed workflows and detects alarming memory utilisation close to the set threshold. In situations where the available memory is low, it proceeds to target preemption of one of the running jobs following specific policies, freeing memory space and allowing co-executing jobs to continue their execution.

4.1 Investigation of the limits

The fetching of limits configured in the executing machine is the first step of the module's workflow. This is done to identify the limits, as well as the severity to which they are going to be applied, i.e. hard or soft limits. The limit inspection on the executing machines consists of several checks on configuration files. The first of these is the cgroup to which the job has been assigned (v1 or v2), as well as its hard memory limit. Then, in case of being a system using HTCondor, its job ClassAd is parsed from which the *PeriodicRemove* expression is extracted. HTCondor ClassAds describe jobs and machines by means of a set of uniquely named expressions defined by custom expressions using a C-like syntax. One of the parameters included in the *PeriodicRemove* among ALICE Grid sites is the *ResidentSetSize*, which is composed of an arithmetic inequality

operation. In these cases, the evaluation of the expression discloses memory consumption close to exhaustion. Finally, the limits set in the */proc* filesystem are checked. Besides being set at the operating system level, they can also be set by the batch system. In case of having no predefined limits, i.e. that the result of the limit inspection is null, the memory resource occupancy of the executing node as a whole is considered to evaluate the threshold. Note that the setting of soft limits implies treating it in the same way as if none were set, as freely running jobs.

4.2 Memory consumption supervision and evaluation

The proposed solution to this problem involves a continuous supervision of the memory usage of the assigned batch queue slot in order to anticipate the termination decisions of the worker nodes when memory limits are exceeded. The live supervision of the used memory resources enables the individual payloads' overconsumption to be detected and, according to the priorities of their executions, to perform a targeted termination, thus guaranteeing that the rest of the workflows can continue their executions smoothly. The memory consumption of the workflows is monitored by periodically parsing */proc* files of the payload processes every of 5 seconds.

Given that the system's capacity to react when alarming levels of consumption are perceived is not immediate, the taking of preemption decisions must be anticipated. For this purpose, memory margins are defined relative to the set threshold taking into account the number of CPU cores that comprise the slot. If no explicit limit has been set, the occupancy levels of the whole execution machine are monitored. In this case, margins are also established at the level of the entire system to guarantee a reasonable time of reaction.

As mentioned in the previous section, the minimal amount of memory/core on any Grid host is 2GB RAM. Both the physical availability and the allowance of swap memory usage varies between Grid hosts. If jobs need more memory than available, they may start swapping out. Limits on memory may or may not include swap memory usage. In cases where it is not included, an extra level of uncertainty is added derived from its availability and system usage allowance.

The module for targeted preemption must evaluate the preemption decision in real time. This eliminates the possibility of contacting the Grid central services to take decisions based on historical records. Multiple parameters can be taken into account for making the preemption decision, such as the memory consumed, the length of time during which the job has been running or the user who submitted the job.

4.3 Evaluation of the module

Prior to taking real preemption decisions, the module's decision-making has been evaluated. To this end, it has been added to the middleware framework to keep track of the eventual preemptions. This is done by recording the preemption conditions in a central database without actually taking any action to the running workflows. Furthermore, the module is able to detect eventual job terminations by the kernel out-of-memory (OOM) or the Batch Queue policy by inspecting the kernel logs. When this happens, the timestamp of this action is also recorded in the database, together with the conditions in which our system would have triggered its preemption.

The analysed data is collected from the 68 Grid sites during a period of nineteen days (May 10th to May 29th 2024). The middleware uses the same decision criteria for preemption decisions at all sites. Nevertheless, very different figures are observed for the number of jobs that would eventually be preempted, or the ratio of preemption decisions versus jobs that are actually terminated by the system. However, the differences are not only to be found in the execution hosts and their policies. Another very important factor is the nature of jobs and execution patterns. There are jobs that are more memory intensive than others, and with more and less sharply defined memory usage patterns. Performing a memory profiling prior to the executions is complex since in particle physics, jobs of similar nature can process highly heterogeneous events [3]. In the absence of such figures, the implemented tool is limited to considering only the memory conditions of the execution environment when slot memory is close to exhaustion.

