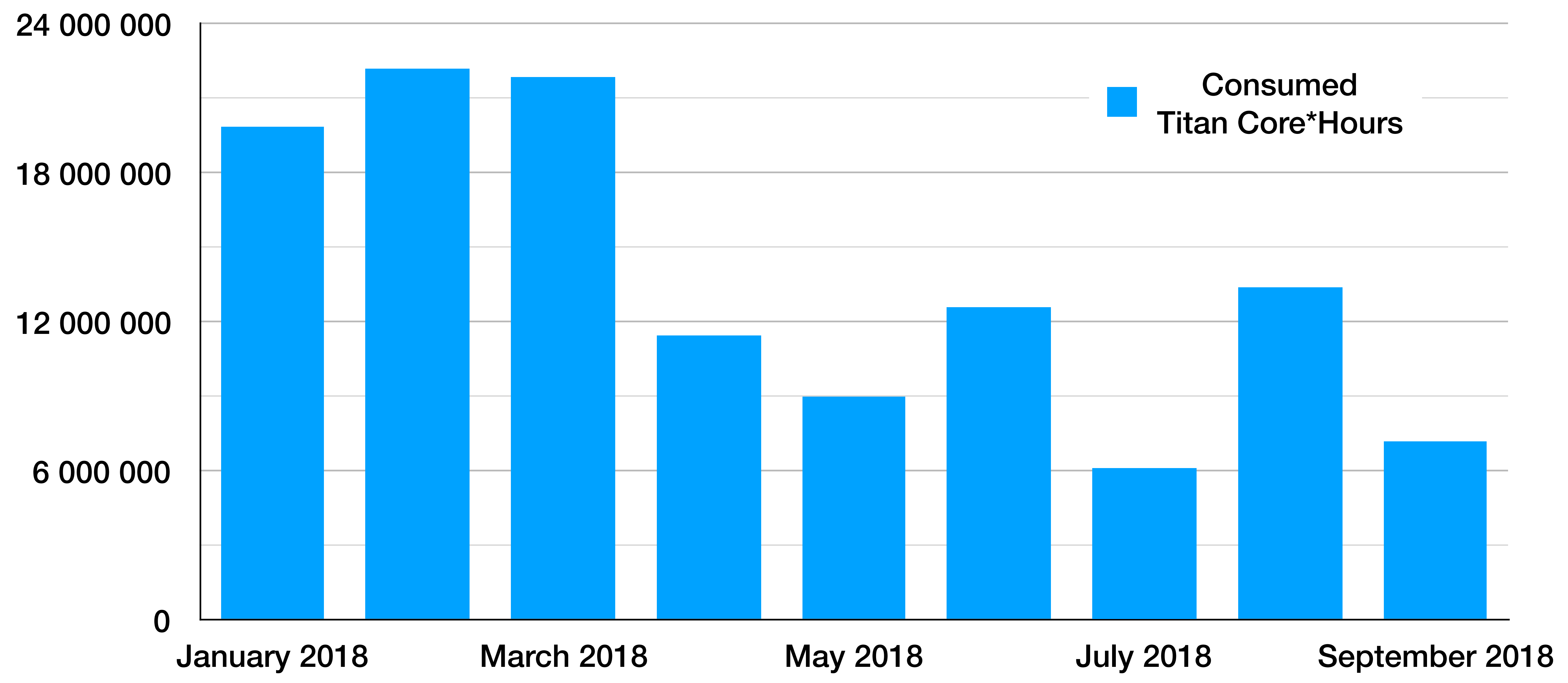
ATLAS PanDA WMS @ OLCF. Status report

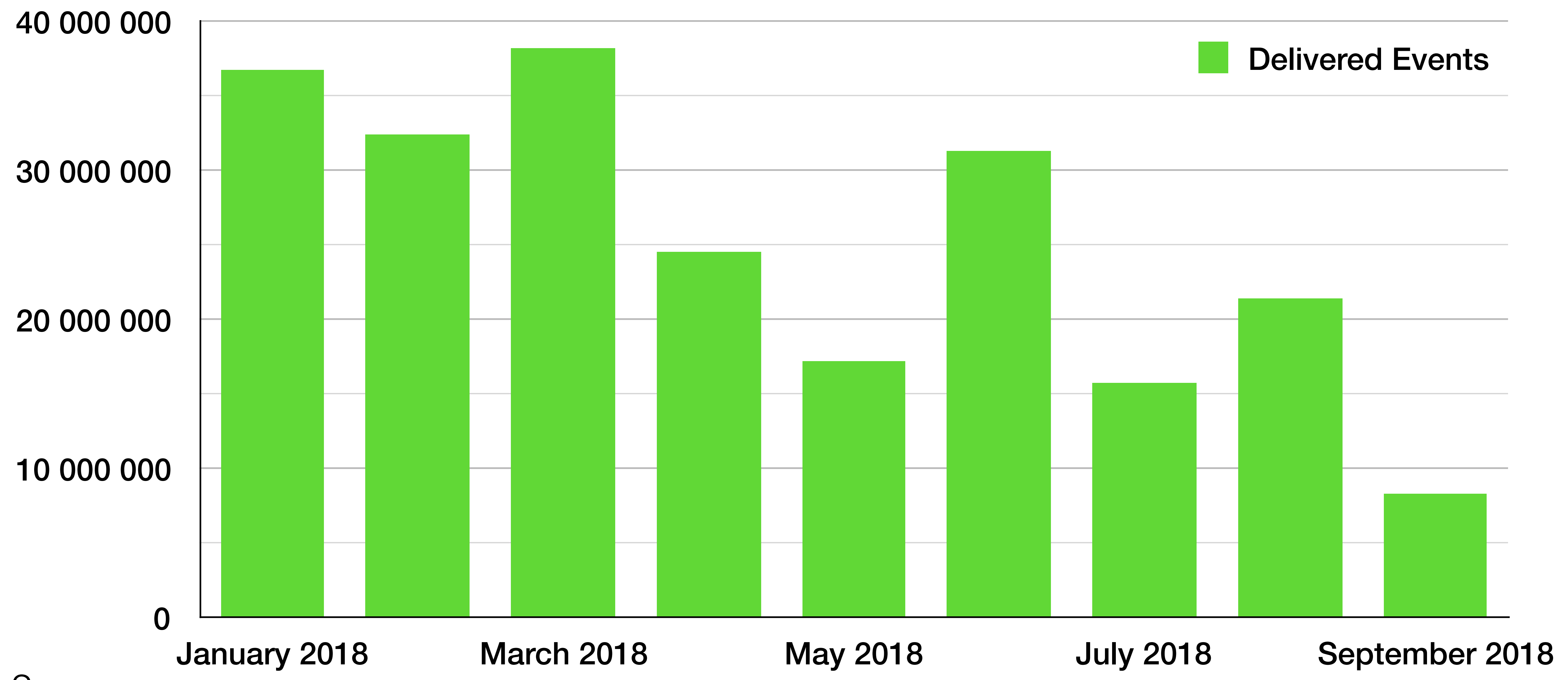
Danila Oleynik

BigPanDA TIM 23-24 Sep. 2018

ATLAS with backfill resources at OLCF

- Stable operation from month to month.
  - 123M Titan core*hours consumed in 2018
  - 225M Events delivered for ATLAS
- Configuration of current setup didn't changed overtime, and looks like more or less optimal parameter for production solution was chosen
  - Unfortunately, it's not possible to increase efficiency with outdated technology





