

Experiments and HPC

David Cameron (University of Oslo)
WLCG workshop, Lancaster, 8 Nov 2022





Disclaimer

These slides are an attempt to summarise the current state of HPC use by the four LHC experiments

It does not represent any official statements from any of the experiments

Any omission or errors are purely my fault

Thanks to all who provided input and feedback

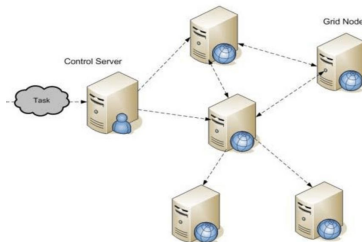
HTC vs HPC

Taken from [EGI Glossary](#)

Term	High Throughput Computing
Abbreviation	HTC
Definition	A computing paradigm that focuses on the efficient execution of a large number of loosely-coupled tasks. Given the minimal parallel communication requirements, the tasks can be executed on clusters or physically distributed resources using grid technologies. HTC systems are typically optimised to maximise the throughput over a long period of time and a typical metric is jobs per month or year.

Term	High Performance Computing
Abbreviation	HPC
Definition	A computing paradigm that focuses on the efficient execution of compute intensive, tightly-coupled tasks. Given the high parallel communication requirements, the tasks are typically executed on low latency interconnects which makes it possible to share data very rapidly between a large numbers of processors working on the same problem. HPC systems are delivered through low latency clusters and supercomputers and are typically optimised to maximise the number of operations per seconds. The typical metrics are FLOPS, tasks/s, I/O rates.

- Long timescale
- Distributed worldwide
- Loosely connected
- Data intensive
- Designed for HEP needs



- Short timescale
- Single room
- Tightly connected
- Data intensive
- Not designed for HEP



Argonne National Laboratory's Flickr page, CC BY-SA 2.0
<https://creativecommons.org/licenses/by-sa/2.0>, via Wikimedia Commons

Barriers to exploiting HPC

- No network access from the worker nodes
- No CVMFS available
- No CE service to submit jobs
- No persistent storage
 - Data needs to be moved in and out for the job
- Jobs must use whole nodes or multiple nodes
- Sociological/political barriers
 - High level of security
 - Access only granted to certain individuals, credentials need renewed regularly
 - HEP software is viewed as poor standard and untrusted
- Not all of these apply to all HPCs, and gradually these barriers are being lowered



So, why bother?



• Total combined performance of all 500 exceeded the Exaflop barrier with now **4.40 exaflop/s [Eflop/s]** up from **3.04 exaflop/s [Eflop/s]** 6 months ago.

[Top500 list, June 2022](#)



Fugaku, #2

<https://www.r-ccs.riken.jp/en/fugaku/>

Frontier, #1

By OLCF at ORNL - <https://www.flickr.com/photos/olcf/5211762388/>
CC BY 2.0, <https://commons.wikimedia.org/w/index.php?curid=11910238>

- A great untapped rapidly growing resource
 - More than 100x WLCG (1 million 3GHz cores x 10 Flops/cycle = 30Pflop/s)
- A substantial part of national computing infrastructure investment now and in the future
- Potential for allocations or “free” opportunistic computing
- Interesting and motivating R&D and PR
- Improving flexibility of experiment workloads and services

The screenshot shows a webpage from the ATLAS Experiment website. The article is titled "Harnessing a supercomputer for ATLAS" and is dated 2 June 2022. The text discusses the ATLAS Collaboration's use of a global network of data centers and the ATLAS LHC Computing Grid to perform data processing and analysis. It highlights the power of purpose-built supercomputers and the challenges of HPC for ATLAS data taking, such as access to supercomputers, CPU architecture, and network connectivity.

