Draft v0.0, Aug 8, 2019

ATLAS HPC Data Processing and Simulation

D.Benjamin, A.Filipcic, A.Klimentov

Abstract

Experiments at the Large Hadron Collider require data intensive processing and traditionally did not use HPCs till Run2. Before 2016 the ATLAS experiment at the LHC was using less than 10 million hours of walltime at HPCs, while over an exabyte of data was processed annually on the grid. A large increase in data volume and data complexity at the LHC in 2016 created a shortage of computing cycles, and HPC systems stepped in to help the LHC achieve its physics goals. ATLAS was able to utilize about half a billion hours of walltime usage on HPCs during 2017/2018. This is a huge increase in usage over a few years - required numerous innovations and improvements. This paper describes the use of HPCs worldwide by ATLAS, primarily for simulations, and specifically focus on how the HPCs are integrated with the workflow management and data management systems, and our vision for future HPC evolution and new architectures and HPCs role for HEP.

1. Introduction

ATLAS [1] has successfully integrated HPC facilities with distributed computing and used HPCs for more than 5 years [2]. HPC facilities are integrated via different technologies because of the unique nature of HPCs, such as access to external network and WLCG storage endpoints [see Appendix A for a complete list of facilities]. Several approaches have been developed and commissioned to address differences in the access, authorization and service requirements of the various HPC centers. The flexibility of ATLAS processing, workload management system and data management system has enabled ATLAS to run most of the workflows on HPC. ATLAS uses the HPCs in the following ways:

- grid-like execution when an HPC provides the external connectivity on the nodes and enables access to cvmfs software area either by directly mounting cvmfs on the nodes or using parrot with access to squid service. Any ATLAS workflow can execute on such HPCs. Typically ATLAS agents (ARC-CE or Harvester [3,4]) are used as a site service for job submission and control. Both payload push and pull can be used, although push mode is preferred on HPCs without close Storage Element, where downloads and uploads are managed by a dedicated service on data transfer nodes.
- Limited connectivity execution when the nodes do not have outbound network access. The ATLAS software can either be installed locally on a shared file system or provided through fat containers on HPCs that support singularity or shifter. Such HPCs typically run ATLAS Event Generation or Geant 4 Simulation where conditions data can be stored in a local SQLite file. A dedicated ATLAS SW infrastructure is used for job submission and control. Those services can be installed locally on an HPC site or they can be used remotely through ssh connection to the batch system and sshfs to manage input and output files.

Some HPCs provide a subset of the services that are required on grid sites, eg. squid, cvmfs, CE, disk cache. If ATLAS is able to transparently use any of these, the HPC services are then described in detail in AGIS.

2. ATLAS Workflows on HPC

Workflow	CPU s/event	Input/event	Output/Event	cores	CPU Arch
Event Generation	1-5000	<0.001	100kB	1, some multi	x86_64,Power9
G4 Simulation	200-1000	100kB	1MB	multi-core	x86_64,Power9,

					ARM
MC Reconstruction	50-100	20MB pile, 5MB overlay	0.5MB	multi-core	x86_64
Data Reconstruction	50-100	1-2MB	0.5MB	multi-core	x86_64
Derivations	1	0.5MB	0.01-0.05MB	multi-core	x86_64
Analysis	0.01-1	0.01-0.05MB	0.001MB	Single, multi-core	x86_64
Fast Chain	*New WF*	*New WF*	*New WF*	multi-core	x86_64

3. Payload Execution Requirements

a. Software Distribution

i. RHEL compatible OS - cvmfs mounted on nodes or parrot or containers

ii. Containers or compatibility libraries for other x86-64 operating systems tested on SLES, Debian, Ubuntu, Gentoo, CoreOS, Cray Linux (any modern OS)

iii. Dedicated compilation required for not x86_64 instructions sets like Power 9.

ATLAS would benefit from dedicated/shared non x86_64 systems to compile ATLAS software on.

b. Database access:

i. SQLite partial conditions dump for selected IOV. Transparent for Geant4 simulation, non-trivial and large for MC.

ii. data (re)processing case for Leadership Class Facilities (LCF) will be addressed when it will be feasible.

iii. Access to Frontier through squid on HPCs that provide it. This is required to run ATLAS reconstruction workflow.

