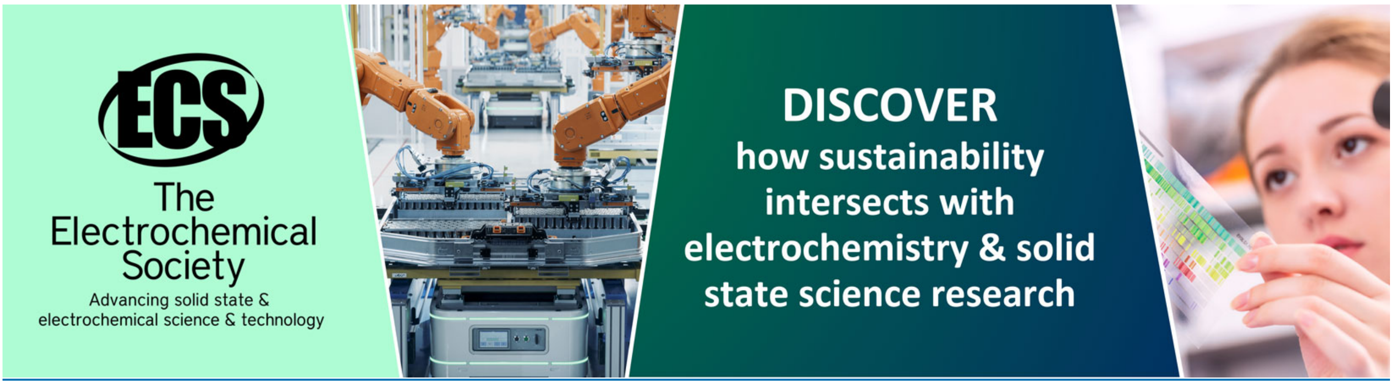
Journal of Physics: Conference Series

PAPER • OPEN ACCESS

Memory handling in the ATLAS submission system from job definition to sites limits

To cite this article: A C Forti *et al* 2017 *J. Phys.: Conf. Ser.* **898** 052004


ECS
The Electrochemical Society
Advancing solid state & electrochemical science & technology
DISCOVER how sustainability intersects with electrochemistry & solid state science research

Memory handling in the ATLAS submission system from job definition to sites limits

A C Forti[1], R Walker[2], T Maeno[3], P Love[4], N Rauschmayr[5], A Filipcic[6], A Di Girolamo[5]

[1] School of Physics and Astronomy, University of Manchester, Oxford Road, Manchester, M13 9PL, UK.
[2] Ludwig-Maximilians-Universitat, Munchen, Fakultat fur Physik Schellingstrasse 4, 80799 Munich, Germany.
[3] Brookhaven National Laboratory, Upton, NY, 11973, USA.
[4] University of Lancaster, Lancaster, UK.
[5] CERN (European Laboratory for Particle Physics), Rue de Geneve 23 CH 1211 Geneva, Switzerland.
[6] Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia.

E-mail: `Alessandra.Forti@cern.ch`

Abstract. In the past few years the increased luminosity of the LHC, changes in the linux kernel and a move to a 64bit architecture have affected the ATLAS jobs memory usage and the ATLAS workload management system had to be adapted to be more flexible and pass memory parameters to the batch systems, which in the past wasn't a necessity. This paper describes the steps required to add the capability to better handle memory requirements, included the review of how each component definition and parametrization of the memory is mapped to the other components, and what changes had to be applied to make the submission chain work. These changes go from the definition of tasks and the way tasks memory requirements are set using scout jobs, through the new memory tool developed to do that, to how these values are used by the submission component of the system and how the jobs are treated by the sites through the CEs, batch systems and ultimately the kernel.

1. Introduction

The ATLAS [1] workload management system is a pilot system that wasn't originally designed to pass fine grained job requirements to the batch systems. In particular for memory the requirements were set to request 4GB virtual memory, defined as 2GB RAM + 2GB swap for every job. However in the past few years several changes have happened in the operating system kernel and in the applications that make such a definition of memory to use for requesting slots obsolete. ATLAS has also introduced the new PRODSYS2 [2] workload management which has a more flexible system to evaluate the memory requirements and to submit to appropriate queues. The need to review how memory is handled during submission stemmed in particular from the introduction of 64bit multicore workloads and the increased memory requirements of some of the single core applications.

