HSF
HEP Software Foundation

WLCG
Worldwide LHC Computing Grid

Site perspective

Handling of high memory jobs. Whole-node scheduling
Standard 16-core instead of 8-core slots

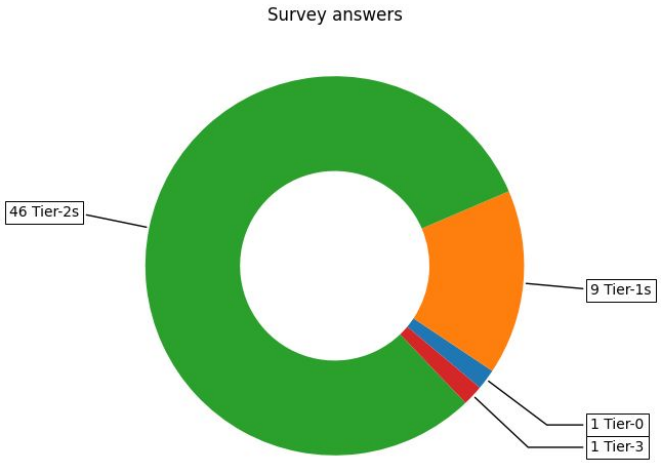
J. Flix (PIC/CIEMAT) for WLCG Ops. Coord.

WLCG/HSF Workshop at DESY Hamburg, May 13-17 2024

PIC
port d'informació
científica

Site Questionnaire

A **site questionnaire** in preparation for the WLCG/HSF Workshop was sent by WLCG Ops Coord. on 9th April 2024 → Crucial input for preparing and making decisions



57 sites answered - Thanks!

[as received by 6th May 2024]

This talk offers an (maybe incomplete, my fault!) **overview of the responses**

High-memory payloads

1. Support for Running High-Memory Jobs:

- Many sites express readiness and capability to run high-memory jobs, with varying configurations ranging from 4GB/core to 8GB/core compute nodes
- Sites are willing to accommodate such jobs as long as the memory requirements are explicitly specified and properly accounted for
- Some sites already support high-memory payloads for specific experiments

2. Accounting Concerns:

- A recurring concern is accounting for resources, particularly regarding idle cores and the CPU utilization for these tasks
- Suggestions are made to adapt WLCG accounting to reflect the usage of both CPU and Memory
- Proposal to adjust billing or charging methods to more accurately match resource utilization, potentially by charging for idle cores according to the respective VOs
  (e.g. N standard 'units' of mem → CPU walltime xN)

High-memory payloads

3. Resource Allocation and Management:

- Sites express a preference for full utilization of cores to avoid idle resources
- Some suggest adjustments such as turning off Hyper-Threading (e.g. to get more memory per job) or procuring dedicated hardware for high-memory queues to better manage resource allocation
- Efficient job brokering and scheduling are seen as crucial for maximizing resource utilization

4. Technical Considerations:

- Technical considerations include the need for adjustments to batch systems, accounting strategies, and routing decisions to accommodate high-memory jobs efficiently
- The distribution of memory/core varies across sites, influencing their capacity to handle high-memory payloads

High-memory payloads

5. Discussion and Collaboration:

- There's an emphasis on the need for discussion, collaboration, and clarification regarding resource requirements, accounting practices, and the impact on CPU pledges if significant high-memory jobs run in WLCG
- Sites are open to adapting their configurations and policies to support diverse job requirements, but they seek clarity and collaboration in this process
    - Some sites suggest isolating high-memory jobs in specific queues to streamline resource allocation and placement
- Some express concerns about the impact on CPU efficiency and the need for clear delineation and reporting of resource usage

Whole-node scheduling

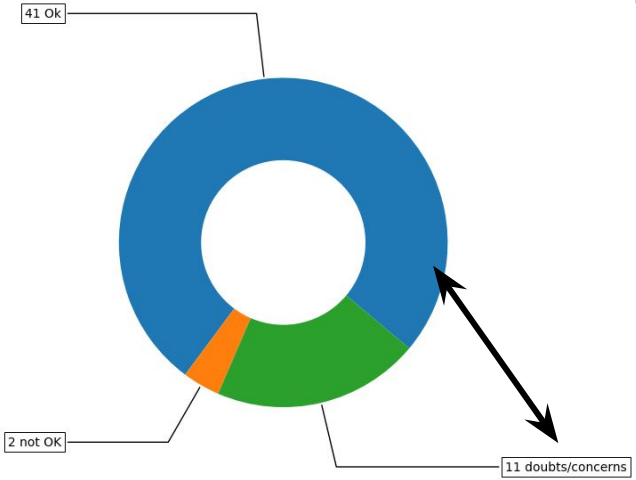
1. **Resource Utilization and Efficiency:** Concerns about the efficiency of resource utilization, especially with nodes having a high number of cores (not many compute nodes available!). Some sites express doubts about the efficiency of whole-node scheduling as compared to running smaller jobs

2. **Complexity and Management:** Complexity in managing whole-node scheduling is highlighted, especially regarding draining processes, queue management, and ensuring efficient turnover of resources, in particular for multi-VO sites. Whole-node scheduling may offer predictability but could limit flexibility for running smaller tasks

3. **VO Requirements:** The needs and preferences of different VOs vary, with some favoring whole-node scheduling for specific workloads, while others are cautious due to potential challenges in resource sharing and scheduling → HPCs suited for whole-node and some specific workloads?

4. **Effort for Implementation and Testing:** Sites welcome to explore whole-node scheduling but emphasize the need to further understand implications before final implementations

16-core vs. 8-core jobs?