AES at OLCF: validation of Yoda

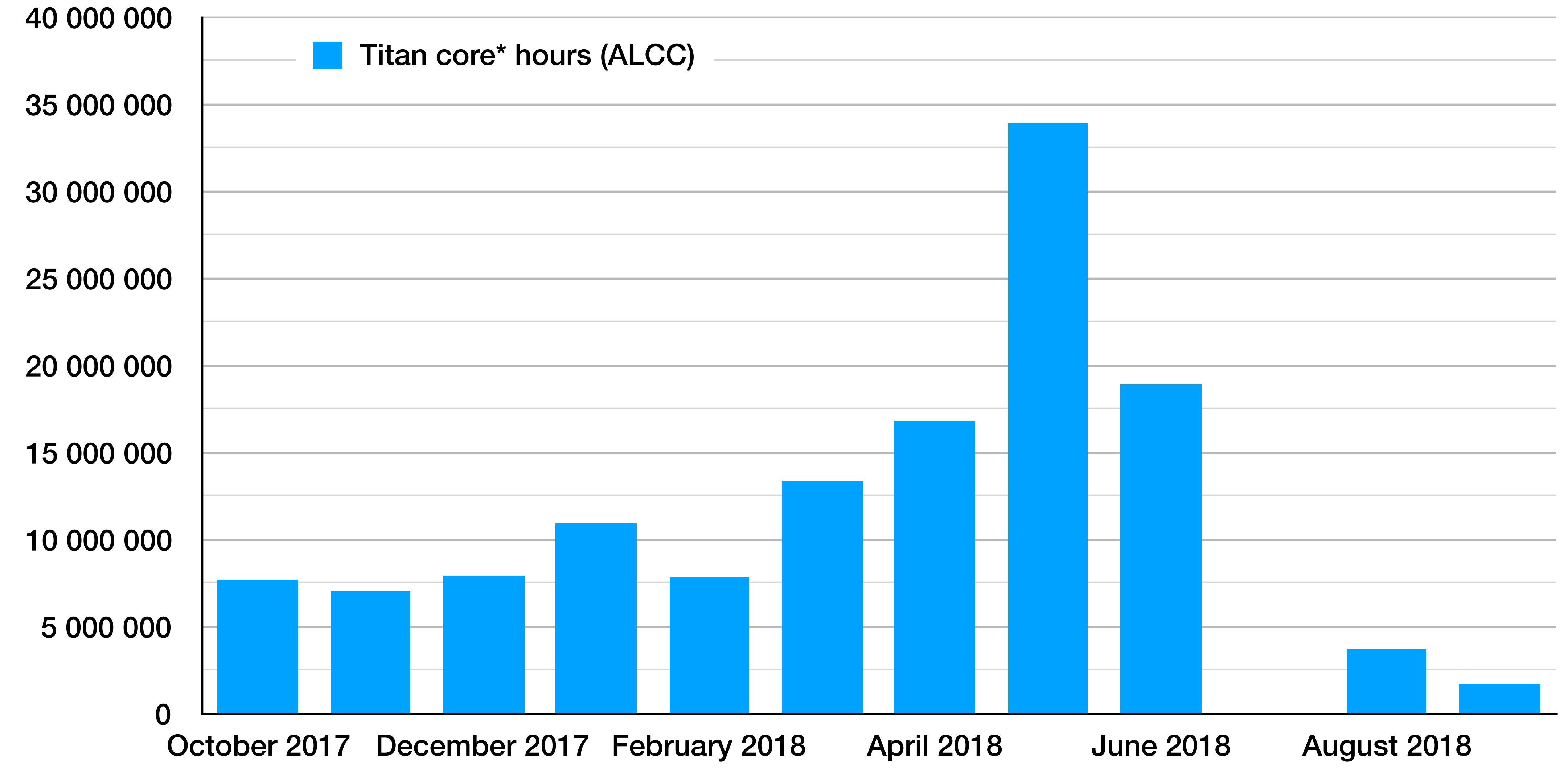
- Validation of Yoda (ATLAS Event Service solution) was started this summer
    - Yoda is an MPI-enabled job payload manager for the ATLAS experiment which plugs into the PanDA Event Service model for running work on HPCs under the supervision of the Harvester service (https://github.com/PanDAWMS/panda-yoda/wiki)
    - *Yoda can be launched as standalone application in case input parameters and data will be pre-placed in advance*
- Set of glitches and bugs were fixed by Yoda developers following of results of initial tests
    - most important issue which was solved: unbalancing of loading of ranks
- Yoda was not tested on scale, due to some concerns about IO