How to handle jobs based on memory requirements in ATLAS is a problem that stems from the interaction between the ATLAS pilot system and the batch systems. Batch systems work on the *early binding* principle. The jobs arrive with a number of requirements such as number of cores, memory, disk space, length of the job and the batch system does its best to find and allocate the resources to run that job and avoid wasting the resources. The requirements are used both for internal scheduling and for limiting excessive usage beyond the required resources. If the job doesn't have any requirement the default of the batch system queue where the job lands will be used to do this. The pilot system on the other hand works on the *late binding* principle. It was developed to compensate for the fragility of the grid at the beginning because the pilots could check the suitability of the node before downloading the real payload. The problem with this method is that the pilot not knowing what the payload is at submission time presents to the batch system always the same requirements independently from the payload needs. This was good enough when the payloads were uniform single core jobs requesting less than 2GB memory per job. Nowadays we have a a variety of payloads requirements (single core, multicore, low memory, high memory, short, long) and sending pilots with uniform requirements is not efficient anymore either because the payload needs more resources than requested and gets killed or because it needs much less and there is a waste. In particular the memory is a precious resource. Sites tend to buy a certain amount of memory per job slot and using more than that means giving up other resources such as the number of usable processors. It is clear that the pilot submission had to adapt to differentiate the requests.

2. Changes affecting ATLAS jobs memory requirements

In the past few years there have been three major changes that affected the ATLAS job memory requirements:

- Higher luminosity in Run 2 (2015-2018) meant bigger events and this, in computing terms, means longer event processing time and bigger memory usage. Some data formats more than doubled their size since 2011, for example ESD increased from 1.1MB per event to 2.4MB.
- In 2014 we also started to move the code from 32bit to 64bit, this helped with the longer processing times being 25% faster but it also increased the memory usage by 25-50%.
- Multicore jobs were introduced to further reduce the longer processing times and the increased memory usage. The latter is achieved in AthenaMP [3] by using the COW (Copy-on-write) resource management technique by which a memory page is shared until one of the threads needs to write on it, the OS then creates a modifiable page for that thread. This is much more economical than copying the whole memory tree for that thread. The use of multicore has reduced the Athena memory footprint per process by 30-50%, however it has introduced an extra degree of freedom in the job memory management at brokering and batch system handling levels because the overall job memory is much higher. It is not a coincidence that the work reported in this paper started as part of the multicore deployment work carried out by WLCG in 2014 [4]

3. Memory definitions, description and handling evolution

In the following the causes for the memory requirements evolution and the effects on the submission chain will be described. Before trying to implement a solution we had to look at how the memory was mapped through the system [5] as reported in table 1.

3.1. Changes to the Linux kernel

The three changes reported in the previous section emphasised another problem that single core jobs, with smaller memory on a 32bit system could ignore. The meaning of virtual memory and how accurately the standard OS systems describe the memory.

Table 1. Parameters through the system.

Experiments	**Corecount**	**rss**	**rss+swap**	**vmem**
ATLAS old	corecount	maxmemory	maxmemory	maxmemory
ATLAS new	corecount	maxrss	maxrss+maxswap	-
Computing Element	**Corecount**	**rss**	**rss+swap**	**vmem**
ARC-CE	(count=corecount) (countpernode=corecount)	memory	-	memory
CREAM-CE Glue1	JDL: CpuNumber= corecount; WholeNodes=false; SMPGranularity= corecount	GlueHostMain MemoryRAMSize	GlueHostMain MemoryVirtualSize	GlueHostMain MemoryVirtualSize
CREAM-CE Glue2	JDL: CpuNumber= corecount; WholeNodes=false; SMPGranularity= corecount	GLUE2Computing ShareMaxMainMemory	GLUE2Computing ShareMaxVirtualMemory	GLUE2Computing ShareMaxVirtualMemory
HTCONDOR-CE	xcount	maxMemory	-	-
Batch Sys Parameters	**Corecount**	**rss**	**rss+swap**	**vmem**
Torque-maui	ppn	mem	-	vmem
*GE	-pe	s_rss	-	s_vmem
UGE 8.2.0	-pe	m_mem_free	h_vmem	s_vmem
HTCondor	RequestCpus	RequestMemory	No default (Recipe)	No default (Recipe)
SLURM	ntasks,nodes	mem-per-cpu	-	No option
Batch Sys to OS	**cgroups support**	**rss**	**rss+swap**	**vmem**
Torque/maui	No	No	No	RLIMIT_AS
Torque/MOAB or PBSPro >=6.0.0	yes	yes	yes	RLIMIT_AS
*GE	No	No	No	RLIMIT_AS
UGE >=8.2.0	yes	yes	yes	RLIMIT_AS
HTCondor	yes	yes	yes >8.3.1	-
SLURM	yes	yes	-	-
LSF >=9.1.1	yes	yes	yes	RLIMIT_AS