If HPC centers can provide such a service for database access then a larger variety of workflows requiring experiment Metadata could be run on these resources. (Note - we need some access to the Conditions DB files that come through CVMFS /cvmfs/atlas-condb/…. I see files added to it from 2019).

c. Parallel jobs

i. Node level parallelism - multi-process or multi-threaded event task farm distributing work within a node
ii. Machine level parallelism - Multi-node event parallelism distributing work on several hundred nodes
   - MPI - Yoda running in production since 2016 [5]
   - TCP/IP - Raythena prototype integrating node-level and machine-level parallelism
   - Several HPCs limit number of "small jobs" and may provide a discount for running massive multi-node parallel jobs. ATLAS has demonstrated packing several grid jobs into a single batch submission [4].

4. Authentication and Authorization

- X509 - where the HPCs do not require custom authorization. With remote CE or Harvester, the grid certificates are used to access the CE and PanDA, the HPC login credentials are used for job submission.
- Multi-factor authentication - Access credentials are typically short lived and need to be manually extended periodically. Local Harvester is typically used for job submission in such cases.
- AAI and other methods are likely to become the standard way to access the resources in the future, the ATLAS services will need to support them in the future

5. Data Distribution

a. Input & Output Files

The data on the grid is typically accessed directly with xrootd protocol or copied to/from local node scratch space (working directory). The remote access is mostly used for analysis jobs. HPCs typically have a large shared posix filesystem which is used to transfer data between remote Storage Elements and HPC. The payload running on the nodes reads directly from the shared filesystem and writes the output files there as well. In some cases, the local scratch storage on nodes or a Burst Buffer is used to increase I/O performance for temporary output files.

The ATLAS agent (ARC-CE or Harvester resources broker) receives the payload description including a list of input files. This list is then processed by the agent's data-transfer subsystem. Dedicated data-transfer nodes can be transparently used and tuned for transfer performance.

The completed payloads are processed by the ATLAS agent and the output files are then transferred to the pre-assigned remote Storage Elements.

ATLAS software typically makes many calls to open/touch small files/libraries. This has proven to be problematic for the shared file system meta-data servers found at HPC centers. The use of software containers has reduced this problem and is required unless ATLAS significantly improves how it access the libraries that it runs.

b. Storage Services, Elements[1] and Caches

HPCs typically do not have close Storage Elements (SE), that would allow direct access to remote data on the compute nodes. There are cases when WLCG T1 or T2 facilities are colocated with HPC centers. In case of PizDaint, the HPC prefers to transfer the data to shared burst filesystem partition prior to payload execution, even with local pledged Tier-2 storage. This improves the data access throughput and has stricter control of the data flow between the SE and shared filesystem. The shared filesystem is then used as the input file cache, which is managed by ARC-CE or Harvester, the oldest input files are deleted automatically when the cache usage is too high. The same space is also to store the output files, and those are automatically removed when transferred to final SE destination.

R&D to use XCache with xrootd protocol on the shared filesystem area or on a nearby dedicated storage is conducted by ATLAS.

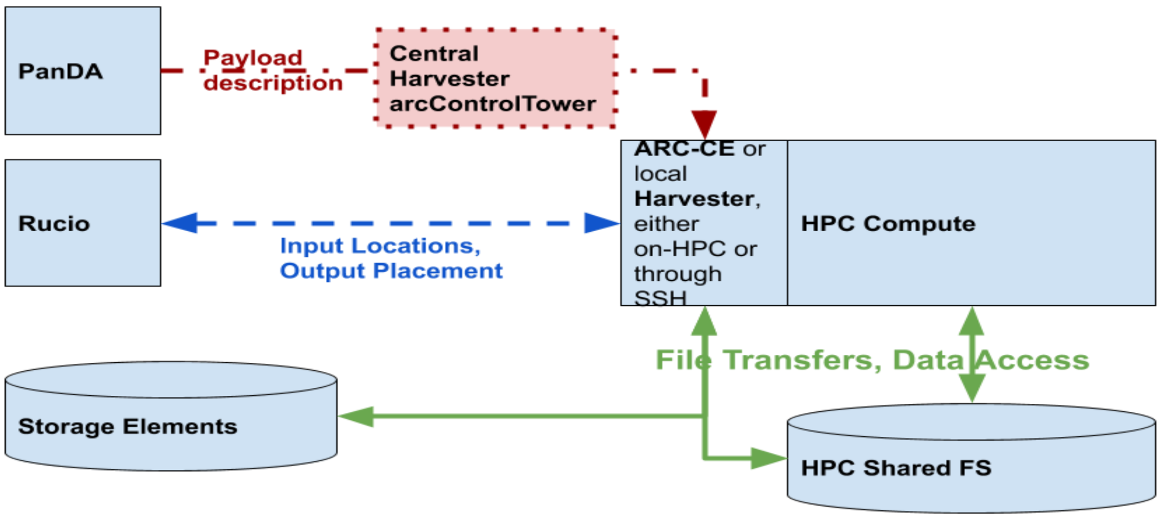
The HPC center may provide a permanent storage in the future. It is assumed that it will be known to the ATLAS distributed data management system (Rucio) and described as a Rucio Storage Element (RSE).