1. **Concerns about CPU architecture compatibility:** Concerns about the compatibility of the CPU architecture with Non-Uniform Memory Access (NUMA) architecture in certain CPU generations: the cores may be distributed over multiple NUMA nodes, affecting job performances?
2. **Efficiency considerations:** Several sites emphasize the importance of ensuring that the CPU efficiency of running 16-core jobs is at least equal to that of 8-core jobs
3. **Runtime and scheduling considerations:** Many sites highlight concerns about cluster entropy, scheduling inefficiencies, and backfill scheduling challenges associated with using 16-core slots
4. **Compatibility with existing infrastructure:** challenges related to handling servers with non-divisible by 16 core counts (e.g., 56 cores), and the potential impact on scheduling and resource utilization. Need co-existence for single-core or smaller multi-core jobs for efficient nodes usage

16-core vs. 8-core jobs?

5. **Need for further study:** A few sites express the preference for 8-core slots, citing issues such as job fragmentation, scheduling efficiency, and the ability to utilize resources optimally. Other sites indicate a willingness to consider the change but emphasize the need for further study to determine the most efficient core count for job allocation

Scheduling multicore workload on shared multipurpose clusters

To cite this article: J A Templon *et al* 2015 *J. Phys.: Conf. Ser.* **664** 052038

https://iopscience.iop.org/article/10.1088/1742-6596/664/5/052038

Conclusions

General support for accommodating **high-memory payloads**. Concerns about resource management, accounting practices, and ensuring efficient resource utilization across different job types. In-depth discussions are seen as essential for addressing these challenges effectively

While there are potential benefits to **whole-node scheduling** for specific use cases, such as high-memory jobs or certain VOs, there are also significant challenges related to resource management, efficiency, and compatibility with existing infrastructure and workflows

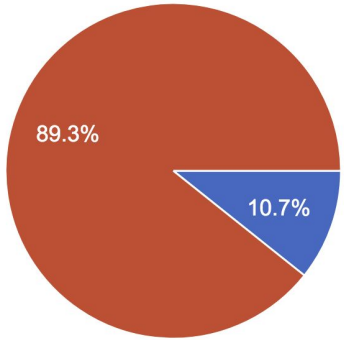
While some sites are open to the idea of using **16-core slots for multi-core jobs**, there are significant concerns regarding efficiency, scheduling, compatibility, and resource utilization that would need to be addressed before implementing such a change

[backup]

ARM resources

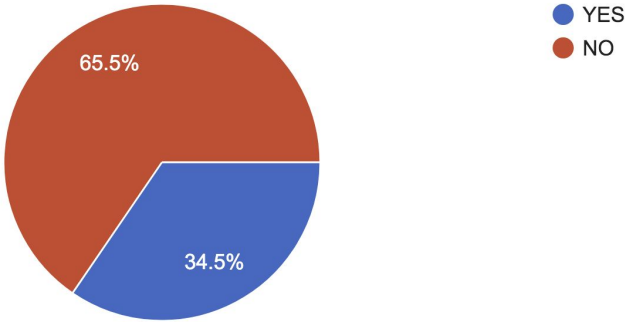
Does your site already host ARM resources?

56 responses



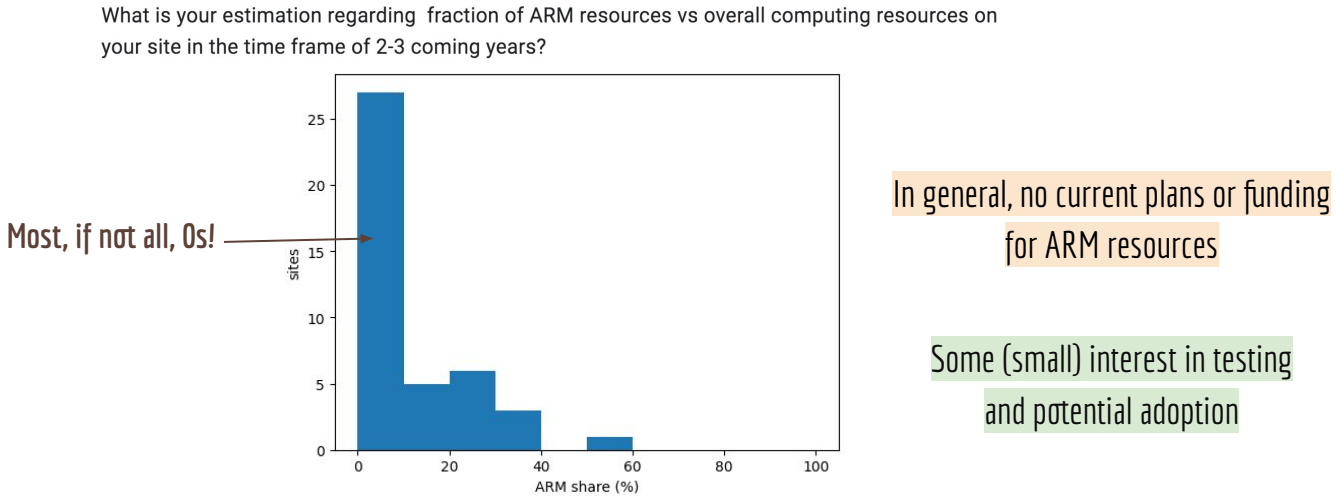
Does your site intend to purchase ARM resources in the near future?