Virtual memory is an address space that can be bigger or smaller than the physical RAM. On older systems, when the memory was much smaller, the virtual memory used to map to RAM+swap. The RAM was the physical memory and RAM+swap was the virtual memory. The ATLAS memory requirements are expressed in in terms of virtual memory and RAM, i.e. 4GB virtual memory of which at least 2GB are RAM and the rest can be swap. Older batch systems also still treat the virtual memory as meaning RAM+swap. However with the introduction of 64bit the address space has just become much bigger and this mapping is no longer adequate.

A second problem that became visible when multicore was introduced is how the shared memory is reported. Older OS tools getting the information from older parts of the kernel report the shared memory incorrectly. We have seen that one of the reasons to introduce multicore jobs was to reduce the amount of memory used by each process by using shared memory, but if the tools used by the system administrators and by the batch systems to handle the memory report the wrong values the gain is only nominal and not effective. For example, a job may use less resources than the batch system limits but the batch system thinks it has used more and kills the job or doesn't schedule other jobs on that node because it thinks all the RAM is being used.

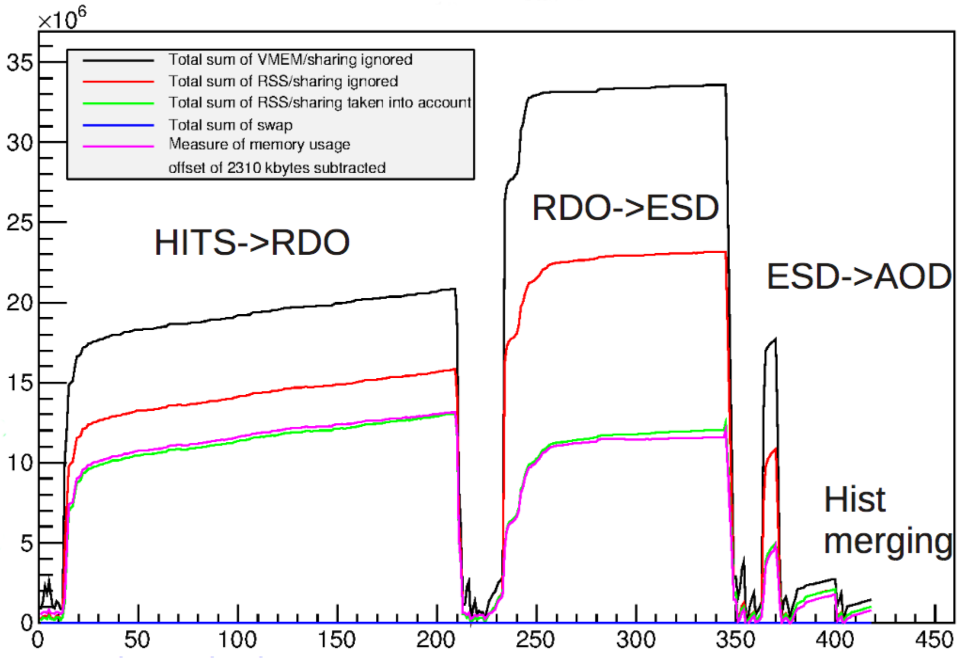


Figure 1. Measures of different type memory used by AthenaMP jobs.

