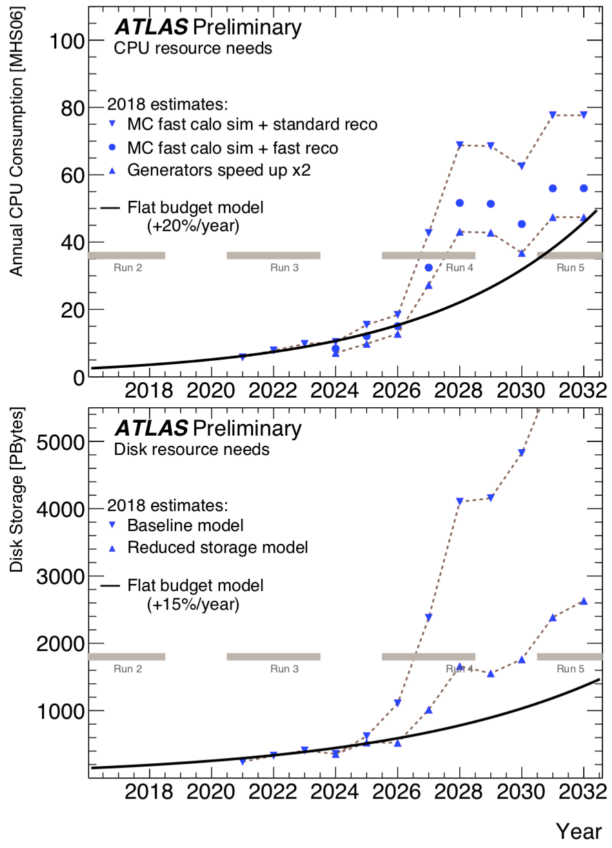
HARNESSING THE POWER OF SUPERCOMPUTERS USING THE PANDA PILOT 2 IN THE ATLAS EXPERIMENT

Dr Paul Nilsson (Brookhaven National Laboratory) on behalf of the ATLAS collaboration

Introduction

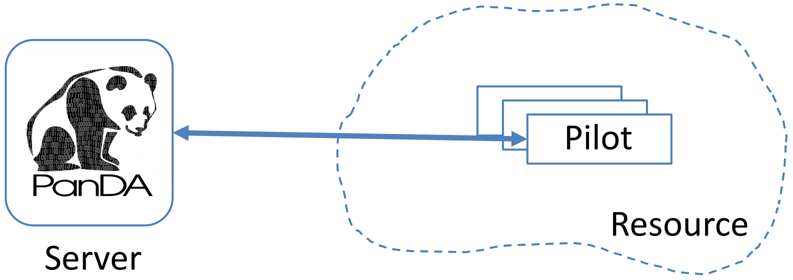
- In the HL-LHC era, both ATLAS and CMS will produce around 10 times more data than now
  - Storage and computing needs continue to grow at a much higher pace than what a flat budget allows for
- At the same time, the IT landscapes, computing infrastructures and funding models are changing
  - National science programs are consolidating computing resources and encourage using HPCs as well as cloud services
- ATLAS is using various heterogeneous resources since several years
  - Successful integration of grids, clouds and HPCs into the ATLAS workflow management system
- But heterogeneous resources are not always tailored for ATLAS workloads
  - New technologies have recently been introduced to address and overcome some of the most challenging limitations while at the same time reducing operational manpower