[ATLAS briefing on Vega HPC, June 2022](#)

HPCs used per experiment

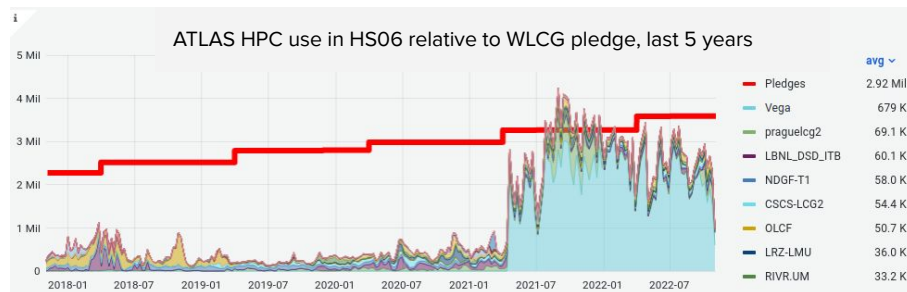
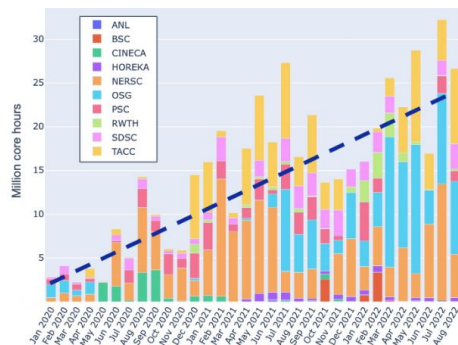
- LHCb
 - Piz Daint in CSCS (Switzerland)
 - Marconi-A2 in CINECA (Italy) – not used anymore
 - SDumont in LNCC (Brazil)
 - MareNostrum in BSC (Spain)
- ALICE
 - CINECA (Italy) - used in 2020
 - LBNL (Berkeley, USA):
 - Lawrenceium (in production)
 - CORI (to be decommissioned)
 - Perlmutter (being commissioned)

- ATLAS
- Past:
 - Titan (Oak Ridge), Theta (Argonne), Edison (NERSC), Tianhe-1 (China), SuperMUC (Germany)
- Currently:
 - Frontera (TACC), Cori (NERSC), Vega and Karolina (EuroHPC), Toubkal (Morocco), RIVR (Slovenia), Tokyo HPC (Japan)
 - As part of pledge: MareNostrum (BSC), CSCS (CH), HPC2N and NSC (NDGF)
- Planned:
 - Perlmutter (NERSC) currently being commissioned

CMS:

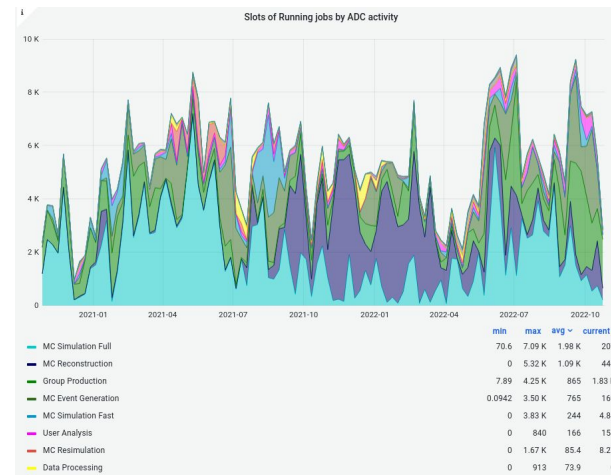
Machine	Location	Architecture*	Status
Piz Daint	CH (CSCS)	x86 + Nvidia GPU	Production
CLAIX	DE (RWTH Aachen)	x86 + Nvidia GPU	Production
HoreKa	DE (KIT)	x86 + Nvidia GPU	Production
Marconi 100	IT (Cineca)	POWER9 + nVidia GPU	Validated, pre-production
MareNostrum 4	ES (BSC)	X86 (+ GPU)	pre-production
Cori	US (NERSC)	x86	Production
Frontera	US (TACC)	x86	Production
Stampede2	US (TACC)	x86	Production
Bridges-2	US (PSC)	x86	Production
Expansive	US (SDSC)	x86	Production
Anvil	US (Purdue)	x86	Production
Perlmutter	US (NERSC)	X86 + nVidia GPU	integration/commissioning
Summit	US (OLCF)	Power9 + nVidia GPU	integration/commissioning
Frontier	US (OLCF)	X86 + AMD GPU	Planned
Polaris	US (ALCF)	X86 + nVidia GPU	Planned
Okami HPC testbed	US	ARM	Planned
Leonardo	IT (CINECA)	X86 + nVidia GPU	Planned
MareNostrum 5	ES (BSC)	x86 / Arm + nVidia GPU (tbc)	Planned
Jureca	DE (FZJ)	X86 + nVidia GPU	Planned