We have established that the virtual memory is the address space, not a portion of physical memory. But the terms to describe the RAM have also evolved. There are two terms to describe the RAM used by a process: RSS (Resident Set Size) which is the portion of memory used by a process resident in RAM, as opposed to the portion resident on the swap area or the file system; and PSS (Proportional Set Size) which is the portion of RAM occupied by a process composed by the private memory of the process plus the proportion of RAM shared with other processes. In the following section we will see how these two are reported by different tools.

The size of the different type of memories can be appreciated in Fig.1 where for different ATLAS jobs RAM, swap and virtual memory are shown.

3.2. Operating System tools

Classic tools like `ps` and `top` don't report correctly the various memory parts anymore. In particular the RSS reported by these tools contains multiple counting i.e. the shared memory of a multicore job is counted multiple times. `ulimit` is another tool that doesn't work anymore, it used to be able to kill a process on a RSS limit and on a virtual memory limit, but now it can only use the latter while the former is ignored. If `ulimit` is used to kill a job there is no flexibility. Older batch systems unfortunately still make use of these tools as we will explain later.

cgroups [6] are a kernel feature introduced to limit and account for resources used by processes. It offers a unified interface for all the resources one may want to limit (CPU, memory, disk I/O, network, etc.). The physical memory reported by cgroups is still called RSS but in this case it doesn't contain multiple counting of the shared memory. cgroups resource handling is more sophisticated than `ulimit` because it is integrated with the kernel and can consider all available resources on the node before killing a job, i.e. it will try to allocate more resources than requested if possible.

If a site doesn't use cgroups because the batch system is too old to support it, it is still possible to at least measure the memory correctly for each process using /proc/<PID>/smaps. The smaps file contains the values of the physical memory without multiple counting, the equivalent of the cgroups RSS, but in this case is called PSS; the classic RSS with multiple counting as seen by `ps`, `top` and `ulimit`; the swap values; and finally the virtual memory values. Surprisingly there are practically no OS tools making use of the information contained in smaps yet.

To summarise the correct values to use when one talks about RAM used by a process or a job are either cgroups RSS or, if the site doesn't use cgroups, smaps PSS. The RSS and virtual memory reported by classic tools and smaps are respectively wrong and meaningless. swap is correctly measured only in cgroups but not in smaps when shared pages are swapped out of memory - the effect though can be ignored in cases when the swap is much smaller than the PSS size as it happens for ATLAS multicore jobs.

3.3. Batch Systems

The batch systems are tools to manage resources and they will be heavily affected by what the kernel reports and does. The ATLAS sites use a variety of batch systems, so for each we had to investigate what kernel tools it used and how it behaved. From this point of view the batch systems can be categorised in two groups: those that use cgroups and those that still use `ulimit`. The batch systems that can use cgroups to monitor and limit the resources can set up both hard and soft memory limits. The soft limit allows the kernel to decide if a job can keep on using extra RAM. This is a big advantage for ATLAS jobs which often exceed the memory requirements but only for few minutes. In this way the memory management passes from the batch system to the kernel, that can take much better decisions because it has the overall view of the resource usage. Sites administrators however usually set also a hard limit which will kill the job to contain run away jobs, i.e. jobs that really use excessive memory, usually 2 or 3 times the one requested, for a long period of time. Both RAM and swap can be limited, but with this mechanism the use of swap is usually drastically reduced.

The batch systems that still use `ulimit` instead do not have any of this flexibility. They can only kill on the size of the address space. Even if they could limit the job memory requirements using the RSS values, they'd still see the wrong one, i.e. the one with the shared memory counted multiple times. This is problematic for ATLAS because 80% of the sites still use old batch systems. There is a shift to newer batch systems integrated with cgroups. Many sites have invested a lot of time in tuning and monitoring their current batch system and consequently the inertia to replace the current with the new system is significant.