Reference: http://cern.ch/go/g7mK

Paradigm shift

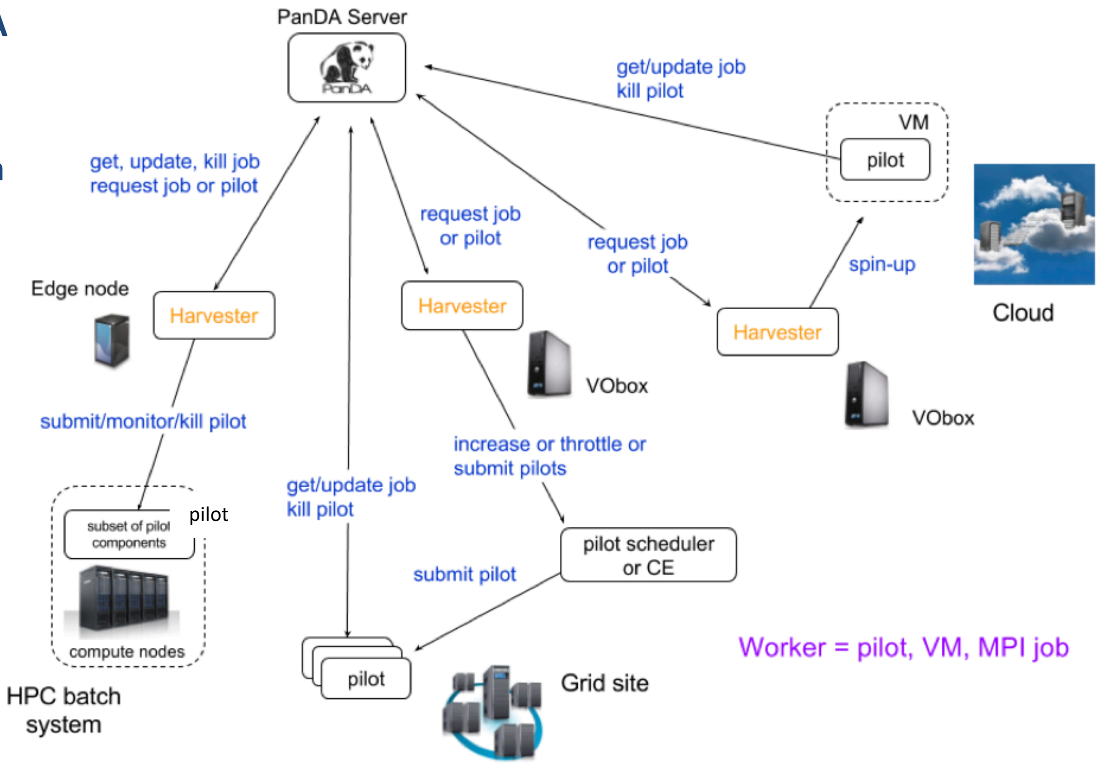
- ATLAS uses PanDA as the workload management system to control all resources
  - Generic factories submit Pilots to the batch systems
  - A Pilot occupies the slot, asks PanDA server for a job, executes it and monitors the job throughout its lifetime and report back relevant metrics to the server as well as handles the file transfers



- The Server-Pilot paradigm worked well for non-stop running on the grid with 250k+ cores for a decade
- But on opportunistic resources such as HPCs it worked less well
  - Different edge services and operational policies at different HPCs led to major challenges with an already aging Pilot architecture leading to different Pilot versions on the grid and HPCs
  - Too many manual interventions to effectively fill the available CPU resources
- New solutions were needed to overcome these challenges
  - The Harvester and Pilot 2 projects were launched in 2016