55 responses



ARM resources

1. **Current Adoption Levels:** The majority of sites currently have zero or very low (less than 1%) ARM resources. Some have around 10% ARM resources, with expectations for growth over time
2. **Future Plans:** some express openness to the idea of purchasing ARM resources in the future, contingent on various factors such as experiment adoption, cost-effectiveness, and energy efficiency. Others mention a cautious approach, waiting for experimental software validation or clear use cases before committing to ARM adoption, but open to experiment with ARM at some point



In general, no current plans or funding for ARM resources

Some (small) interest in testing and potential adoption

ARM resources

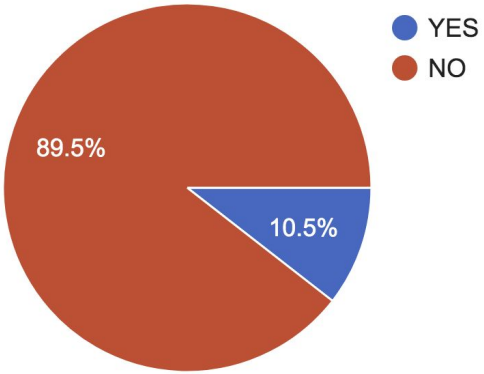
3. Factors Influencing Adoption:

- Readiness of experiments to utilize ARM resources
- Comprehensive comparisons with AMD processors are necessary to fully evaluate their efficiency and effectiveness
- Compatibility with existing cooling infrastructure (such as liquid cooling systems)
- Limited funding and space constraints are mentioned as challenges in expanding ARM adoption

4. Other comments:

- Limited benefits of ARM resources for specific workloads, especially in physics, when compared to modern x86 CPUs with large vector extensions?
- Compatibility issues, administrative hurdles, complexity in resource management, immaturity of datacenter-grade ARM servers, and/or limited vendor support
- Mix of interest, skepticism, and practical considerations regarding the adoption of ARM resources, with some seeing potential benefits in terms of power efficiency and cost savings

Running x86_64-v1 or even older?



1. **Amount? What is the plan for retirement?** Minimal remaining resources. Many sites have plans to retire x86_64-v1 and (even) older resources, at most by end of 2024 (partial retirement or gradual replacement processes). Older resources to be replaced with newer ones that support x86_64-v3 or higher

Conclusions

General support for accommodating **high-memory payloads**. Concerns about resource management, accounting practices, and ensuring efficient resource utilization across different job types. In-depth discussions are seen as essential for addressing these challenges effectively

While there are potential benefits to **whole-node scheduling** for specific use cases, such as high-memory jobs or certain VOs, there are also significant challenges related to resource management, efficiency, and compatibility with existing infrastructure and workflows

While some sites are open to the idea of using **16-core slots for multi-core jobs**, there are significant concerns regarding efficiency, scheduling, compatibility, and resource utilization that would need to be addressed before implementing such a change

Mix of interest, skepticism, and practical considerations regarding the adoption of **ARM resources**, with some seeing potential benefits in terms of power efficiency and cost savings, while others are cautious due to various challenges and current uncertainties

10% of the sites have plans in place to retire their older **x86_64-v1** resources by end 2024. The majority have already decommissioned them or never had such resources

Complete answers

[backup]

High-memory payloads

Some experiments have an interest in running high-memory payloads at the T1 and T2 sites with the understanding that they keep the average RSS/core under what is physically available (usually 2GB/core). It implies that some cores may be left idle, to allow their memory allocation to be used by those high-memory jobs. Could you, please, share our opinion regarding this requirement from the site perspective?

We offer 4GB/core and 8GB/core nodes.

We can run such jobs. The only requirement is that the relevant resources be explicitly specified (per job) to be passed to the local batch manager.

This seems reasonable, of course usage of HPCs or other resources should also be considered, which often have a larger RSS/core ratio (but of course different I/O).

I believe that all LRMS and CE used have abilities to both request and bill the resources required by jobs (In particular for ARC-CE + slurm, for a X mem/cpu the billing will do automatic translation for the resources meaning that if 2X memory was requested, 2CPUs will be automatically billed.) So, it actually does not matter for the site what users/experiment are requesting as the used resources will be correspondingly bill/accounted for and reported.

We run 4GB/HT as standard, we have 8 GPU servers with 1TB per node (10GB/HT). we have no problem running higher memory jobs as long as we were not penalised in the accounting (i.e. if they needed 16GB for a job they would request 3 job slots (3*4GB) and the accounting would see this as 3 * CPU)

Most of our nodes are 3 GB / core to 4 GB / core, as we never trusted the assertion that 2 GB / core was still okay. If jobs requiring significantly more than this became commonplace, we would have to look at how we allocate resources. A fresh approach to accounting may also be needed.

This is already a problem today, a lot of jobs already die if we impose a limit of 2GB/core. To avoid this we are already imposing a maximum of 4GB/core.

Apart from making accounting more complicated, pledged resources are pledged to the experiments to do what they want on.

We can fulfill the requirement either by only setting the memory value per job (then some cores would be left idle) or by also letting those jobs run on high-memory nodes. This second option can minimize the number of idle cores.

All our nodes have ~ 4 GB/core average RAM. Instead of wasting cores, job brokering should be improved to make most efficient use of the most suitable resources.

High-memory payloads

All our servers have 4GB/thread memory, we can handle higher memory jobs easily. Also as Core counts rise it becomes easier to schedule a mix of workloads on the farm allowing high memory jobs to be run alongside normal jobs lowering average memory usage. If VO need higher memory jobs we have no problem providing it and don't mind even if a small number of cores are idle.

We provide 4GB/core so hope this is not an issue at our site.

90% of worker nodes have 4GB/core. Wouldn't be a problem for us.

Mix under physical average fine. Needs accounting if cores left idle.

We do not currently have any large memory nodes planned for this. We could support that if the HTCondor-CE - SLURM job submission interface requests the appropriate amount of memory for jobs, though.

fundamental problem at the site level is that without time estimates/requirements of pilots no proper scheduling is possible. With VOs withing for 96h max runtimes efficient scheduling becomes even more unreasonable with pushing for 16core and highmem jobs - if not sufficient entropy of jobs is in a cluster and no runtimes availabled (beyond a worst case 96h). I.e., VOs will have to provide runtime requirements for their pilots and sufficient entropy dor draining backfilling to allow a site to reduce badput during draining due to VO tasks with different core:mem profiles.