CMS HPC use in M core hours per month Jan 2020 - Aug 2022



What runs there

- The main workflow is MC generation/simulation
 - Easy on I/O, small or no input data, low memory and few dependencies on external services
 - Stable software releases used for long periods
 - Not time-critical
- Where possible, other production workflows are run (eg CMS full chain on all DOE/NSF HPC)
- Some however run all workflows including analysis
 - Vega, CSCS and NDGF for ATLAS
 - HPCs at LBNL for ALICE (helped by co-located T2 site)
 - CSCS, CLAIX and HoreKa for CMS



ATLAS activities running at CSCS, last two years

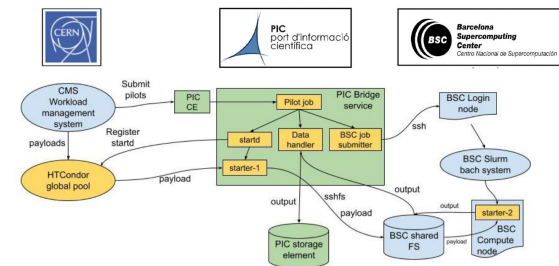
Software availability

- CVMFS is the de facto standard throughout HEP for software distribution
- Some HPCs install CVMFS + squids natively, which makes things a lot easier
 - In fact it is mandatory for some experiments to use the resource
- Various solutions exist for HPC without CVMFS
 - Copy part of the tree (eg particular software release) to local file system and point jobs there
 - Install CVMFS as an unprivileged user with [CVMFSExec](#)
 - Develop tools such as [subcvmfs-builder](#) to deploy releases automatically
 - Pack software into a container in which the job runs
- Some of these solutions are generally only suitable for HPC running specific releases of a specific workflow
- To make use of HPC as a general purpose resource CVMFS must be available natively, or at least a squid provided to make use of CVMFSExec

Data management and communication

- Two main issues to solve:
 - No local storage element
 - No network connectivity to access remote storage or experiment services
- Some sites provide proxies or gateways for network traffic
- Another method is to “tunnel” traffic through a login or edge node
- Otherwise typically solved by deploying an edge service which handles data transfers independently of jobs
- How does the pilot model work in the most heavily constrained environments?
 - Pilot runs on the edge node or even outside the HPC: pull job, do data transfer then submit payload to batch system
 - DIRAC PushJobAgent
 - Edge service which handles job communication, data transfers and submits to batch
 - ATLAS Harvester
 - Central service which handles job communication, and submits fully-formed jobs (payload and input/output) to edge service which handles data transfers
 - ATLAS ARC Control Tower (central) and ARC CE (at HPC)
 - Edge service which handles communication with jobs through shared file system
 - CMS at BSC

HTCondor split-starter mode



J.M.Hernandez et al, “Integration of BSC CPU resource in CMS”

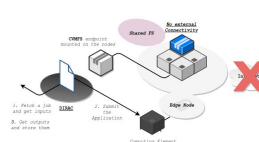
- Tackling the distributed computing challenges
- Push model. No Ext. connectivity

In Progress: v7r2?

Some SC do not provide any external connectivity at all, neither on the WNs or the edge node.

PushJobAgent

- Works like a pilot outside of the SC
- Fetches jobs, deals with inputs and outputs, submits the application part to a SC
- Require a direct access to the LRMS



Other issues to solve

- Access to external services for eg conditions data
 - “Fat” container image with all software and conditions data
 - Mirror to local shared file system
 - Edge service proxy
- Batch system policies
 - Whole node scheduling required
 - Single pilot running many parallel single core jobs inside
 - Multi-threaded/multi-core jobs using whole node
 - Minimum nodes per job
 - “Fat pilots” running many jobs in a single batch job, ATLAS “Jumbo jobs”

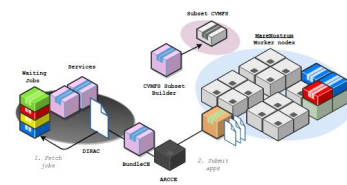
Work involved from expt side

- Significant effort can be required from central experiment teams and site contacts (the people who have access)
 - Proportional to difficulty but not always to benefits
- Each HPC is a snowflake, with unique challenges
 - Although work invested into a single site can help for others
 - Specific technologies like edge services or container images can be used on multiple sites
- HPC lifespan is short (3-5 years) compared to WLCG sites
 - Continuous work to commission the next new one (although can be made easier by existing relationships with sites)