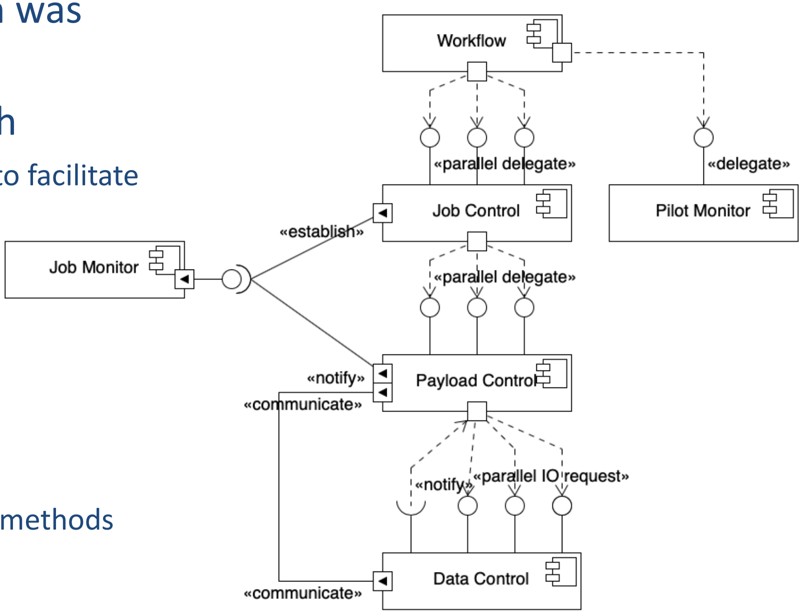
Harvester in the PanDA system

- A resource-facing service between **PanDA Server** and **Pilots** for resource provisioning and workload shaping
- A lightweight stateless service running on a VObox or an **edge node of HPC centers** to provide a uniform view for various resources
- Modular design for different resource types and workflows
- Coherent implementations for HPCs
- Timely optimization of CPU allocation among various resource types and removal of batch-level partitioning
- Better resource monitoring
- Tight integration between PanDA system and resources for new workflows



Pilot 2

- A complete rewrite of the original PanDA Pilot which was used in the ATLAS Experiment for over a decade
- Its architecture follows a component-based approach
  - Evolves the system according to modern functional use-cases to facilitate coming feature requests from PanDA users
  - Improves system flexibility
  - Enables a clear workflow control
- Supports different workflows
  - Normal grid mode, HPC mode, stage-in mode
- Uses containers
  - Payload executed in container launched by the Pilot (different methods supported) using images from cvmfs or DockerHub
  - Pilot itself may be launched in a container
- Offers simplified APIs
  - While any part of the code can be imported, some functionality can be accessed by special APIs streamlined for external use to bypass complex function calls or data structures that otherwise would need to be defined
  - Harvester is e.g. using Pilot 2 Data API for file transfers on/off HPCs



Data API

```
StageInClient()
.transfer()
StageOutClient()
.transfer()
```

Communicator API

```
JobClient()
.get_job()
.get_jobs()
.update_job()
.update_jobs()
.get_event_range()
.update_event_range()
```

Services API

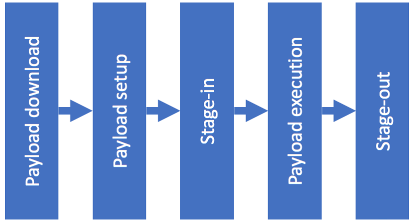
```
Benchmark()
.get_command()
.get_filename()
.get_results()
MemoryMonitor()
[same as Benchmark()]
.get_linear_fit()
```

Pilot HPC workflows

- Depending on the type of the HPC, Pilot may either run payloads on the worker nodes like on any grid site, or in a simplified mode via a plugin
  - Grid-like execution when an HPC provides external connectivity on the nodes and enables access to external software areas
  - Limited connectivity execution when the nodes do not have outbound network access (a plugin may be needed)
  - On Titan, a plugin was developed where Pilot acts like an MPI application under the control of Harvester and runs a set of jobs in an assembly
    - Harvester takes care of prestaging input data, stage-out of output (using the Pilot Data API) and communication with the PanDA server
    - Pilot intercommunicates with Harvester through the shared filesystem and reads the job definitions from pre-placed files and declares outputs for later processing by Harvester
- Other HPC plugins are being planned e.g. for Summit
  - Pending suitable GPU payloads (ML user jobs exist)
- Raythena/Yoda-Droid
  - New workflows where Pilot runs ATLAS Event Service jobs on HPC

Grid-like workflow on generic HPC

- Collect job definition from a pre-placed JSON file or download from PanDA server
- Prepare for execution: pre-placement of input data in shared file system or download directly from Storage Element



- Report job status directly to PanDA server or create special JSON file to be picked up by ARC Control Tower or Harvester