It is a bit contradictionary if some VOs have to justify low CPU utilizations, when their overall pilot utilication is due to inetrnal pool inefficiencies wrt to payload scheduling/draining in pilots, while site scheduling inefficiencies due to VO job mixtures and requirements are supposed to be not accounted for and are supposed to be to the expenses of a site.

While accounting as of now is fixed to a 2GB/1core quantization, we see the need by the VOs for a more flexible job requirements - however, it is unclear to us, how this would crystalize into pledge requirements and how accounting would have to be realized in a memory:core plain.

While we can currently absorb and run different core:mem profiles, e.g.,

https://syncandshare.desy.de/index.php/s/4NZcBt6NMQJGdJp

we worry that without an effort by all parties, this might lead to issues disadvantagious for all.

As fallback option whole node jobs might be an option, where we effectively segement the clusters and allocate VO jobs to dedicated nodes (either 1 job per node or assigning a node to a VO) and account the node utilization as such to a VO including idle and draining.

High-memory payloads

We could already have pilots with 4GB/core, that should help. Otherwise, filling the pilot by core or memory is the same for us.

On the grid, we have up to 4GB per core. We can provide also 8GB per core if you are using our OpenStack infra.

The site provides 4GB/core and can run jobs with higher memory requirements if that is the desire of the experiment, but at a cost of using up more core-equivalent allocations.

Our site has at least 4gb/core and many resources with lots more memory up to 128gb/core. We would be happy to participate in this paylod

We are open to the idea of teasing cores to allocate more memory, depending on your needs. However, this can be detrimental to CPU accounting calculations, so it would be nice to have a new metric for resource utilisation that includes CPU, memory, and slot disks.

Each job is free to use the resources it has requested. This includes leaving some requested cores idle to use more memory on the remaining cores.

However, we cannot support the exchange of resources between jobs - e.g. one job using only half its request and another job using the remainder. Each job must request an honest estimate of its resource usage for us to properly schedule jobs.

We have 2.67GB/core so we are ready to try running these jobs.

We feel that there should be some suitable accounting bonus for sites that choose to provide such capability. For example, the full number of CPUs should be charged corresponding to 2GB/core...if a job uses 8GB it should also be "charged" for 4 CPUs.

We already heavily support ATLAS in running HIMEM jobs @ Wuppertal. We have nodes with 4 GB / RAM physically and - together with Rod - support scheduling jobs which even need a bit more than that.

interesting but need to find correct configuration with batch manager

We as standard purchase nodes with 4GB/thread so this would be our acceptable/preferred limit for jobs at our site.

We have no swap space and our current core/memory ratio is 4GB/core (38 physical cores and 170GB of memory)

we can already run such jobs for ATLAS (push job request including amount of RAM requested). We have not paid attention to the accounting implications of this up to now. If the number such workloads is to increase significantly, we would like some solution in place (e.g. request more cores so that the total RAM requested scales to 2Gb/core). Averaging the amount of RAM across all jobs submitted cannot guarantee to be a solution (e.h. himem jobns packing nodes), of course it does help.

High-memory payloads

Fine as long as the memory is properly requested from the batch system. The equivalent amount of blocked cores should be reflected in the accounting.

We've been purchasing compute with the ratio 4GB/physical core for many years (10+), and it seems to just suffice for a wide variety of grid and local user jobs.

In the case where high memory usage requires leaving cores on a node idle a concern for a site would be accounting - jobs are accounted for in terms of CPU-hours, not GB-hours. This might need changing if we see lots of idle cores.

It has not been an uncommon practice in the past to see VOs, requiring more RAM for a job, to submit jobs requiring higher number of cores as a means of securing that memory in an environment where jobs are assigned a fixed amount of RAM per core. This is not useful.

A natural counter to wastage from high memory jobs would be for an attempt to generate useful low-memory job loads, which could perhaps take up any spare CPU slots.

Up to 4GB/core is ok.

we do have roughly 2GB/core. if we can account propely for idle cores, we can also provide more mem per core

We would upgrade our site with new hardware. The upgraded worker node of our site could satisfied 2GB/core requirement and no idle cpu core there.

We do not favor this proposal. We don't want to leave cores idle.

Yes, we should do this. Several workflows nowadays use high memory and that would make the site more useful.

The only concern is accounting. If there is a proper metric to recognize the high memory jobs and site’s contribution, it would be OK as a site point of view.

We've been providing 4GB/core for many years. Having a reasonable job mix that averages to the standard memory requirements from VO card is completely ok. Jobs must request the correct amount of mem at the CE, though in order to help scheduling them correctly on the site

I think it is preferable for handling diverse jobs. Some changes to our local batch system will be necessary, but probably not too hard. We may need to discuss the accounting strategy.

Clearly define if more than 2G/Core is required and future spend can work towards this.

High-memory payloads

Generally we are fine with high-memory jobs, however such jobs will cause low CPU utilization what might be seen as if the site is providing computational resources below the pledge. This issue have to be addressed in the accounting.

This should be discussed at the workshop. The need for high-memory must be quantified and the accounting should probably be adapted. In any case, we will consider the request. We already provide 3 GB up to 4 GB/core on recent workers. Leaving too much unused CPU on Tier 2 sitesand using more and more opportunistic CPUs at the same time would

require some clarification. In other words, a clarification of what is required from WLCG sites and CPU pledges which could not be provided by opportunistic CPU resources would help a lot.

For SWT2, we are focusing on running HIMEM workloads in the SWT2_GOOGLE_VHIMEM PanDA queue.