3.4. Computing Elements

Computing Elements are the middleware between the ATLAS submission system and the batch systems. Like for the batch systems there are many flavours and they don't behave all in the same way. Each type of CE supports several batch systems as a back-end. One of their functions is to map the job requirements usually expressed in the CE terms to the right batch system attributes. So the transmission on the requirements is CE flavour dependent and had also to be mapped. The developers of the CREAM-CE, one of the CEs, intentionally left it to the system administrators to write the code to do the translation with the result that for many sites it is missing and there is no standard behaviour at the sites where it was written. Part of the effort was then to harmonise these scripts and use matching parameters in the component of the ATLAS system that submits the pilots.

3.5. PandaQueues

The final piece in the chain is the site description in the ATLAS brokering system. The parameters the pilot passes to the batch systems are taken from the PandaQueues, which are the sites resources description units in the ATLAS system. The brokering system brokers to PandaQueues with a set of resource requirements and these requirements are then passed to the CE which translates them into underlying batch system requirements. Until the changes described in this paper were applied, ATLAS also used the approximation virtual memory = RAM+swap and that is what it sent to the sites. The same parameter was however used internally for brokering decisions with the meaning of RAM creating a lot of confusion. To add to the confusion the value translated into different things depending on the the CE/batch system combination: at some sites the value was effectively mapped to a RAM value, at others it was mapped to virtual memory.

So the first action was to rename the memory parameter from a generic maxmemory to a better defined maxrss and adjust the values in the PandaQueues to represent exactly that. To use maxrss with correct values sites can setup lo/hi memory PandaQueues. These PandaQueues can be mapped to one single batch queue with large values: there is no need for sites to enable more queues because now pilots will pass the memory values to the batch system and the jobs will be brokered correctly to sites that can handle it.

4. Memory monitoring

So far we have described the submission chain and how the numbers propagate. Memory requirements for jobs are usually approximations even for more standard applications. For particle physics, profiling applications accurately [7] is even more difficult because the events processed in a single job are not constant, they contain a different number of particles and require different amounts of memory to be processed. This, as well as the late binding characteristic of the pilot system, is one of the reasons the LHC experiments never really tried to pass more accurate resource requirements when a first approximation was sufficient. But now with the new type of jobs we need to know how they behave more systematically. For this purpose CERN-IT wrote a memory monitor tool that can be run within each pilot and take measures of the memory used by each job. The tool sums the smaps numbers for each job subprocess every few minutes and records these values in a log file. There are several advantages to do it this way:

- smaps is a feature automatically available in most linux distributions - certainly in the Red Hat derivatives most WLCG sites use - since kernel 2.6.16 was introduced. It is memory data stored in the /proc/<PID>/smaps files for each process.
- Results in the log files means that when there is a problem it is now easy to compare what the job actually used with what the site batch system saw and identify the fault accordingly.
- Results can be used for brokering (see below).
- Results can be used to kill jobs that significantly exceed the requirements at those sites that don't have cgroups and can't effectively protect the WNs. The threshold has been put to twice the memory requested and if the job exceeds that the pilot will kill it. The advantage of this is that the pilot can exit gracefully, collect all the logs and send back an appropriate error. Both analysis and production jobs receive the same treatment (i.e. they get killed if they exceed the requirements), but production jobs will be automatically rescheduled to higher memory queues, while the analysis jobs will not. In practice though only analysis jobs behave abnormally in this way.
- Aggregate results can be used for systematic studies and refine the jobs initial requirements. For example Fig.2 represents the differences between PSS/swap/VMEM similar to Fig.1 but on an aggregate of all ATLAS multicore jobs over the past 6 months. This was really useful