Generalized workflow on Titan-like HPC

- Collect job definition from a pre-placed JSON file
- Prepare for execution: pre-placement of input data in high speed transient storage associated with the computing node; RAM-disk, Burst Buffer, SSD
- Setup environment; incl. platform specific preparations
- Execute payload
- Publish job report in a JSON file, which includes state of payload and other metrics
- Process job report
- Transfer output data to shared file system
- Archive and store payload logs in shared file system
- Declare files for later stage-out by Harvester

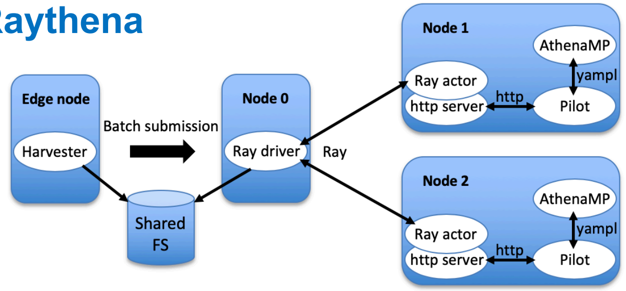
HPC tools

- Two tools are being developed / evolving for use with **ATLAS Event Service** workflow on HPCs
- **Raythena** is new Ray-based tool that provides a highly scalable solution
  - Ray driver in charge of communications with Harvester via shared file system
  - Ray actor communicates with Pilot which is running the actual payload (multi-process)
  - With Pilot running in full mode on the worker node, we get all the functionalities provides by the Pilot (especially job monitoring and error handling)
- The already existing **Yoda-Droid** tool is a stand-alone application used in production since 2016
  - MPI-enabled job payload manager which plugs into the event service model for running work on HPCs under the supervision of Harvester
  - Work is underway to migrate the tool into Pilot itself as a dedicated HPC workflow

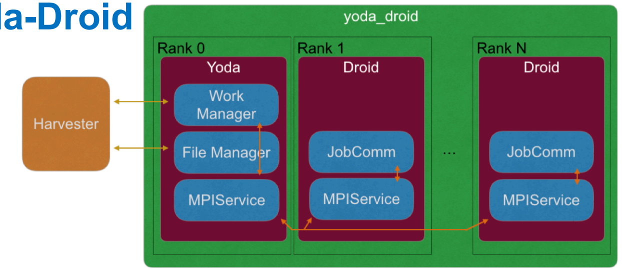
ATLAS Event Service

An approach to event processing capable of the finest granularity in all steps of the computing chain, by decoupling processing from the chunkiness of files, streaming events into a worker and streaming the outputs away in a quasi-continuous manner