●●●● LHCb-supercomputers collaboration

●●●●● Mare Nostrum, BSC: Development

- No external connectivity: Use the push model
- No CVMFS mounted on the WNs: Use the *Subset-CVMFS-Builder* variation
- To get multi-core allocations: Use the *BundleCE* variation

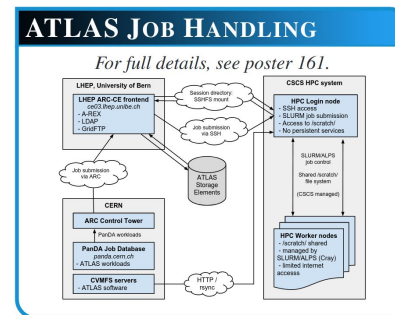


Virtual DIRAC Users' Workshop - Monday, 10th May 2021 21/21

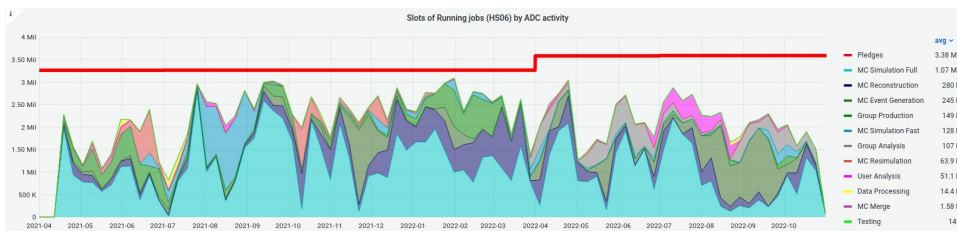
A.Boyer, "Integrating DIRAC workflows in Supercomputers"

Evolution from the HPC side

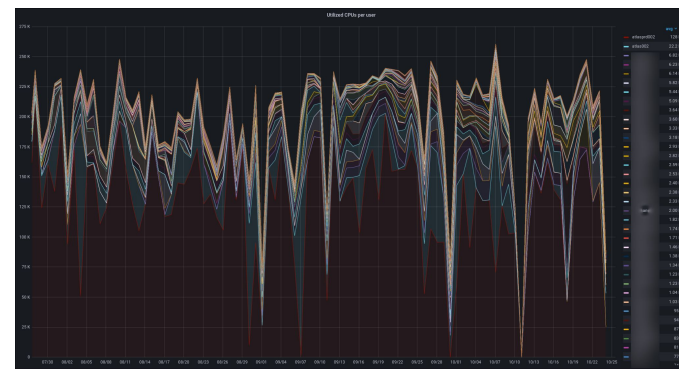
- In the past, typically ssh access was granted to a login node
 - Could then use tricks like wrapping batch commands around ssh and using sshfs
- CSCS is a good example of a restrictive site that now looks like a (pledged) grid site
- Vega EuroHPC was designed to be HEP-friendly
 - Worker node connectivity, performant shared FS, access through ARC CE
 - ATLAS can use it when other users don't



M. Hostettler, "ATLAS computing on the HPC Piz Daint machine", CHEP 2015



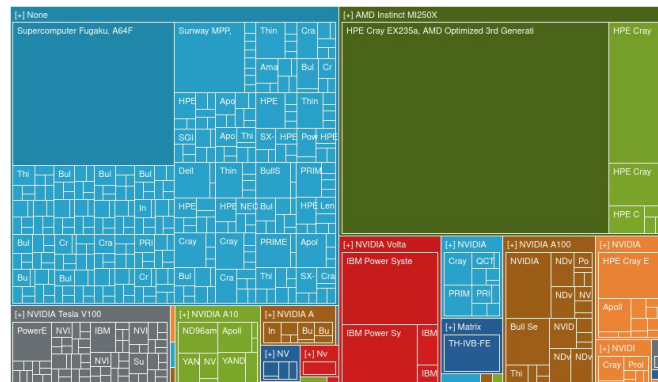
Running ATLAS jobs on Vega, last 18 months, compared to entire WLCG pledge



