Minutes for the second Multicore TF meeting.

Attendance: Alessandra and Antonio plus Thomas Hartmann, Carles Acosta, Sam Skipsey, Stefano Dal Pra, Jeff Templon, Rod, D. P. Traynor, Andrea Chierici, Manfred Alef Di Qing, Jerome Belleman, Alexey Sedov. Gareth Roy Jose Hernandez, Jose Flix, Chris Walker, Raul (brunel), Simone Campana, Andrej Filipcic.

Slides presented by Rod Walker on recent ATLAS multicore activities. Comments follow the presentation and are summarized below:

Slide 3: some discussion on what the dynamic provisioning exactly translates into, and how the request for resources is made depending on how the resources are partitioned locally. The technical details on how this request is actually performed should be included. Some discussion on the actual definition of a "slot" is. For those sites with static partitioning some differences are noted on the multicore jobs requests being made as 1 slot x 8 cores versus 8 slots x 1 core.

Slide 4: Concerning the use of resources by ATLAS, they foresee to move the whole production activities to multicore, whereas the analysis jobs are expected to remain single core only. There is no plan on sending them through multicore pilots, as ATLAS considers this a scheduling problem, therefore a site business. In contrast, CMS could in principle send both single core and multicore jobs via multicore pilots.

Slide 5: The autumn milestone for multicore deployment, originally in the CMS schedule, is shared by ATLAS as well.

Slide 6: In addition to the slide content itself, the discussion focuses on Maui. First point is whether it is still supported or not by EGI or by the developers. Maui seems not to be supported anymore neither by EGI nor by Adaptive Computing, the latest version on the AC site being 3 years old. This is something that will be discussed at one of the next GDB meetings. Secondly, whether it would be really capable of handling scheduling as well as other systems. However, even if free open source alternatives exist, perhaps some sites using torque/maui would resist to move unless strong evidence of its replacement is shown.

Slide 7: Accounting of the allocated and used resources. With respect to resources used by multicore jobs, batch systems provide total CPU time, walltime and the number of cores used, therefore it should be easy to account for the total usage and its efficiency, as all the needed ingredients are already available. However they are still not properly used by WLCG on their reports, which only records cpu time and walltime. While the walltime report hasn't changed, the cputime is already multiplied by the number of cores in the batch system logs. So the efficiency should be redefined from cputime/walltime to be calculated correctly taking the factor 8 into account. The TF should investigate what apel is doing.

The accounting of the empty cores which would result from the nodes being drained to accommodate multicore jobs however is more difficult in general, although simpler if there is only one VO running. ATLAS proposes that it is accounted on the site, not the experiment, since it derives from the scheduling, which in their vision is a problem of the sites. An similar example is provided by the high memory jobs, where one job may take up the mem resources of the entire node, which is thus charged completely on the VO. However, it was agreed the high memory jobs and the multicore jobs cases are not equivalent. The former is unavoidable without buying more memory and also affects only a portion of T1 resources, the other affects everyone and should be minimized by good scheduling.

---

The following presentation is given by Jeff, with some observations on running ATLAS multicore jobs from Nikhef experience.

The problem observed is the very long time it takes for a multicore job to run, being in the queue for up to 20 hours, while the actual running time is generally 2 hours or below. This is because of the mixture of single core jobs (of varied running times) and multicore jobs in a common queue prevents maui from early scheduling multicore jobs. This is improved by sending short single core jobs, such as analysis running for up

to about 4 hours, to a different queue with a short limit on walltime. This way maui can do backfilling.

A comment on the allocation policy defined for these tests is made. The results may differ if the maui policy is set to "minresources", which tends to fill the machines completely, instead of "cpuload", which tends to spread the load across the whole farm. This of course depends on the level of usage of the farm: if it is completely full it will probably deliver the same results, because the allocation of resources happen according to when jobs end and how long they last. However it should still be tested in a dynamic environment with a good mixture of single core and multicore jobs where multiple jobs of both types are ending in a short interval, to see if the actual proportion of running jobs and their waiting times fluctuate or a dynamic equilibrium is obtained.

An additional source of fluctuation is the presence of other VOs. As the fair share of the experiment is exhausted it becomes increasingly more difficult to schedule multicore jobs, until the fair share is recovered and a new cycle begins.

A question is raised from a member of CNAF on whether some type of fair share values could be implemented based on users and groups in addition to VOs, which could help in stabilizing the system as it performs the resource allocation. Per group scheduling can be used to separate analysis from production jobs, and used to increase production priority and fairshare to avoid analysis occupying all the freed job slots

With respect to the memory usage of the multicore jobs, it is quite stable at around 2GB per used core. However, the "per job virtual memory" management by maui seems to be a bit complex. The key lies in whether the multicore jobs are "true multicore", therefore sharing memory among all cores, or multi process. In this second case maui still multiplies (8x4)x8, therefore it thinks the requirement is for 256 GB instead of 32. Since ATLAS jobs are actually multiprocess, requesting N independent cores from maui should solve the confusion, so that the system correctly understands the request. ATLAS can use ulimit to limit pvmem in the pilot. if the main process exceeds the memory limit set in the pilot the job gets killed by the pilot shell itself. Many sites have chosen this way to limit the memory also for single core jobs.

---

Finally some slides with questions and comments from KIT. The major concern is that in order to to perform backfilling at the site, an accurate (or at least "good enough") estimation of jobs running time is essential. This is however a complicated problem in itself, as the jobs would need to have a reference of the CPU power of the machines on which they landed.

In KIT view, if the backfilling process needed to schedule multicore jobs causes gaps (empty cores), then they should be accounted on the VO. This is emphasized by the different interests of the VOs and the sites summarized in slide 2.

The discussion focuses then on whether doing the backfilling internally to some long lived multicore pilots could actually solve the problem of not having accurate running times. This is being studied by CMS, however worries are expressed on whether or not this constitutes an additional layer of complexity. This proposal will be explained in the coming meeting by CMS.

The matter could actually center on the evaluation of the pros and cons of doing part of the backfilling logic "internally", by the pilots carrying the payload jobs or "externally", by the local batch system. However, backfilling done at batch system level would still be needed as in the first model, in order to take care of the variety of jobs from other VOs supported by shared sites.

---