Raythena

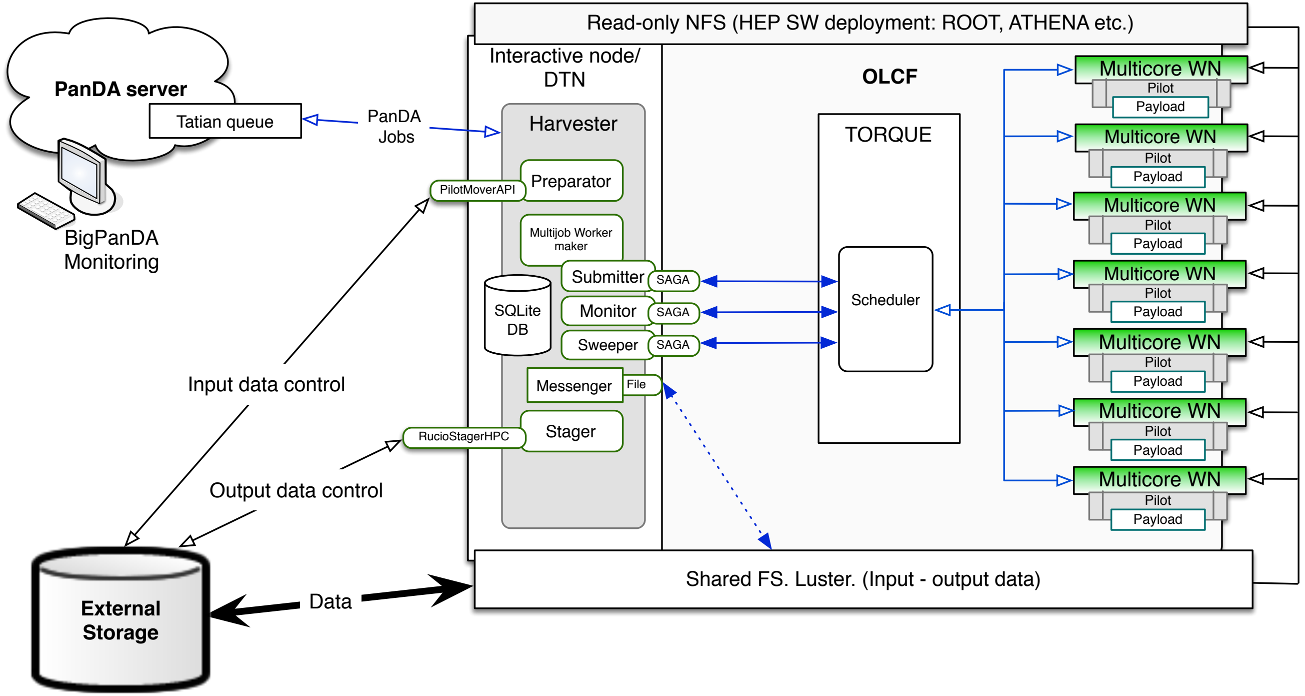


Yoda-Droid



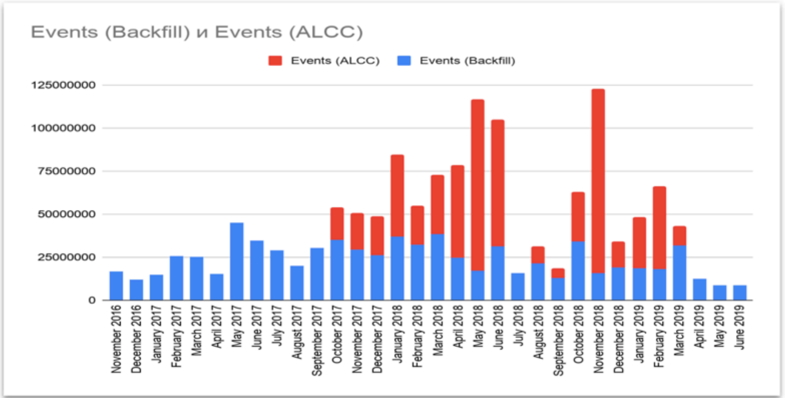
Harvester and Pilot on OLCF

- Harvester handles all communications with PanDA server, as well as file transfers using Pilot Data API
- Special Pilot plugin for Titan running on the worker nodes
- Pilot communicating with Harvester via the shared file system