Vega usage per user, last 3 months
ATLAS average 150k cores

Software architectures

- GPUs make up the majority of the FLOPS of the Top500
 - GPUs are becoming heavily used in the online world
 - ALICE has GPU-ready offline software today
 - CMS plans for opportunistic use of GPUs (offloading 10% of reconstruction) starting next year
 - For others not at significant scale before Run-4 if at all
 - Except small-scale ML (e.g. flavor tagging, fast simulation NNs, ...)
 - Experiments' views on GPUs are collected in a [WLCG twiki page](#) - time to update it?
- ARM builds are at various stages of validation
- ATLAS, CMS and LHCb
- POWER is validated for production for CMS
- In general, a chicken and egg problem
 - If a resource is available work can be done to target it
 - We are not targeting specific resources because software is not available
- Also thorny question of pledge



<https://www.top500.org/statistics/treemaps/>

Pledges

- Most HPC use is opportunistic, i.e. outside the WLCG pledge framework
 - Either dedicated allocations to groups/institutes or backfilling
- Could they be pledged instead of grid sites?
 - If they look like a grid site, yes
 - Already done for CSCS and parts of NDGF-T1
 - MareNostrum used as part of Spain T1 pledge for ATLAS and CMS (although it does not run all workflows)
 - INFN will pledge Leandro (CINECA EuroHPC) as part of Italian T1 resources from 2023
- If not, difficult to accept as pledge
- In addition to running all workflows, the HPC must be constantly available at some level with a multi-year commitment
 - Not possible with allocation and fair-share models used by many HPC
 - But in some cases (eg EuroHPC) multi-year allocations can be granted for long-term production
- How to pledge non-CPU resources? Plans to benchmark GPUs are still in the very early stages
 - But needs to be thought about years in advance of pledging these resources

How to organise better

- Perhaps it's useful to have a set of “WLCG requirements for HPC”?
 - Suggested requirements below are just to be able to use them, with probably stricter requirements to be pledged
 - Some experiments could run some workflows with even fewer requirements
- Minimum requirements to be used by all LHC experiments
 - CVMFS installed natively or infrastructure (squids and modern OS) which allows experiment to use CVMFSExec
 - Limited external network access from worker nodes (for communication, not data transfer)
 - A “normal” way to submit jobs (well-known batch system)
 - Allow running containers
 - Allow persistent edge service
 - x86 architecture (possibly soon other CPU architectures like ARM)
- Useful to have
 - Local persistent storage, accessible through our usual interfaces
 - Good enough hardware (memory, storage, network) to support all workflows
 - Experiment-friendly admin :)
- Table from CMS note looks like a nice format/starting point

A synopsis for handshaking with sites

Category	Explanation	CMS standard solution	CMS preferred solution for HPC	CMS fallback solution (full utilization)	CMS fallback solution (for a fraction of workflows)	CMS no-go scenario	Possible CMS devels to solve the no-go
Architecture	Base system architecture	x86_64	x86_64	x86_64 + accelerators (with partial utilization)		Currently, OpenPower accelerators they could be used but at the price of physics validation	QEMU? Reconfiguring + ARM... physics validation?
Memory per Thread/core	Memory available to each thread / process	2 GB/thread	2 GB/thread	Down to 0.5 GB/thread needs heavy multithreading, at the expenses of CPU efficiency	GEN and SIM workflows need less than 2 GB/thread (0.5GB/thread would be a limit) in order to run efficiently	Less than 0.5 GB/thread	
I/O	I/O demand per process	5 MB/s/core	5MB/s/core		GEN and SIM workflows are mostly CPU bound still ok with 0.1 MB/s/core	Less than 0.1 MB/s/core	
Local Scratch space	Local space per production job	20 GB/thread	20 GB/thread local	Less than 20 GB/thread ok if a shared high performance FS is available on all the machines. Large multithreading lowers 20 GB/thread requirement to ~ 10	Some CMS workflows run for hours without creating huge local disk areas (GEN, SIM)	No sizeable local space and no shared usable FS	
Outgoing networking	Needed on WNs in order to access remote data, conditions, and to speak to the CMS Global Pool	Full outgoing connectivity	Full outgoing connectivity	Connectivity to only a subset of the IP ranges (for example, to CERN, and to a close xrootd proxy cache) And to everywhere we have condor services?	NAT with a very limited bandwidth via an edge service	No outgoing connectivity from the compute nodes and no NAT available	Edge service running Hadoop or HTCondor? Prepare a single container to be deployed at the edge and doing NAT for Condor, Squid, Xroot proxy cache, ...?