6. Payload Distribution and Execution Control

a. ATLAS Central Services

ATLAS central services have four major components : data management (Rucio), Workflow management (ProdSys2), Workload management services (PanDA) and information system (AGIS) [Pic.1]. PanDA has been extended to support full-node and large massively-parallel jobs and delegate the payload heartbeat to custom services such as arcControlTower and Harvester [Pic.2 shows harvester roles in ATLAS Distributed Computing. PanDA / Harvester installation for former OLCF Titan is shown on Pic.3]. HPC facilities are described in ATLAS Information System (AGIS). The development to optimize the HPC resource usage and data transfers is ongoing, since many workflows have been tuned to run well on grid resources. The next step will be optimisation of HPC resources utilisation to minimize data transfer to/from HPC centres

[1] In this context it is Rucio Storage Element

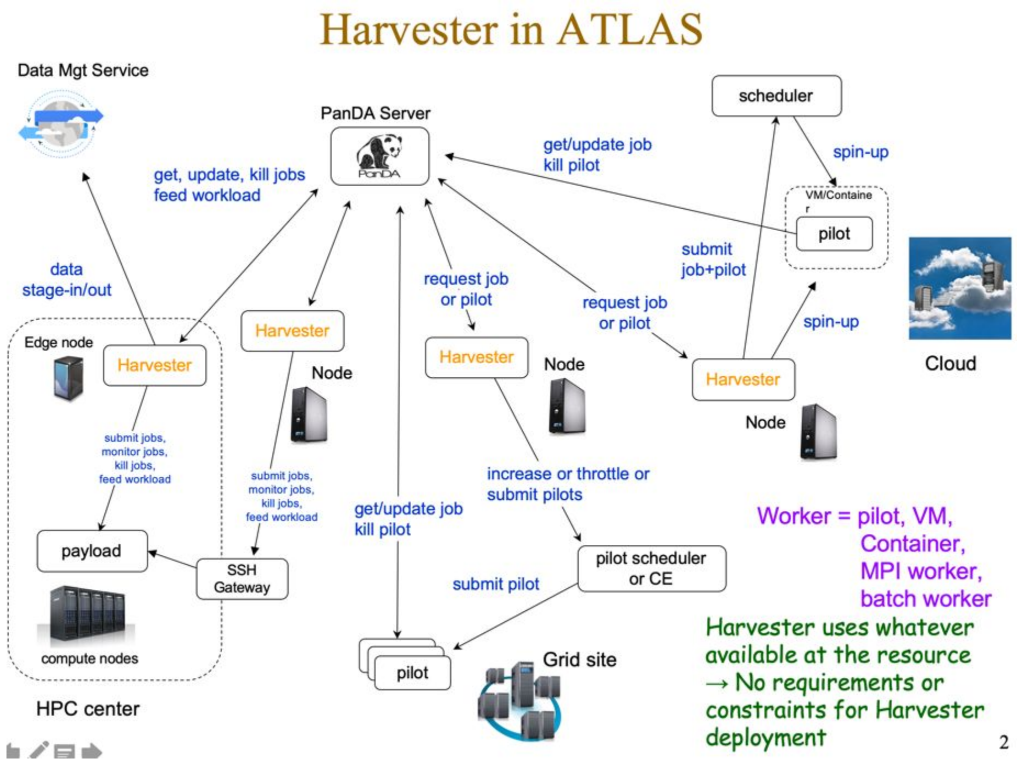

PanDA
Payload description
Central Harvester arcControlTower
ARC-CE or local Harvester, either on-HPC or through SSH
HPC Compute
Rucio
Input Locations, Output Placement
File Transfers, Data Access
Storage Elements
HPC Shared FS

