ATLAS production at OLCF 2018

Danila Oleynik, UTA

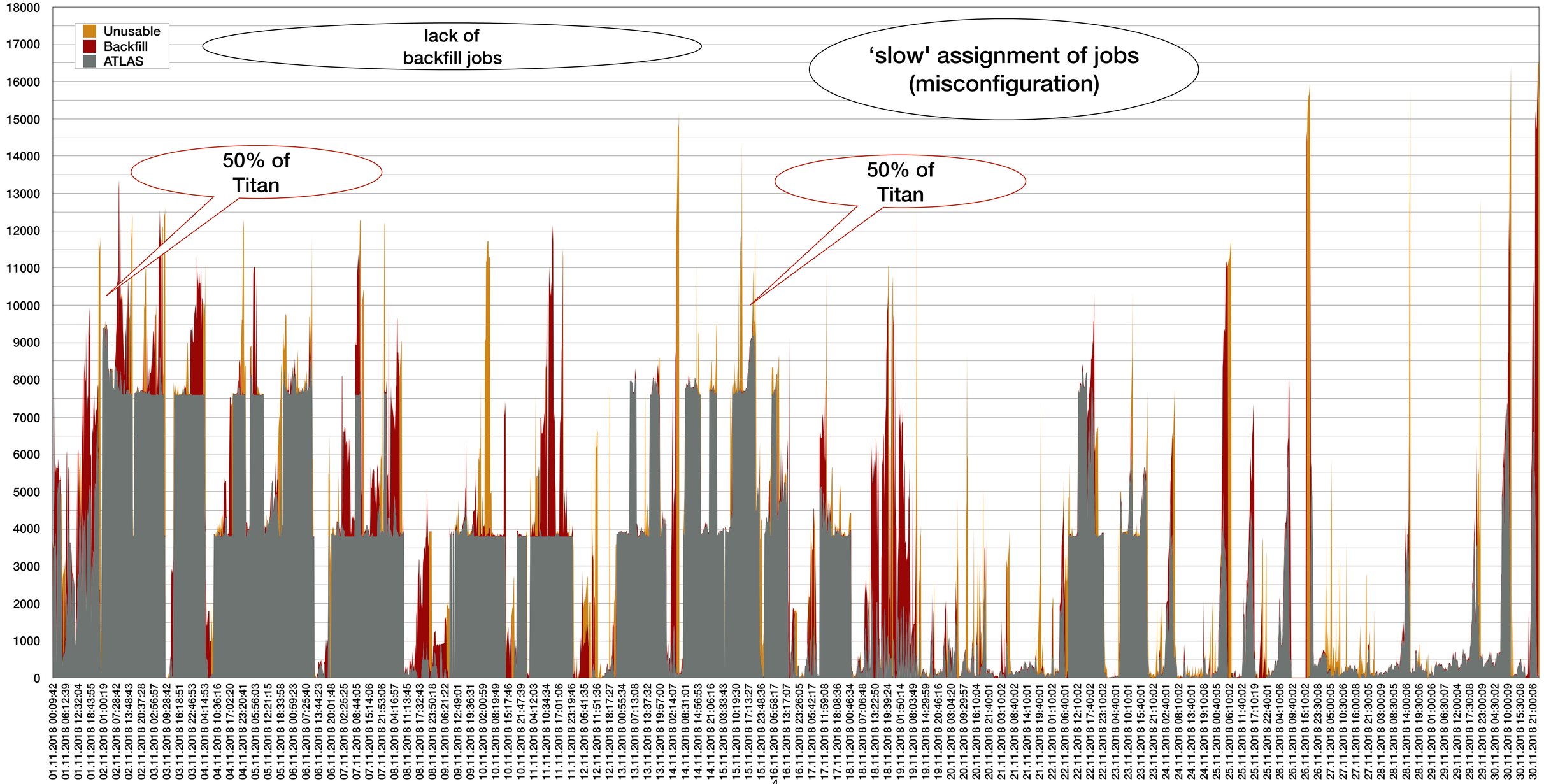
PanDA tools at OLCF in 2018

- During initial RnD in BigPanDA project ‘MultiJob Pilot' application was implemented
  - Based on code of PanDA Pilot
  - Interacts with batch system scheduler to perform execution of a set of jobs simultaneously
  - Workflow of some subcomponents was modified, but not general architecture
- Worked as “proof of concept” but with significant limitations by scale: not more than 350 jobs in one set (pilot)
  - Oriented for “backfill" consumption
  - Difficulties in long-term support, not very common
  - Used initially for ALCC allocation consumption as well (after a set of improvements to manage the bigger size of submissions)
- Towards to Pilot 2 and Harvester projects