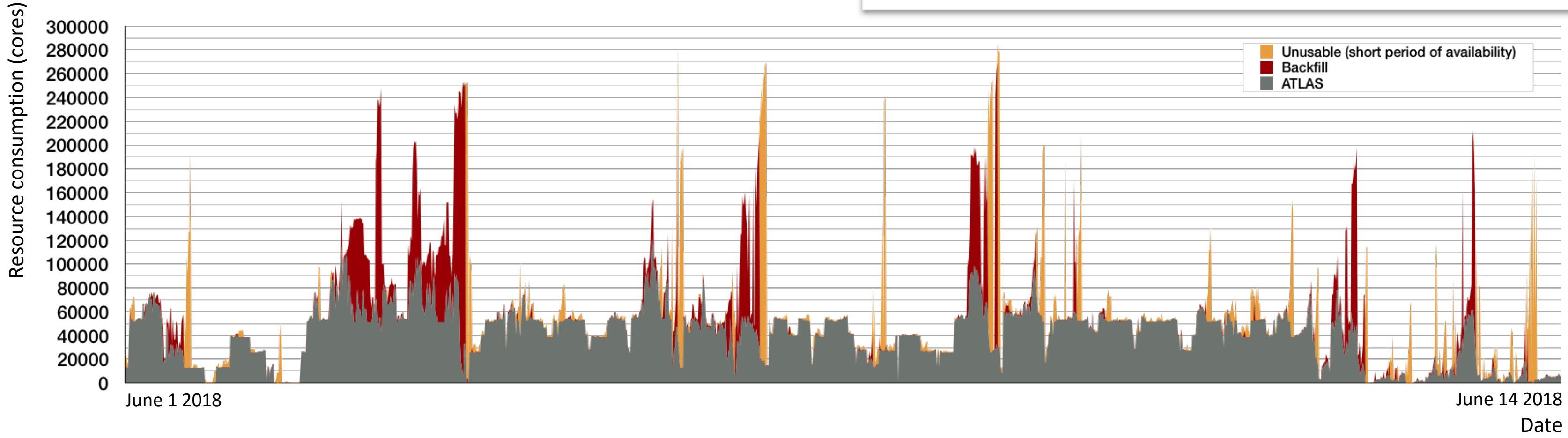


ATLAS jobs on OLCF (Titan)

- 1,4 billion events were simulated at Oak Ridge LCF during production phase (2016-2019) in backfill mode
  - A tailormade version of the original PanDA Pilot was used between 2016-mid 2018, while the new common Pilot 2 version was used from mid 2018 to 2019 until Titan was retired