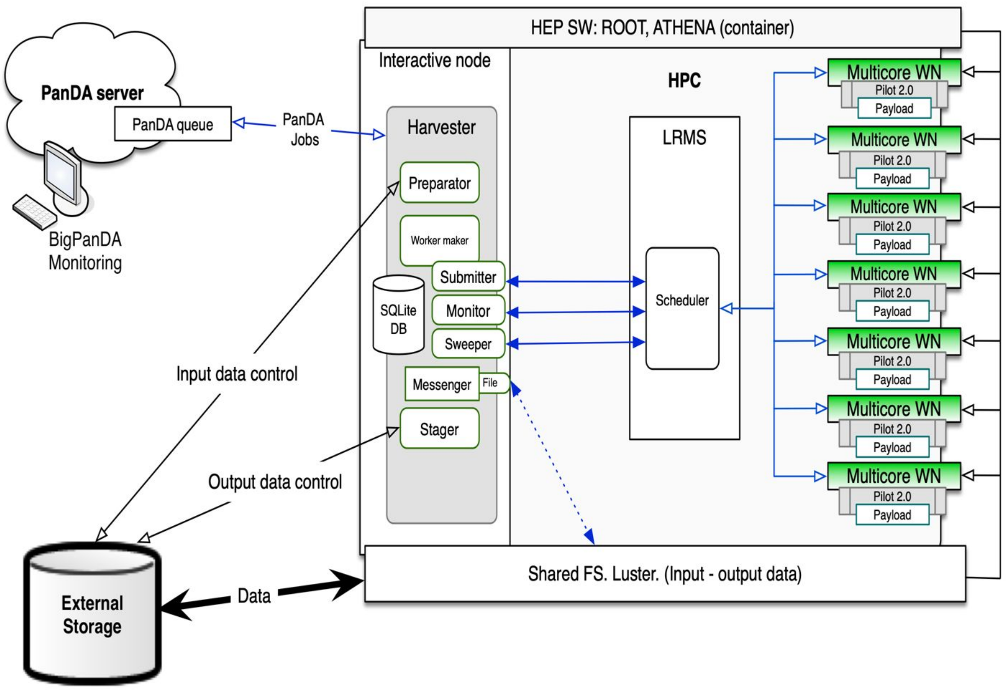
b. Payloads distribution and control

On grid sites, the pilot jobs typically pull the payload from PanDA (payload pull mode) and the input files and software environment are then prepared matching the payload description requirements. The pull mode relies on the running jobs to have the full access to grid storage and to ATLAS central services.

For HPCs with Compute Elements, a Central Harvester Instance is being used to dispatch the payload/jobs to HPCs in the same way as it is done for grid sites, while Condor client or arcControlTower are used to submit the jobs to the Compute Elements in payload push mode. Harvester performs all the communications with PanDA, while the other components take care of job submission, status and completion.

If there is no local or remote Compute Element, a Harvester instance running on a login or edge node of the HPC is being used to submit jobs to the HPC batch system and perform the data transfers.


Harvester in ATLAS
Data Mgt Service
PanDA Server
scheduler
get, update, kill jobs
feed workload
get/update job
kill pilot
spin-up
VM/Container
pilot
submit
job+pilot
data
stage-in/out
request job
or pilot
request job
or pilot
spin-up
Cloud
Edge node
Harvester
Harvester
Node
Harvester
Node
Harvester
Node
submit jobs,
monitor jobs,
kill jobs,
feed workload
submit jobs,
monitor jobs,
kill jobs,
feed workload
increase or throttle or
submit pilots
get/update job
kill pilot
Worker = pilot, VM,
Container,
MPI worker,
batch worker
payload
SSH
Gateway
pilot scheduler
or CE
submit pilot
Harvester uses whatever
available at the resource
→ No requirements or
constraints for Harvester
deployment
compute nodes
pilot
Grid site
HPC center
2