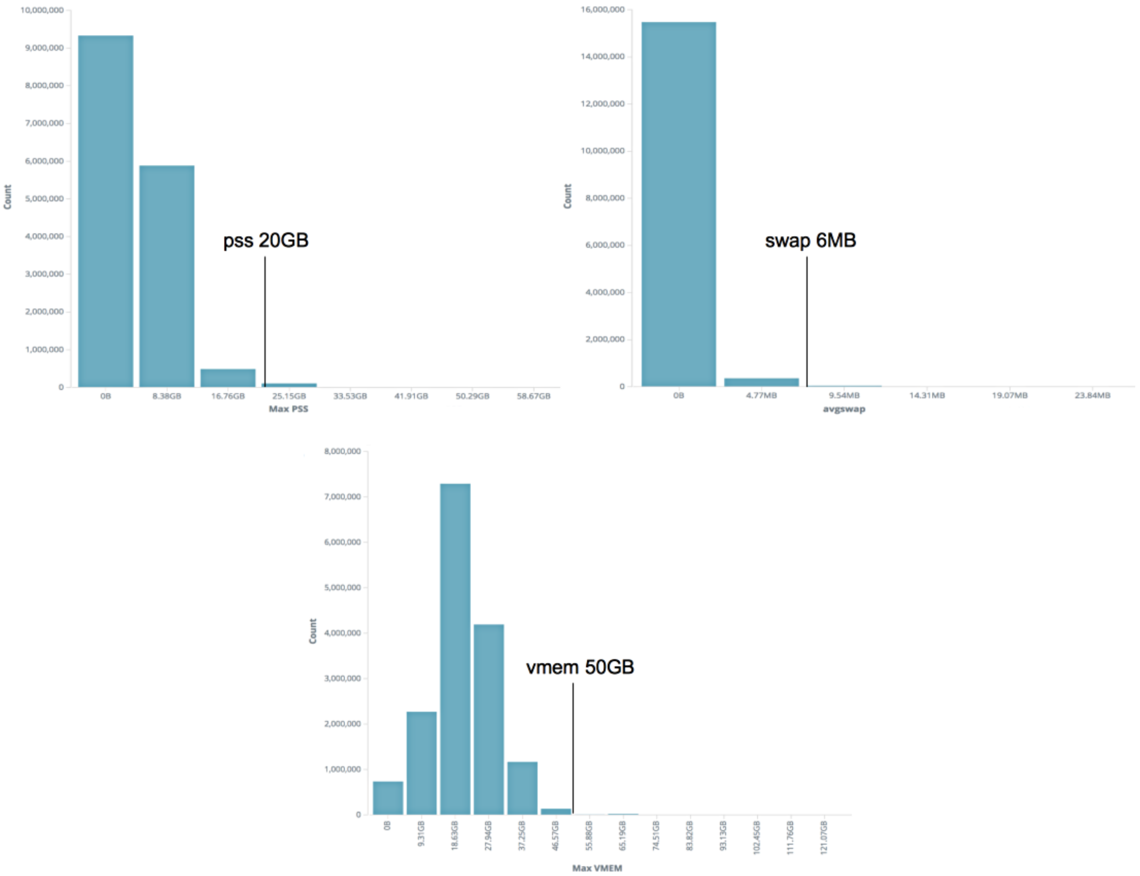


Figure 2. Aggregate measures of all multicore jobs PSS/swap/virtual memory for the past 6 months. The virtual memory is several times larger than the sum of RAM+swap on all the jobs not only on single profile ones. Same holds true for single core jobs.

to demonstrate that jobs 95% of the time don't exceed the RSS requirements and that virtual memory doesn't correlate at all with RAM+swap.

5. ATLAS brokering

ATLAS jobs are organised in tasks. Jobs belonging to a task have similar requirements. To assess these requirements initial jobs called scouts are sent to the larger computing sites which usually have larger resources. The memory monitor output from the first 5 jobs is used to assess the task requirements. The biggest PSS value of those jobs will become the PSS requirement for all the jobs of that task which will be brokered to PandaQueues by comparing the task PSS with the PandaQueue maxrss. The PandaQueue maxrss value is in turn used as parameter to pass to the CE and ultimately to the batch system and it is the maximum PSS the job can use. It can happen that some jobs can use in excess of the scouts value. If they are production jobs and they are killed by the batch system, the brokering system increases their requirements and brokers them to PandaQueues with a higher maxrss value. If this happens to analysis jobs instead the users will have to manually resubmit them.