MWT2 does not idle cores to process high memory jobs but we do have over 20k job slots having a least 3 GB/core and about 3k job slots that have at least 4.5 GB/core. Only ~1% of our servers have only 2.0 GB/core.

For many years we have been buying servers with at least 2.666 GB/core. About a year ago this was the distribution memory/core:

GB/Core Cores % of Total
2.0 600 1.3%
2.4 10,520 22.3%
2.7 15,456 32.8%
3.0 1,408 3.0%
3.2 6,640 14.1%
4.0 9,216 19.6%
4.6 2,016 4.3%
4.8 320 0.7%
5.3 960 2.0%
Total 47,136 100.0%

The current distribution of memory/core is similar.

Given that we have been buying more than 2.0 GB / core, we would prefer not to idle cores to meet the memory requirements. We would consider turning off Hyper Threading on some servers to meet the need for high memory jobs. This would double the memory / core and make the slots faster. Turning off Hyper Threading could be a better way to deal with high memory jobs because it guarantees that the job will run on two physical cores and that only the 20%-30% of the server compute power associated with the disabled virtual cores is lost.

High-memory payloads

I'd be interested in what kind of volumes and what kinds of "high memory" we're talking about. For local (ie non-grid) submission, we have a limited number of nodes available for "high memory" jobs. If we needed something special (ie very high memory jobs) then there may be some adjustments. If we're just talking about running 16Gb memory but only say 1 "core" equivalent of cpu in an 8 core pilot, then the impact really is on things like how we measure and account for low efficiency.

On a technical point, we run with cpu shares in cgroups, rather than dedicated cores. So it doesn't necessarily follow on a more global view of a pool that we necessarily have unused or idle cores. Running more hetereogenous workflow though would be made more simple if it were clearly delineated so that sites could make routing or node balancing decisions.

Nodes provisioned in the last ~5 years are provisioned with 4GB/core, so in principle we can accommodate such jobs. It would be good if such jobs can be isolated in specific queues to ensure efficient placement.

If the experiment wants that, it would be OK for us

PIC currently provides CPU resources with more than 2 GB per core (indeed, the current purchases are made with 4 GB/core RAM memory). In the context of high-memory payloads running inside a 'pilot' job, any CPU inefficiencies for these high-memory jobs are absorbed by the VO. However, directly submitting high-memory jobs shifts the responsibility for inefficiencies arising from unused cores to the site itself. This resource allocation strategy could impact accounting practices, as such tasks often demonstrate reduced CPU efficiency despite their substantial memory demands. The question of accountability for these inefficiencies warrants discussion, particularly since WLCG does not currently track memory usage. Understanding these non-trivial CPU inefficiencies becomes increasingly crucial if VOs increasingly deploy a significant number of high-memory jobs using this method.

Not particularly positive about having idle cores, but the request is manageable. Already in place for ALICE. A note: we guarantee 3.5GB/jobslot, not 2GB. If the request is on long term basis we can procure dedicated hw to HIGHMEM queues, maybe should somehow be reported in the pledge requirements.

We already support high-memory payloads. If the requested memory goes above 8 GB, then more cores are accounted (1 core up to 8 GB, 2 cores from 8 to 16 GB, etc). We have a cap on the number of such jobs allowed. If experience proves that the experiments can keep the average at an acceptable level, we can relax the cap.

We prefer full utilization of all cores and avoid idle cores since our site is configured with 4GB/core physical memory, which is significantly above the minimum 2GB/core required by the experiment.

OK if idle CPU cycles count against VO's pledges

Whole-node scheduling

What is your experience/opinion/concern regarding whole-node scheduling, which some of the experiments are interested in?

We Don't allow whole-node scheduling at our site

We don't have this right now, but it can be implemented. I have doubts, though, that it would be worthy given our site architecture (a central remote storage for all IL sites, with the network bandwidth being bottleneck).

Usually not advantageous since it means beneficial effects from mixing different workloads on since sockets are lost.

The whole-node scheduling can be problematic for nodes shared between users/queues/partitions as the reservation and waiting for a whole-node job when nodes with 168 CPUs(slots) have 1CPU jobs running. If using a dedicated set of nodes/partition for whole-node usage, these problem vanish and the memory management problems for complex software (O(1000) processes per job like in ALICE) are pushed to the experiment which might have a better tuned management then the simple memory-per-cpu restriction.

For the ALICE usage we did not move yet to whole node usage but we plan to (we have dedicated nodes/partition for ALICE).

Unless we are able to run preemptable jobs i doubt it would be an efficient use of our resources. It might work for GPU nodes.

Not keen. We hacked something together for some testing with ALICE, but attempts to do it "properly" (i.e. incoming jobs requesting a large number of cores) failed due to slow change over from one workload to another, even on an almost empty test cluster. Providing reasonable quality of service for incoming jobs (i.e. keeping queuing times down) without risking great under-utilisation by keeping resources ringfenced for whole-node use will be very tricky, and will likely make it even harder for sites to manage resources efficiently (pilot jobs make things hard enough as it is).

This is a problem for us, we don't have many machines and these machines have increasingly a high number of cores. Allocation of full machines may led to high inefficiencies. Furthermore the Tier-2 service relies on an underlying multidisciplinary shared high throughput infrastructure, this means that we cannot allocate machines and have them half empty, this also means that much of the capacity comes from shared/available job slots in the infrastructure.

Sure, this is the normal case for HPC workloads that most of our batch system operators are most familiar with.

We can run jobs letting them request the whole node on which they run. For that we would only need to add the corresponding submit option ("--exclusive" for "sbatch") for jobs from a whole-node queue. The whole-node job should the usage of all resource on the node.

We have virtualized nodes (VMs) of different sizes and use Kubernetes as our batch system. The node size is arbitrary. Whole node scheduling implies another level of workload management inside a job (e.g. running a node daemon). On Kubernetes there are other more flexible ways to approach this. I don't think whole-node pods make sense.