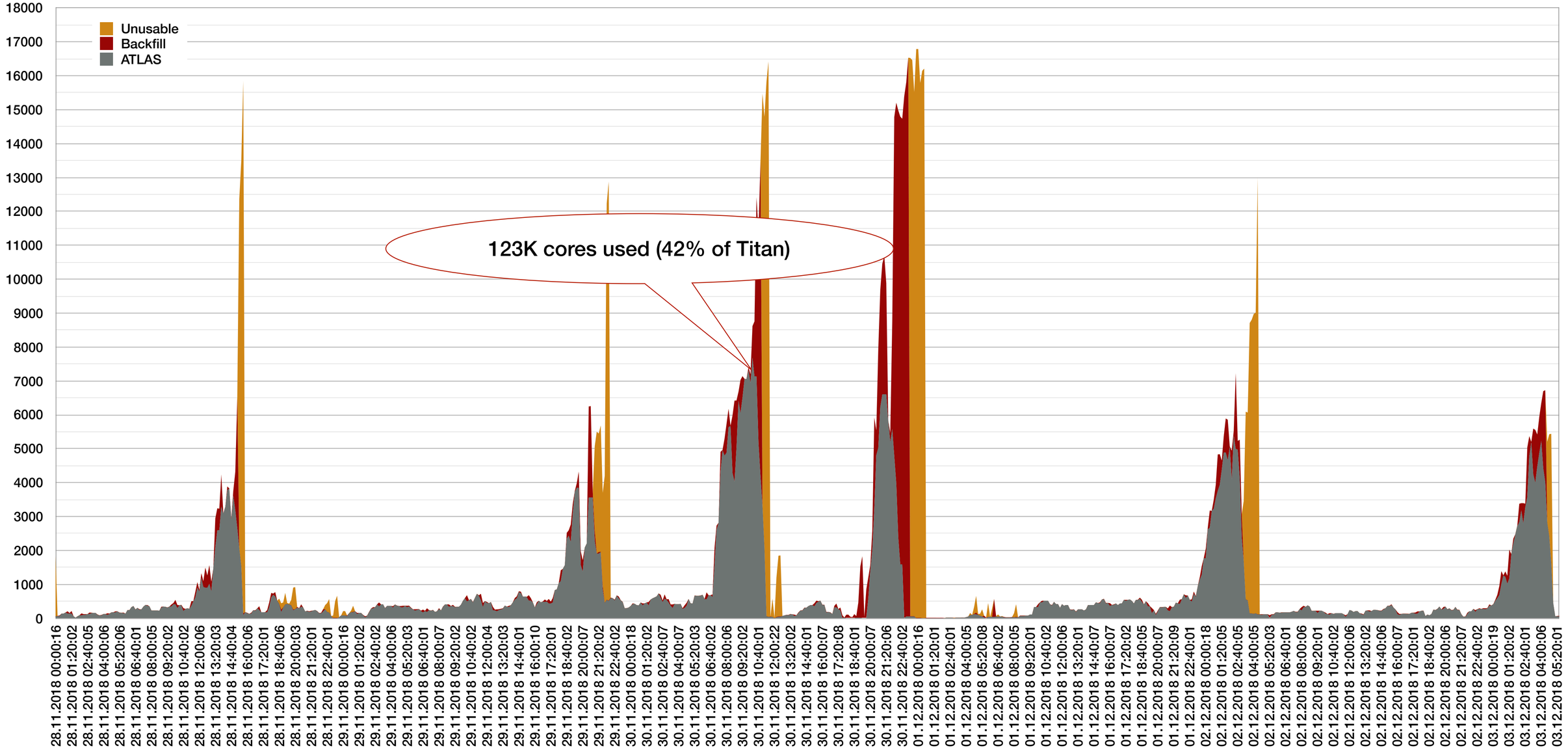
Harvester at OLCF

- Validation from beginning of 2018
  - In production for ALCC from March 2018
  - A set of improvements were implemented with increasing of capability and finally Harvester at OLCF manage workers with size of LCF class tasks (3800/Nodes, 60800 CPU per submission) with a few ‘workers' in the row
  - In September and October 2018 Harvester was extended with “backfill” capabilities: dynamic shaping of size of workload in configurable limits; managing of workers which 'stuck' in the local queue; proper processing of ‘closed' jobs etc.
    - Migration to the production was done step by step in November
- All ATLAS workloads at OLCF now managed by Harvester

Resources usage at OLCF with Harvester



Backfill consumption with Harvester



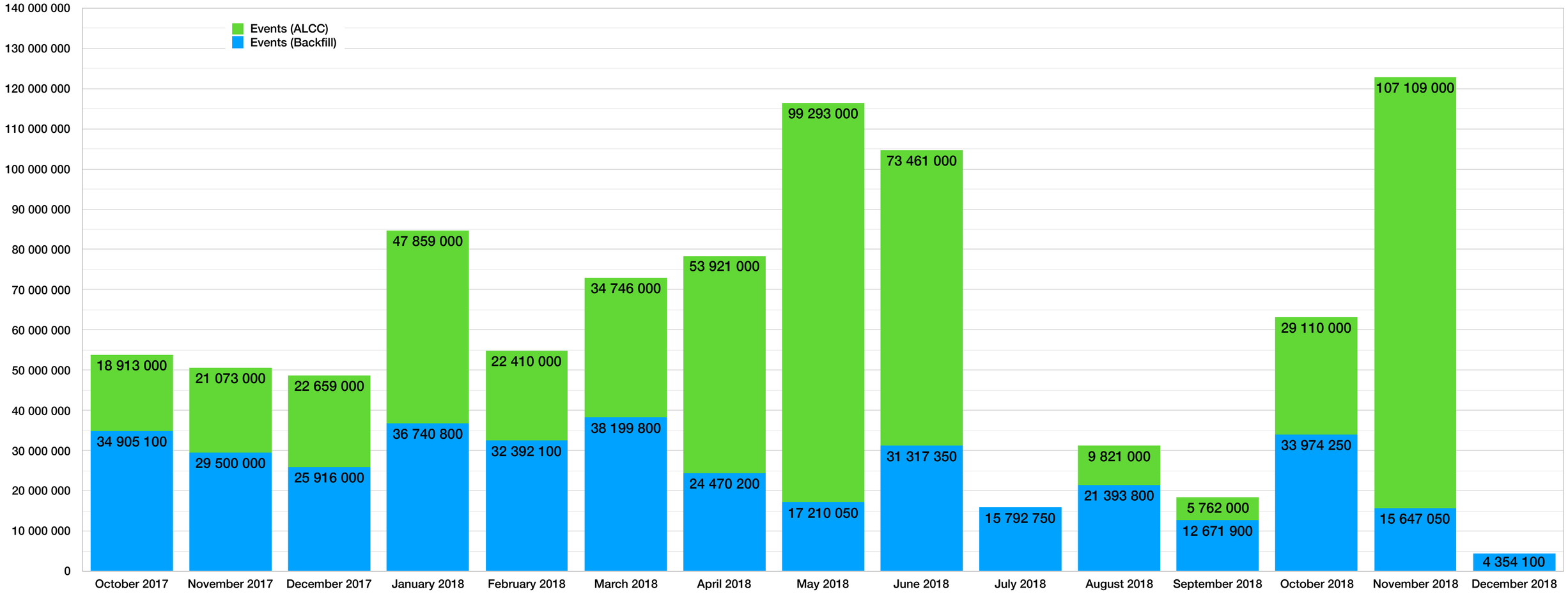
Harvester from operational point of view