HEP SW: ROOT, ATHENA (container)
PanDA server
PanDA queue
PanDA
Jobs
BigPanDA
Monitoring
Interactive node
HPC
Harvester
Preparator
Worker maker
Submitter
SQLite
DB
Monitor
Sweeper
Messenger
File
Stager
LRMS
Scheduler
Multicore WN
Pilot 2.0
Payload
Input data control
Output data control
External
Storage
Data
Shared FS. Luster. (Input - output data)

7. ATLAS view on future HPC evolution

The next generation of HPCs is evolving from pure computational facilities to resources that can be used for extreme data processing (BigData, HPDA and Artificial Intelligence (Machine and Deep Learning...)). These machines will also provide multi 100Gb/s external connectivity and data processing capacities of several TB/s, thus providing a prime infrastructure for HEP simulation, reconstruction, data processing and analysis. The architecture will however be very different to the usual grid-site:

- Large shared file systems as primary data storage, object stores will be available later (>2022)
- Burst buffers - fast NVMe, SSD dedicated storage for data processing
- >100 core nodes with limited memory (~1GB/core or less)
- Limited outbound throughput on the nodes

The fundamental characteristic of current and next-gen HPCs: to achieve their FLOPs target HPC nodes will have a heterogeneous architecture with most FLOPs being provided by GP-GPU accelerators. The main feature of these new HPC facilities is the fact that large amounts of compute power is delivered by accelerators.

ATLAS C/C++ code was designed and developed initially for single x86 CPU cores. It has been further modified for multicore CPUs. We have also demonstrated that the code can be recompiled for other CPU platforms such as Power9 and ARM. It has not been ported to GP-GPUs or/and new architectures which limits our ability to use the new generation of LCF machines (such as Summit). To use the large number of flops available on GPUs at HPC centres, some of our kernel needs to be reengineered and new algorithmic code needs to be developed. Demonstrators of GPU code have been shown using CUDA to run on dedicated NVIDIA GPUs. (Trigger demonstrator, BNL Fast Calorimeter simulation) but we have no production-ready code. One important challenge is to ensure the SW is sustainable in the long term, hence a development language needs to be used that is independent of the hardware where the software is running. While this is currently possible for Machine Learning applications, general software technologies are not yet at the level of maturity where large re-engineering would make sense.
We are experimenting with toolkits such as Kokkos and SYCL, and will evaluate Intel’s One API once it becomes available. We expect these technologies to mature in the next 2-3 years, in time for the LHC Phase2.

Running ATLAS code as-is on next-generation HPCs will result in an order-of-magnitude performance penalty. Even after porting ATLAS applications to heterogeneous CPU/GPU execution, to optimize the HEP workflows on HPCs we will still need to take into account:

- Data/Event pre-placement prior to payload execution
- Using custom data transfer services (for example Globus)
- Multi-threading and multi-node payload execution

- Limited I/O metadata operations (eg stat calls, other filesystem operations)

Custom, user provided private services are not appreciated on the HPCs, a common general-purpose services will be encouraged, so a common HEP plan is needed to address this.