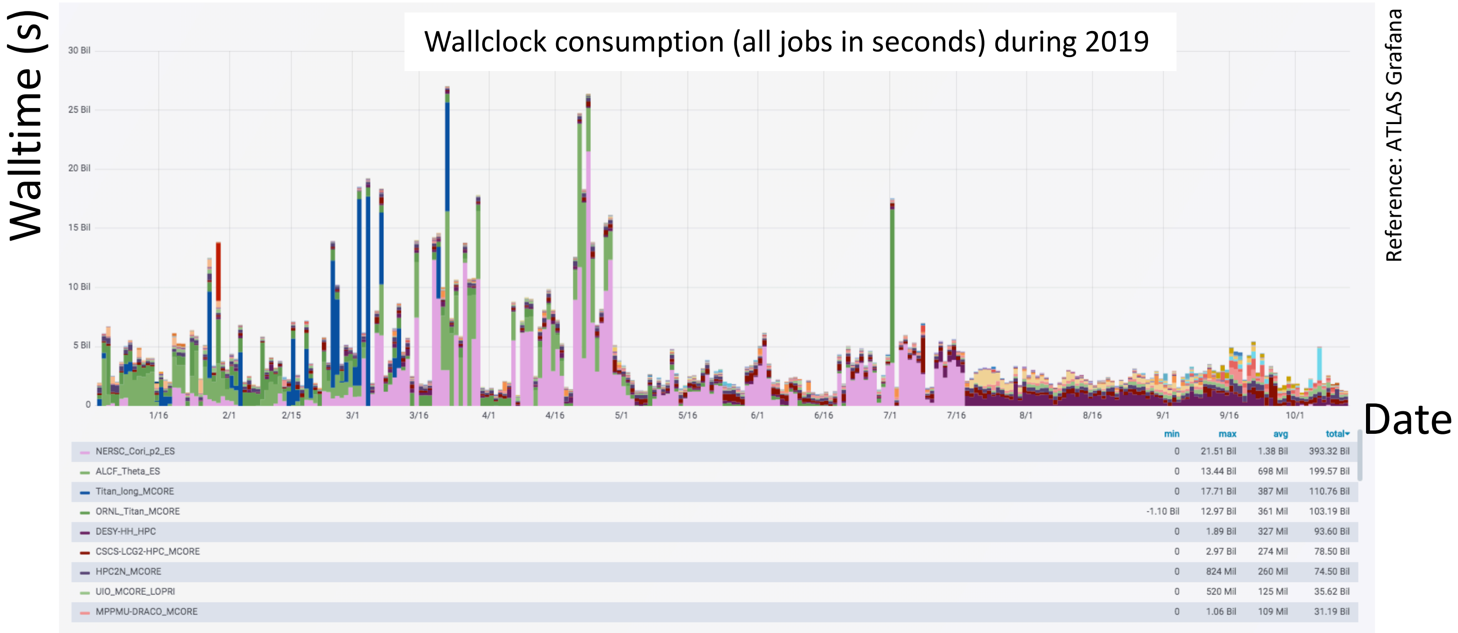
Reference: monit.cern.ch



Reference: D. Oleynik and OLFC

ATLAS HPCs

- ATLAS was able to utilize about half a billion hours of walltime usage on HPCs during 2017/2018, 0.34 billion hours during 2019 on 20+ HPCs (list below is cut)
- Pilot 2 was used either directly with Harvester (on especially US HPC resources) or in combination with the ARC Control Tower on grid-like HPC resources and other HPCs that do not use Harvester directly
  - In total 15 HPCs with Harvester/aCT and five with local Harvester



	min	max	avg	total
NERSC_Cori_p2_ES	0	21.51 Bil	1.38 Bil	393.32 Bil
ALCF_Theta_ES	0	13.44 Bil	698 Mil	199.57 Bil
Titan_long_MCORE	0	17.71 Bil	387 Mil	110.76 Bil
ORNL_Titan_MCORE	-1.10 Bil	12.97 Bil	361 Mil	103.19 Bil
DESY-HH_HPC	0	1.89 Bil	327 Mil	93.60 Bil
CSCS-LCG2-HPC_MCORE	0	2.97 Bil	274 Mil	78.50 Bil
HPC2N_MCORE	0	824 Mil	260 Mil	74.50 Bil
UIO_MCORE_LOPRI	0	520 Mil	125 Mil	35.62 Bil
MPPMU-DRACO_MCORE	0	1.06 Bil	109 Mil	31.19 Bil

ATLAS and the next generation HPCs

- Next generation HPCs are evolving from pure computational facilities to resources for extreme data processing
  - Big Data, High Performance Data Analysis and Artificial Intelligence (Machine and Deep Learning)
- Architecture will be very difference from the usual grid-sites
  - Primary data storage will be large shared file systems
  - Burst buffers; fast NVMe, SSD dedicated storage for data processing to cope with I/O intensive payloads
  - Nodes with over 100 cores with limited memory (1 GB/core or less)
  - Limited outbound throughput on the nodes
- A few examples
  - Most of the computing power will be provided through heterogeneous architectures mixing CPUs with GPU or FPGA accelerators (EU, US)
  - A future exascale EuroHPC is planned to be based on non X86_64 CPUs
- ATLAS C/C++ code was initially designed and developed for single x86 CPU cores but has been modified for multicore CPUs
  - The code can be recompiled for other CPU platforms as well such as Power9 and ARM
  - But to use the large number of flops available on GPUs at HPC centers, some of the kernel needs to be reengineered and new algorithmic code needs to be developed

Conclusions

- Compared to present ATLAS levels, an order of magnitude more data is expected in the looming HL-LHC era
- Innovation is ongoing to further improve the PanDA system with a strong focus on supporting workflows on HPCs
- ATLAS has been using HPCs in production for over five years, primarily for Monte Carlo simulations
    - ATLAS was able to utilize about half a billion hours of walltime usage on HPCs during 2017/2018
    - To handle this huge increase in usage, numerous innovations and improvements needed to be done
- Harvester and Pilot 2 are new tools designed with HPCs in mind
    - Harvester is a resource-facing service between PanDA Server and Pilots for resource provisioning and workload shaping
    - Pilot 2 is a complete rewrite of the original PanDA Pilot which was used in the ATLAS Experiment for over a decade, responsible for executing payloads on worker nodes
- Pilot 2 has so far been used on 15 HPCs with Harvester/ARC Control Tower and on five with local Harvester
- Fully optimized use of existing and future HPC facilities is a long-term goal which ATLAS is consistently working towards