6. Results

In the ATLAS system to keep track of jobs the pilot contacts the brokering system every 30 minutes to say it is still alive. Lost heartbeat is a catchall error message for when the server loses contact with a job for 6 hours. There are many causes for this error, in fact anything that kills the pilot without warning like a crash or a loss of connectivity, but a major cause is the batch system killing the job for exceeding the memory requirements. This error became the overwhelming cause of wall time loss since the memory requirements increased as explained in previous sections. However since the progressive introduction of memory handling both in ATLAS and at sites there has been a parallel reduction of wall time wasted due to Lost Heartbeat. In 2015 the wall time wasted was 15% of all walltime and 32% of it was caused by Lost Heartbeat; in 2016 this was reduced to 24% out of 12.5% as shown in Fig.3.

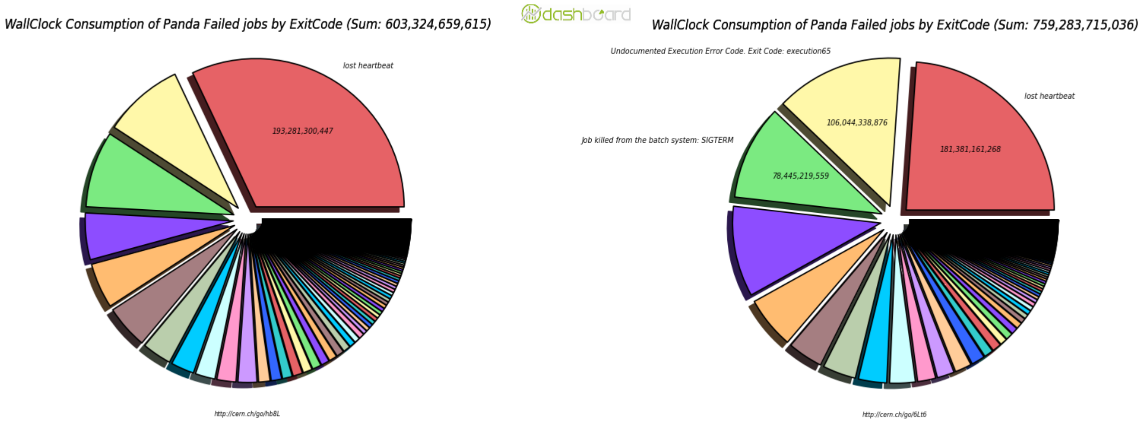


Figure 3. On the left is the wall time wasted in 2015 and on the right the wall time wasted in 2016. The lost heartbeat portion was 32% in 2015 and is about 24% for 2016

7. Conclusions

One of the longest standing requests from sites is now satisfied. ATLAS can better distribute the workload because brokering is done on more accurate measurements of the jobs real requirements. The system was designed to support older batch systems that cannot support cgroups and don't handle memory correctly anymore thanks to the pilot now killing on a sensible RSS value. The introduction of memory handling has reduced the weight of the *lost heartbeat* error overall partly because less jobs are killed, partly because the errors are now better reported.

References

[1] The ATLAS Collaboration 2008, The ATLAS Experiment at the CERN Large Hadron Collider , JINST 3 102 S08003 [doi:10.1088/1748-0221/3/08/S08003]

[2] M Borodin, K De, J Garcia, Navarro, D Golubkov, A Klimentov, T Maeno and A Vaniachine 2015 J. Phys.: Conf. Ser. 664 062005

[3] P Calafiura, C Leggett, R Seuster, V Tsulaia and P Van Gemmeren 2015 J. Phys.: Conf. Ser. 664 072050

[4] A Forti, A Pérez-Calero Yzquierdo, T Hartmann, M Alef, A Lahi, J Templon, S Dal Pra, M Gila, S Skipsey, C Acosta-Silva, A Filipcic, R Walker, C J Walker, D Traynor and S Gadrat 2015 J. Phys.: Conf. Ser. 664 062016

[5] Passing parameters project, 2015, `https://twiki.cern.ch/twiki/bin/view/LCG/BSPassingParameters`

[6] Red Hat Enterprise Linux 7 Resource Management Guide `https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/pdf/Resource_Management_Guide/Red_Hat_Enterprise_Linux-7-Resource_Management_Guide-en-US.pdf`

[7] N. Rauschmayr Memory Profiling and Cgroup Tests `https://twiki.cern.ch/twiki/pub/ITSDC/ProfAndOptExperimentsApps/ATLAS_SC_Week.pdf`