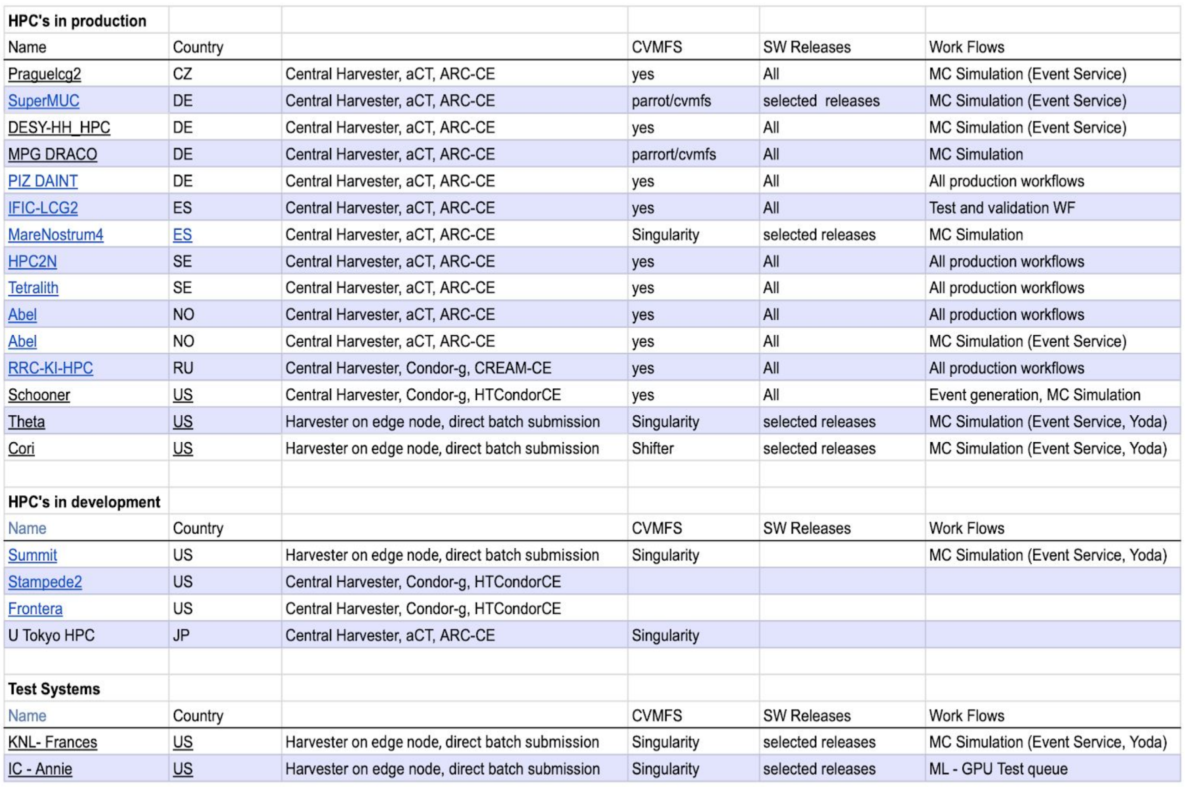
It is now accepted that ATLAS simulation must undergo a transformation to be able to make good use of current and future computer architectures. Only by using hardware efficiently will ATLAS be able to achieve the massive computing throughput that will be needed in the next decade. HPC facilities are a particular target because of the massive computing power they can bring to bear. R&D efforts are underway to evaluate the feasibility of running parts of simulation on GPUs.

Fully optimized use of HPC facilities is a long-term goal. Early access to (pre)exascale generation of HPC facilities will provide an important stimulus to this work. Access to facilities coupled with collaborative help in the transformation of ATLAS code would be a major scientific contribution to the physics discoveries of the next ten years. HPC planners acknowledge HEP experiments (and ATLAS in particular) and the importance of our compute intensive science (thanks to our previous effort of HPC/HTC integration). We must live with their requirements and limitations :

- They rely on accelerators, so **we must use the accelerators**
- Data intensive computing with them may be a challenge

We need to win **our place at the table for future design** of the HPC landscape and there are some promising signs of more attention to LHC BigData from EuroHPC.