Whole-node scheduling

This is a bad idea. Sites are better at scheduling jobs than VOs. Sites understand where their own bottlenecks are and can over commit resources when appropriate.

It's not practical with our new nodes having 512 virtual cores, we have very few nodes now.

We have some very large nodes with 256 cores. Might be inefficient use of resource?

Needs efficient draining for good turnover of slots. Not enough to say VO needs very long walltime.

We have a heterogeneous cluster, so we believe 8 or 16-core jobs would be more efficient. We're open to exploring this option, though -- again, as long as the incoming jobs request the correct configuration from SLURM.

It is a possibility, but we will have to account a VO for the utilization of a node including idle and draining inefficiencies. So far, major VOs have profited at us to be able to flow over to under-utilized resources, i.e., gaining more HS23 then pledged.

Since we have also non-WLCG VOs on our cluster, a move to WNJ would have to be agreed with all parties.

no experience. but we would not like long pilots. turnaround between pilots and local users is paramount to good user experience on our batch system.

It may be easier to manage for us, but we need some time to test the scheduling.

We have whole node scheduling running at some sites and it would have helped ATLAS in the past. Probably not a route we would go down for Canadian Tier-2's based on batch in the near future, especially as nodes could have close to 200 cores.

we ave at least 48 cored/node and we let local users use whole nodes

In any case, I don't think it's a big deal, because whole-node scheduling can cope with tasks that require larger CPU cores, so I think it's worth considering and applying if there are more such tasks. However, HTC seems to have a lot of single-core tasks, so I think whole-node scheduling can be applied to some extent.

We have made good experience with ALICE whole node jobs. We would prefer whole node jobs over experimental/large jobs (e.g. pilot overloading, 16-core pilots).

As a multi-VO site, whole node jobs raise similar issues as fragmentation from single/multi-core jobs. We currently can only offer a small testing partition for whole node jobs unless multiple VOs commit to a reliable pressure of whole node jobs.

No experience but we could try it.

Whole-node scheduling

Practically, we could provide this but we need to understand if ATLAS can actually use whole nodes without wasting resources. Since our Tier-2 is ATLAS-only, this is important to verify.

Would be fine for us as long as we can integrate it in our regular Slurm scheduling.

loss of time if it need draining but ok if always ur=sed as whole-node

Fairshares would be a concern, node size variation would be another, some nodes will are still as small as 16 job slot others as 100+ the control of which node gets reserved would be a concern.

We have no opinion or concerns but beware that we have a single SLURM partition for multiple experiments (CMS, Alice, Belle2)

for ATLAS, most of workloads do not scale well beyond 16-core (declining CPU efficiency). Some workloads cannot run at all on a whole node, considering node sizes up to 256 slots

Given that we are planning to replace our compute resources by resources of an HPC cluster with whole node scheduling (which we currently fill with the smaller jobs via an overlay batch system), this would be no problem.

Whole node scheduling could be a novel solution for instances where high-and-low resource jobs can be mixed within "whole node pilots", the pilot "filling in" any CPU/RAM resource gaps it might have. But the batch system can in theory do this too if given precise resource requests.

I have no problem with whole-node scheduling in principle (it's one of the ways our local users run on our cluster), but it should be carefully reviewed if the primary motivation is to reinvent/circumvent the batch system.

That is acceptable.

since we are a single VO site, there should be no blocker

Sorry, we haven't had such experience.

We don't use whole-node scheduling.

A job requesting a node can wait for a long time for resources to be freed in a merged queue, especially for modern machines with a high core count. Separated queues may be required, granted the queue can be kept busy.

We are OK with whole-node submission but it definitely requires more effort of experiment to use them more efficiently comparing to single core job processing.

Whole-node scheduling

It's not batch-like, actually. Do experiments want to look for cloud-based resources, actually?

Not sure about "whole-node scheduling", but if it can efficiently handle high memory jobs and each site can choose whether or not to accept such a mechanism, it would be good.

Whole-node scheduling is fine for us.

Whole-node scheduling can be configured and supported without difficulties. This has been done in the past for non-grid local users.

Possible for us. However, would likely require some new configuration(s) in the batch system (slurm).

Since MWT2 is an ATLAS-only site (with some low memory opportunistic use by OSG jobs also), we have no experience with whole-node scheduling. ATLAS has not asked us for it.

We already have some whole-node scheduling for some dedicated pools of resources. The issue isn't the whole-node scheduling really, I'd argue it was more about shared vs dedicated resources. Increasing the amount of dedicated as opposed to shared resources normally results on the one hand in more predictable and bespoke capacity at the expense of overall throughput and efficiency. In our pools VOs that rely more on shared resources end up often with more overall compute, because they benefit from surplus.

No problem from our point of view. We don't have experience so far. We would need to study how to provide that functionality. A concern is that it would make the sharing of resources with other VOs/groups more static and less flexible.

Currently, the recent servers we purchased boast a large number of cores. However, whole-node scheduling might not be particularly advantageous for us. This approach doesn't leave slots available for smaller tasks, and we support a variety of projects (approximately 60 at the moment), each with different requirements. Many of these projects still submit single-core jobs or multi-core jobs with a very low number of CPU cores requested. In the case of HPC facilities or exclusive-VO WLCG sites, whole-node scheduling makes sense, and thus users are typically encouraged to use it. This is however not the case with a multi-VO multi-project site such as PIC. We need to thoroughly understand within WLCG the advantages and needs associated with submitting whole-node jobs in multi-VO Grid sites.