- A lot of configurable parameters allows flexible management of instance
  - The well-documented procedure of deployment
  - A simplified procedure of management of service (start/stop)
  - Basic parameters do not require expert knowledge of the particular implementation
  - Most important parameters can be configured through AGIS (does not require access to the facility) or PanDA server
- Current monitoring tools are enough to get an understanding of the health of the system:
  - Harvester monitoring in BigPanDA: excellent progress
  - New accounting dashboards: better than few month ago, but still some more improvements will be good to have
- More improvements on the way: for example streaming of Harvester logs to Logstash

ALCC at OLCF

- OLCF have friendly handling of ALCC allocation: project can run other allocation with priority reduction, "period of clemency” at the end of ALCC year (May-June)
- 2017-2018 ALCC Award 80M Titan core*hours (1 Titan core*hour = 30 CPU hours)
  - **124M** Titan core*hours consumed, before significant reduction of priority
- 2018-2019 ALCC Award 80M Titan core*hours:
  - 70,5 Titan core*hours already consumed (88% of allocation).
  - 47,7 Titan core*hours consumed in November (60% of allocation)
    - Paused for the moment by request from Doug

OLCF Deliverables to ATLAS



- 920M Events (from October 2017)
  - ALCC: 546M
  - “Backfill”: 374M

Pilot 2. HPC Workflow

- HPC workflow in Pilot2 is the special workflow to manage execution of PanDA job on computing node under Harvester control
  - Act like simple MPI application
  - Used in “Many-to-one” mode of Harvester
  - Work for any generic PanDA job
- HPC workflow much simpler than traditional. Literally, only a few simple steps are needed:
  - Collecting the job definition from a pre-placed JSON file
  - Preparing for execution: pre-placement of input data in high-speed transient storage associated with the computing node (if needed)
  - Environment setup, incl. platform specific preparations
  - Payload execution
  - Publishing of job report in a JSON file, which includes state of payload and other metrics
  - Processing of job report
  - Transfer of output data to the shared file system
  - Archiving and storing payload logs to the shared file system
  - Declaring files for later stage-out
- Due to some of the steps requires platform-specific treatments, implementation of this steps are placed in dedicated plug-ins

Testing of Pilot2 HPC workflow

- Before testing of HPC workflow on HPC, the resource-specific plugin should be implemented
  - Not a huge development: there are 6 -7 required simple methods
  - For Titan plugin, it’s less than ~300 line of code
  - The code should follow Pilot2 code convention

Pilot2 deployment and configuration for HPC

- Code repository:
  - https://github.com/PanDAWMS/pilot2/tree/next (stable)
  - https://github.com/PanDAWMS/pilot2/tree/hpc_workflow (testing)
- Deployment:
  - git clone --single-branch -b next https://github.com/PanDAWMS/pilot2.git
- Configuration:
  - pilot2/pilot/util/default.cfg
  - section [Information] - set path to local cache of queue config file
  - section [Harvester] - should be correlated with harvester configuration
  - section [HPC] - path to local high-speed storage
- How to launch:

```
pilot.py  -q ORNL_Titan_MCORE -r ORNL_Titan_MCORE -s OLCF --pilot-user atlas
-w generic_hpc -d --hpc-resource titan
```

  - - w parameter, to specify workflow
  - - d parameter, to specify resource plugin name

Nearest plans

- Provide more information for monitoring of Harvester instances:
  - CPU/Memory consumption
  - stream of logs to Logstash at CERN
- Validate and put in production containers support in Pilot2 HPC workflow