Appendix A. HPC Facilities used by ATLAS



HPC's in production					
Name	Country		CVMFS	SW Releases	Work Flows
Praguelcg2	CZ	Central Harvester, aCT, ARC-CE	yes	All	MC Simulation (Event Service)
SuperMUC	DE	Central Harvester, aCT, ARC-CE	parrot/cvmfs	selected releases	MC Simulation (Event Service)
DESY-HH_HPC	DE	Central Harvester, aCT, ARC-CE	yes	All	MC Simulation (Event Service)
MPG DRACO	DE	Central Harvester, aCT, ARC-CE	parrort/cvmfs	All	MC Simulation
PIZ DAINT	DE	Central Harvester, aCT, ARC-CE	yes	All	All production workflows
IFIC-LCG2	ES	Central Harvester, aCT, ARC-CE	yes	All	Test and validation WF
MareNostrum4	ES	Central Harvester, aCT, ARC-CE	Singularity	selected releases	MC Simulation
HPC2N	SE	Central Harvester, aCT, ARC-CE	yes	All	All production workflows
Tetralith	SE	Central Harvester, aCT, ARC-CE	yes	All	All production workflows
Abel	NO	Central Harvester, aCT, ARC-CE	yes	All	All production workflows
Abel	NO	Central Harvester, aCT, ARC-CE	yes	All	MC Simulation (Event Service)
RRC-KI-HPC	RU	Central Harvester, Condor-g, CREAM-CE	yes	All	All production workflows
Schooner	US	Central Harvester, Condor-g, HTCondorCE	yes	All	Event generation, MC Simulation
Theta	US	Harvester on edge node, direct batch submission	Singularity	selected releases	MC Simulation (Event Service, Yoda)
Cori	US	Harvester on edge node, direct batch submission	Shifter	selected releases	MC Simulation (Event Service, Yoda)
HPC's in development					
Name	Country		CVMFS	SW Releases	Work Flows
Summit	US	Harvester on edge node, direct batch submission	Singularity		MC Simulation (Event Service, Yoda)
Stampede2	US	Central Harvester, Condor-g, HTCondorCE			
Frontera	US	Central Harvester, Condor-g, HTCondorCE			
U Tokyo HPC	JP	Central Harvester, aCT, ARC-CE	Singularity		
Test Systems					
Name	Country		CVMFS	SW Releases	Work Flows
KNL- Frances	US	Harvester on edge node, direct batch submission	Singularity	selected releases	MC Simulation (Event Service, Yoda)
IC - Annie	US	Harvester on edge node, direct batch submission	Singularity	selected releases	ML - GPU Test queue

Glossary and abbreviations

ArcCE - ARC Compute Element (CE) is a Grid front-end on top of a conventional computing resource and HPC computing resources.
ArcCT - ARC Control Tower: A flexible generic distributed job management framework.
IOV - Interval of Validity - It is an interval in time (UTC timestamp) or an interval in run number/luminosity block number
Harvester - is a resource-facing service between Workflow Management System (WFMS) and the collection of pilots for resource provisioning and workload shaping. It is a lightweight stateless service running on a VObox or an edge node of HPC centers to provide a uniform view for various resources.
HPC - High-performance computing (HPC) is the use of parallel processing for running advanced application programs efficiently, reliably and quickly. The term applies

especially to systems that function above a teraflop or 1012 floating-point operations per second.
LCF - Leadership Computing Facility. These systems are high end computers that are among the most advanced in the world for solving scientific and engineering problems.
MC - Monte-Carlo method is a (computational) method that relies on the use of random sampling and probability statistics to obtain numerical results for solving deterministic or probabilistic problems.
MPI - The message passing interface (MPI) is a standardized means of exchanging messages between multiple computers running a parallel program across distributed memory.
PanDA - The PanDA Production ANd Distributed Analysis system has been developed by ATLAS since summer 2005 to meet ATLAS requirements for a data-driven workload management system for production and distributed analysis processing capable of operating at LHC data processing scale.
Raythena - a vertically integrated scheduler for ATLAS applications on heterogeneous distributed resources.
Rucio - Rucio is a project that provides services and associated libraries for allowing scientific collaborations to manage large volumes of data spread across facilities at multiple institutions and organisations.
Run2 (LHC Run2) - The second phase of LHC running period, 2015-2018
Yoda - Python software using MPI communication to as an Event Service implementation for HPC's.

Bibliography

1. The ATLAS Collaboration, G. Aad et al., "The ATLAS Experiment at the CERN Large Hadron Collider", Journal of Instrumentation, Vol. 3, S08003, 2008.
2. K.De et al, Integration of Panda Workload Management System with Supercomputers, ISSN 1547-4771, Physics of Particles and Nuclei Letters, 2016, Vol. 13, No. 5, pp. 647–653
3. Smirnova O., Kónya B., Cameron D., Nilsen J.K., Filipčič A. ARC-CE: updates and plans, Computer Research and Modeling, 2015, vol. 7, no. 3, pp. 407-414
4. T.Maeno et al, Harvester : an edge service harvesting heterogeneous resources for ATLAS, ATL-SOFT-PROC-2018-029
5. P.Calafiura et al, Fine grained event processing on HPCs with the ATLAS Yoda system, ATL-SOFT-PROC-2015-004