Not in favor. This make is more difficult to handle the scheduling of other, non-wholenodes VOs.

theoretically nice, practically a bad idea. We only see ATLAS multicore jobs, and the supply of these jobs regularly (once every 24 hours) dries up briefly, meaning either a whole node doing nothing, or else that we allow other jobs to run there and then ATLAS waits for a day or so for a new whole node to be emptied of jobs - you spend a lot of time draining. Alternately, we could just assign some nodes to ATLAS, but then we would charge the wall time on these nodes/cores regardless of whether they were being used.

In principle, we can support it and are not against it. However, the main concern is that it can result in more draining process, i.e., more idle CPUs. Effort should be put in place in experiment workflow to ensure high cpu utilization at all times since there are limited resources at the site.

OK if whole node CPU cycles count against VO's pledges

16-core vs. 8-core jobs?

Some of the experiments are interested in standard 16-core instead of 8-core slots. What is your opinion from the site perspective?

Of course works in general, but many CPU generations have a mismatching NUMA architecture. Has that been evaluated?

Anything bigger than 8 might as well be whole-node submission.. a very popular cpu is the 84 cores single-socket which is not divisible by 16 (you can have 16 threads or 4 cores unused)

Sure, if the efficiency is >= that of 8 core jobs

combined with 96h of runtime requirements as only and upper limit, cluster entropy will be crap and scheduling becoming inefficient. A VO has to provide job runtime requriements as well as sufficient entropy in their jobs so that draining inefficiencies can be somewhat attenuated.

Fairly agnostic on this point, other than will make back-fill scheduling harder to take advantage of unused resources at the large shared sites

For our site, we have a server with 24 cores. This leaves us with an ambiguous 8 cores, and I was wondering if this would be handled.

16-core pilots would be a major problem for our scheduling. This would intensify the issues of single/multi-core job fragmentation; we confidently estimate the amount of resources blocked by draining etc. to double (from ~3% to ~6%). 16-core pilots are only economic if single-core jobs are deprecated. Notably, we consider whole node jobs or flexibel size pilots significantly superior from an efficiency standpoint.

If ATLAS can provide us with sufficient SCORE jobs to handle hosts where the number of CPUs are not divisible by 16

This would be whole node in some cases, No issue with it otherwise.

CPU Efficiency should be studied first

Both options are ok. But our machines are 56 cores so 8-core slots are preferred.

Yes, for the same reason as high-mem.

It depends on the job distribution mix on the site. Multi-VO sites running many single-threaded jobs as well will the scheduling issues.

I have no objection to running 16-core jobs along with 8-core jobs.

16-core vs. 8-core jobs?

We are ready to consider this change

We would want to study this to see whether 8 or 16 core jobs would use the site most efficiently. Our get guess is that 8 core jobs are more likely to use the site more efficiently than 16 core jobs simply because it is quicker to assemble a vacant cores on a single server than 16 vacant cores. However we have not studied it.

We don't see any inconvenience. The drainage in the computing farm would be a bit slower.

prefer to keep 8-core as maximun, but 16 is still manageble

more or less the same argument as with whole-node scheduling, although less severe. If the experiment can provide a steady supply of waiting jobs with average time of many days between moments of zero waiting jobs, then the number of cores per slot doesn't matter much. Given the current situation, moments of zero waiting jobs occurring on a daily basis, fewer cores is always better. See the following paper for more information: [Scheduling multicore workload on shared multipurpose clusters](https://iopscience.iop.org/article/10.1088/1742-6596/664/5/052038) , esp. the discussion about the size of slots starting on page 5.

this is not an issue; however, similar comment as mentioned previously regarding cpu utilization. All assigned cores should be kept busy at all times, and synchronously as much as possible.

Generally fine, but we have some concerns regarding the efficiency, as already there are jobs which allocate 8-cores but actually use 1-2 cores. Also the number of slots need to be adjusted to what is available at worker nodes (e.g. 16-core jobs are not optimal for a server with 40 cores available).

ARM resources

What is your estimation regarding fraction of ARM resources vs overall computing resources on your site in the time frame of 2-3 coming years?

0

0%

50%

Low / zero.

Benchmarks show that normalized cost/performance and energy/performance for CPUs like AMD has is much higher than anything ARM have at this moment.

0 to 50%

~33% ARM

For the time being we don't have plans for ARM. We may exploit some ARM resources from a national HPC service.

Maybe a small share as an experimental resource, but no current plans.

~30%

0 % < ARM_fraction < 1%

We might get just 1 node for experimentation and testing for now.

upto 20%

Probably 0

depends on the funding

ARM resources

0 %

10%

Close to zero for use by grid and HTC.

Depends on Alliance Canada. We indicated our interest

Predicted at 5% or less.

Currently we have ~10% ARM, we expect the share to grow over time, but we will of course adapt to the needs of the VOs.

ARM resources seem to provide good CPU effectiveness for the amount of electrical power consumed. We may want to invest in some ARM resources once we better understand what it would take to manage them and ensure that wouldn't cost our site extra effort and that the ARM resources are also cost effective for $/HEPSCORE23.

Would be happy to purchase but currently not enough interest for the limited funding we have to be spent on it.

We don't rule out that we might buy ARM nodes in the future. Currently there are no plans

none

Completely unknown. Our current purchase plan is small test bed for grid and local user evaluation.

NA

I am afraid we might not have ARM in 2-3 coming years.

Zero ARM resources.

No plans right now.

Probably 10%

Maybe. There is no real use case for them atm. There is only a slightly better HS23/watt performance of ARM-based systems. Nothing that would justify a completely new platform.

ARM resources

Probably 0% or <1% for the 2-3 coming years.

likely >20% - If ARM is cheaper/more efficient then this may grow higher.

5-10%

We remain open to the idea of purchasing some in the future for production. For now, we have no clear estimation regarding the fraction of ARM resources in the next coming years

We could potentially increase the fraction of ARM resources in the cluster to ~15-20% during this period.