Site	Slot type	Total preemptions	Anticipated terminations	Saved co-executors	Misterminated jobs	Saved jobs
LBL	Whole-node	67	16	490	51	90%
Vienna	8-core	197	43	312	122	72%
ISS	8-core	770	407	2354	407	85%

Table 1: Evaluation of preemptions at three sample sites with different configurations. The impact is evaluated on a per-slot basis, considering the effect on co-executors.

Results from three sites with different resource allocation policies (8-core slot and whole node allocation) are detailed in Table 1. The first one is the LBL site. It allocates whole-node slots, so the memory bounds are solely defined by the hardware characteristics of the systems. To measure the impact of the module operation, we study how many of the slots in which a job would have been preempted are affected by a system job termination. If the preemption action anticipates a terminated job within a short time window, it is considered that the rest of the co-executors would be exempted from OOM action. As for the slot's memory consumption, its average value at the time of preemption is dependent on the executing system.

In the sites allocating 8-core slots the preemption actions would impact less co-executing jobs given the amount of cores per slot. In the site Vienna, even though the slots have a hard memory limit of 33.5GB set by cgroups, the average slot memory consumption at preemption is 32 GB, as a margin is set to the threshold prior to taking the decision. At the ISS site the slot memory bounds are defined by the hardware of the systems, as there is no memory limiter policy configured. The average value of physical RAM per CPU core is 4 GB, which is equivalent to 32 GB per 8-core slot. Yet the average value of memory at the time of preemption is 35 GB, showing how the implemented module does not limit itself to confining jobs to their theoretical allocated portion, but is able to let jobs expand and use the memory unused in the machine.

5 Further work

The memory control module will be further refined as its results are assessed. Its rollout for the target preemption of over-consuming jobs will be progressive on Grid sites, along with constant monitoring of its operation and controlled site error rates. The algorithm for selecting the specific job to preempt within the slot co-executors will be improved. For preemption decision-making to result in the minimum number of well-behaved killed jobs, all ways of defining OOM killing policies need to be anticipated. In order to standardise it between sites, the utilisation of cgroups will be promoted. Additional tools will be added for requesting the estimated memory that jobs require at submission time, allowing for improved resource usage planning and accounting.

6 Conclusion

One of the biggest challenges on the ALICE experiment Grid is to accommodate the co-execution of tasks with very heterogeneous resource utilisation patterns. From an execution infrastructure management point of view, tasks that make reasonable use of resources must be facilitated to complete successfully. This article describes the memory controller module that has been incorporated into the ALICE Grid middleware framework, to anticipate the process killing and batch slot termination decisions made by the executing system when exceeding the set thresholds. These thresholds can be defined through different custom methodologies by sites, with the module being able to infer them in all cases. In multi-core environments, execution resources can be shared between jobs that demand a certain share, being the combined over-utilisation the trigger for slot terminations. In scenarios where memory usage is alarming, targeted preemption of over-consuming jobs reduces memory pressure and promotes the successful completion of co-executors. A consistent job preemption decision-making enables new enhanced job and resource control mechanisms that can be used to design a Grid strategy that evolves in line with the needs of the experiment.

7 Bibliography

[1] ALICE Collaboration. Upgrade of the online-offline computing system. Technical report, CERN-LHCC-2015-004, 2015.

[2] WLCG - Worldwide LHC Computing Grid. `https://wlcg.web.cern.ch/`. Accessed: 2024-07-15.

[3] AC Forti et al. Memory handling in the atlas submission system from job definition to sites limits. In *Journal of Physics: Conference Series*, volume 898, page 052004. IOP Publishing, 2017.

[4] Carl Witt et al. Learning low-wastage memory allocations for scientific workflows at icecube. In *2019 International Conference on High Performance Computing & Simulation (HPCS)*, pages 233–240. IEEE, 2019.

[5] Andréa Matsunaga and José AB Fortes. On the use of machine learning to predict the time and resources consumed by applications. In *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 495–504. IEEE, 2010.

[6] Slurm Workload Manager. `https://slurm.schedmd.com`. Accessed: 2024-07-09.

[7] HTCondor Manual. `https://htcondor.readthedocs.io/en/latest/`. Accessed: 2024-07-09.

[8] cgroups - Linux control groups. `https://man7.org/linux/man-pages/man7/cgroups.7.html`. Accessed: 2024-07-09.