From CMS note 2020/002: “HPC resources integration at CMS”,
https://cds.cern.ch/record/2707936/files/NOTE2020_002.pdf

Summary

- A lot of great work has been done to exploit difficult resources, with variable return on investment
 - From zero to equivalent of WLCG grid pledge
- Our HEP model has been bent to fit the HPC environment in a variety of ways
- But HPCs look like they are becoming more friendly to HEP
 - At least in terms of accessibility, if not architectures
- Current challenges may be more on the software side to exploit different architectures
- A lot of commonalities in our requirements, can we present a united front?





Acknowledgements

Thanks very much to the following people for providing input and material for this talk:

Federico Stagni, Maarten Litmaath, Latchezar Betev, Christoph Wissing, Tommaso Boccali, Daniele Spiga, James Letts, Danilo Piparo, Dirk Hufnagel, Andrej Filipcic, Lincoln Bryant

Extras

Use of Marconi A2 (CINECA)

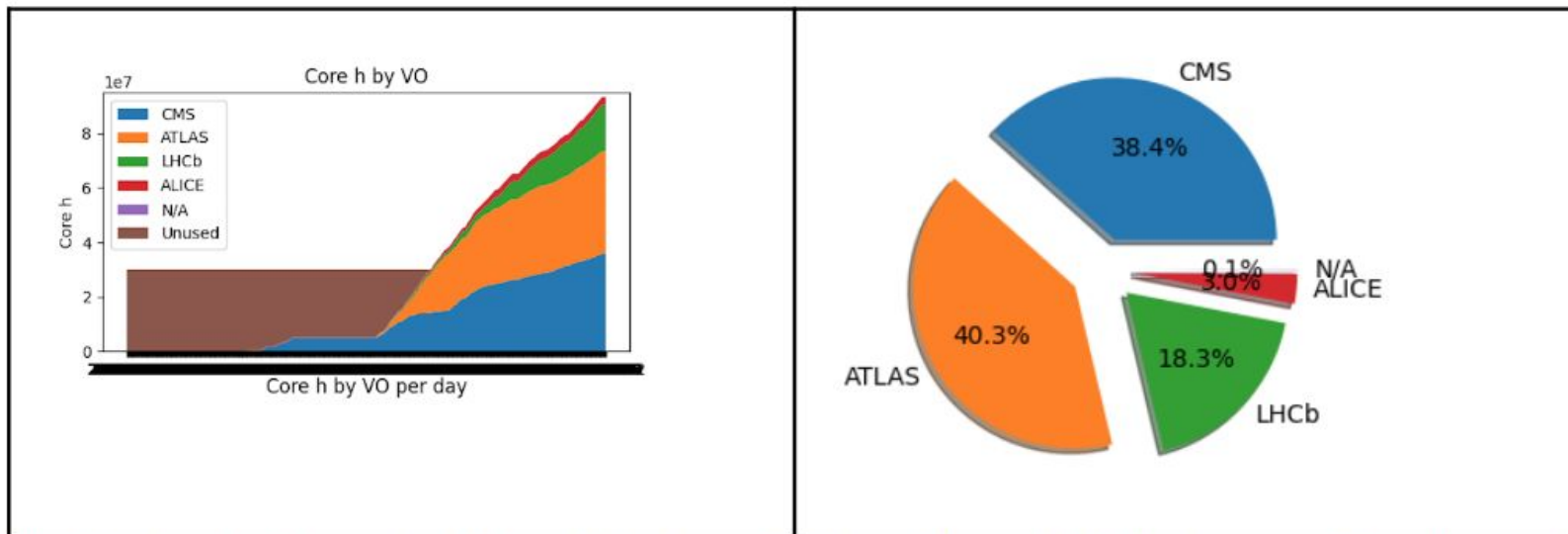


Fig. 7. Left: total utilization of the Marconi A2 from April 2019 to February 2021. The brown area shows the amount of remaining grant (30 Million core hours). (right) utilization by experiments as a fraction of the utilized 93 Million core hours.

From <https://pos.sissa.it/378/003/pdf>