A guess is that in the few months we will buy 1500-2000 ARM cores (3%-4% of total cores) for one our sites where we are more limited by the available cooling capacity. Within 2-3 years that could raise to 10%-20% depending on our experience with the initial purchase.

We are currently at <1% and I don't expect that to be drastically different in just 2-3 years.

We are waiting for the experimental software to finalize, but we expect to replace up to ~20% of capacity with ARM in the next years if experiment adoption continues at the current pace.

No plan so far.

At PIC, energy efficiency is a top priority. ARM processors are known for their superior performance per watt compared to traditional x86 processors, which translates to reduced energy consumption and operating costs. Given our limited space and power constraints, the adoption of ARM processors is a matter we must seriously consider. Currently, experiments are assessing the compatibility of their workflows with ARM processors, and we're closely monitoring the results of these validations conducted in Glasgow. These findings will inform our internal discussions regarding the potential adoption of ARM technology in the future. However, we use liquid cooling immersion, and we are not yet sure about ARM processors compatibility in this setup.

small fraction, but it also depends on the request from the VOs and experiment sw availability

There are no plans to roll out ARM at this time.

We are flexible in purchasing ARM resources in the future; however, it all depends on the readiness of the experiments to use ARM resources in full production (i.e. not for a limited set of use cases) in the future. As of today, we don’t believe it is the case.

ARM resources

Any comments regarding ARM resources or your experience of hosting ARM resources?

N/A

no

handled by different team

Many physics workloads profit greatly from large CPU vector extensions if modern coding style is used, so benefits from ARM resources seem limited especially at T3 level.

We do not have ARM resources nor we plan to.

we note that the power efficiency of the latest generation of x86 CPUs is much closer to ARM than 6 months ago.

ARM ecosystem is still a little less developed / refined than x86 (e.g. we hit a problem in grub configuration during provisioning), but generally we've had few issues and most things just work. This will only get better as ARM hosts become more widespread.

While the HS23/watt numbers are impressive compared to Intel CPUs, the latest AMD servers are also really delivering good numbers.

So far good.

Need to make sure all of the WLCG software stack and yum repos provide RPMs for ARM.

We have a single test ARM server, it is slightly harder to configure and manage than x86 but support is improving over time. We haven't seen ARM be that much better than x86 from a cost / power usage.

No funding

No

never comes up during purchases rounds

Our first tests we ARM is very positive. We have planned to purchase additional ARM server if it is needed by CMS.

ARM resources

No experience yet. Might see some deployment, but likely only for HPC sites and not HTC.

We understand that the compatibility issue has been resolved, so we're pushing for adoption to address the overheating issue. However, there are some administrative issues in introducing the server, especially that there are not many ARM vendors in Korea.

first benchmarks show a higher power efficiency than our current x86 based WNs

Waiting to see other site's experiences.

Happy to purchase if enough desire.

we don't have any experience. There might be concerns with OS and software stack

NA

No.

We have no experience with ARM.

We have no experience with arm so far.

no comment

We have a single ARM node for now. If the good compute performance, the low power consumption, and acceptable cost of the operations are validated, we're interested in purchasing the ARM nodes for Tier 2 operations.

ARM resources are now used only for testing.

Our site CPU power and electrical power consumption are dominated largely by AMD EPYC servers. If one simply looks at the whole power consumption it seems clear that ARM CPUs produce many more calculations than x86 per unit of electrical power used which would be green and more likely to fit within the existing cooling capacity. It seems clear that we need to start using ARM to replace x86 and that we will need to make whatever adaptations are needed to support servers equipped with ARM CPUs.

ARM resources

ARM so far is quite new for us, most of our ~2k cores are newly installed.

No experience so far, we aim to have a few test nodes soon.

It seems interesting from the efficiency point of view, but would probably make resource management more complex

None

datacenter-grade servers not fully mature yet, in particular for what concerns ancillary tools, i.e.PXE. A huge obstacle for us is the level of support we obtained from vendors, i.e. on-center and not on-site. We are satisfied about general performance and HS/W ratio even if an in-depth TCO vs benefits analysis is still ongoing, also taking into account that Intel and AMD are selling CPUs with a similar number of cores. Testing of combined CPU-GPU SoCs (i.e. grace-hopper) is also still in progress.

There are some performance arguments that for our data center and use cases work out unfavorably for ARM. We could provide more information if desired.

We don’t have any experience in deploying and managing ARM resources.

Running x86_64-v1 or even older?

If your site still runs x86_64-v1 or even older, is it still a considerable amount? What is the plan for retirement?

N/A

Not applicable

To be retired by June 2024

Not exactly sure what "v1" is, is that 20 years old?

v2 will be retired this year

N/A, only v3 or newer here

Probably won't be running any significant CPU resources that are more than 5 years old in the next year

It's a significant amount at the moment, but we expect it to be replaced by the end of the year.

n/a

We have no such resources left.

Not as part of batch system, some user access point desktops remain X86_64-v1. Plans to retire are as soon as the users of those ones have a need for newer they will get.

6% of cluster capacity. Did not plan to retire them yet

None of our kit is quite that old, but we still have a lot of x86_64-v2 kit.

We are currently replacing them with the ones that supports x86_64-v3. But the machines will be still x86_64

no x86_64-v1 on site

We are currently in the process of decommissioning all x86_64-v1 nodes.

Running x86_64-v1 or even older?

If your site still runs x86_64-v1 or even older, is it still a considerable amount? What is the plan for retirement?

~25% retirement due soon.

No. The oldest CPU we operate in PIC is Intel Xeon E5-2640 v3.

x86_64-v1 not in the datacenter, neither in production nor in testbeds (referring to the micro architecture levels defined here: https://en.wikipedia.org/wiki/X86-64)

no