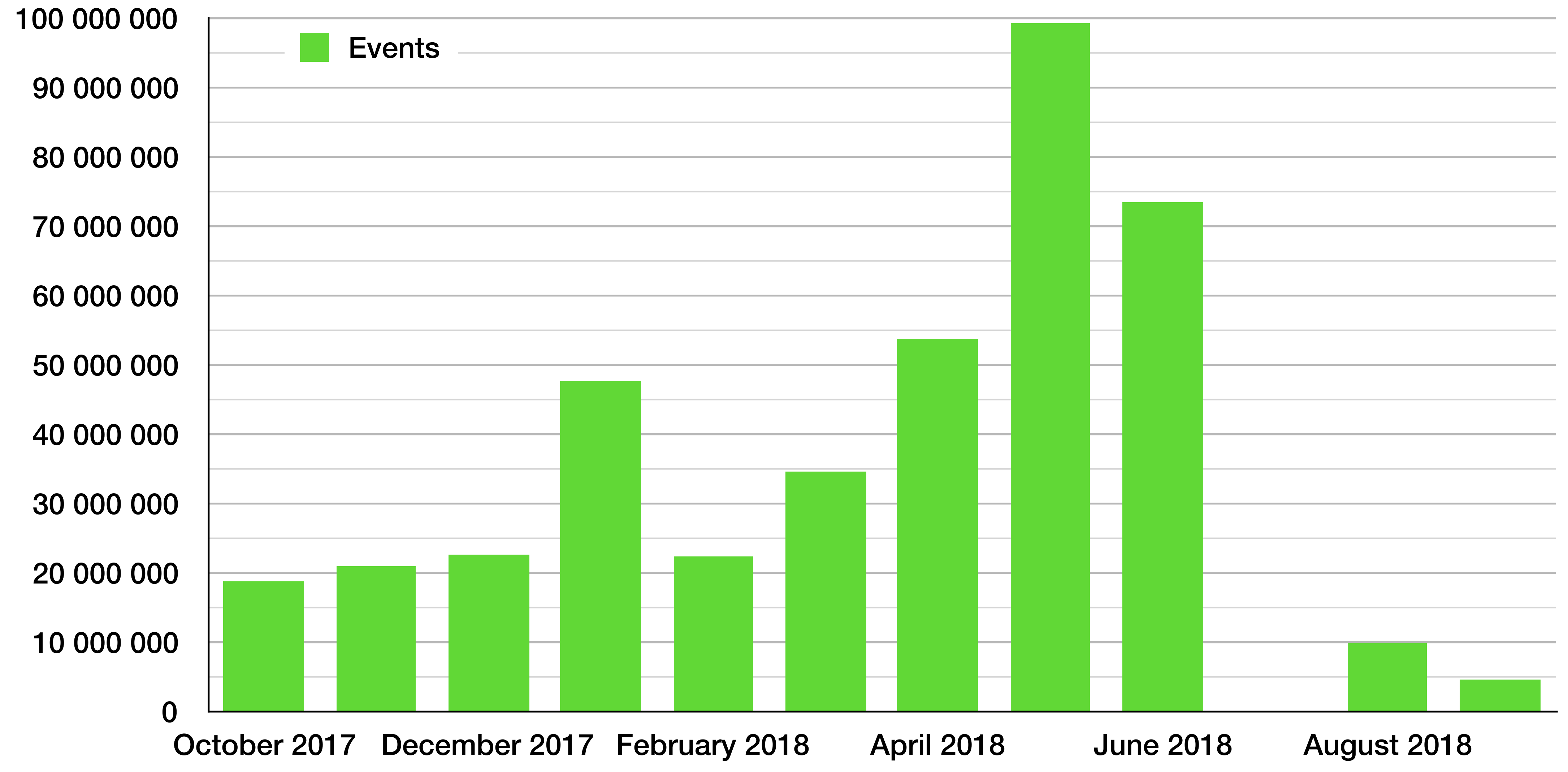
Required improvements for Yoda

- Yoda + ATHENA MP payloads are quite IO intensive for example: with a test on 10 nodes during 2 hours 4791 files (transient, logs, and outputs) were produced. 3589 of them were output files so, this number should increase linearly with the number of ranks
  - Output files pretty small - less than 1Mb
- To have a reasonable number of stage-outs, output files should be (periodically) packed before stage-out
  - Harvester have this feature, but it’s only will increase IO on Lustre (and loading to DTN)
- Also, a high number of transient files of Luster is not very good from IO point o view
- My suggestion will be to move working directories to RAM disk of Titan working node.
  - Droids should periodically pack output files and logs and move them to Lustre for future stage-out
- I hope we will have implementation plan for this from Yoda developers soon

ATLAS with ALCC Allocation

- Despite a late start, ALCC 2017-2018 was completed for more than 150%
  - 124M Titan core*hours (80M was the size of allocation)
  - 394M Events delivered for ATLAS
- ALCC 2018-2019 started in August
  - Not impressive in deliveries for the moment
  - But very stable in operation and easily tuned: current configuration up to 2 submissions (workers) simultaneously 1200 nodes (jobs) each





Harvester at OLCF

- Scalability issues, which were illuminated last spring were fixed
  - Time metrics were collected for a major operation, and it gave the possibility to identify weak places and fix them
  - Size of workers was increased to 1000+ nodes per worker, and it allowed significant increase of outcome in May and June
- Dedicated 'HPC Workflow' were implemented in Pilot2
  - Set of plugins to cover specialty of Titan was developed for Pilot2 HPC workflow
  - The ongoing process of development of plugins for other HPCs
- Harvester + Pilot2 with HPC Workflow were successfully tested at OLCF and in process of moving into production
  - Backfill capability for Harvester in progress
- In progress validation of full chain: Harvester + Yoda + Jumbo jobs
  - Again a little concerns about IO and optimisation of stage-in
  - Jumbo job triggers downloading of whole dataset for task (usually 1000 files)
- Differences between the launching of different types of workers: Pilot2 or Yoda through Harvester is only in a couple of lines of configuration

Monitoring and operation of Harvester at OLCF

- BigPanDA monitoring was extended with functionality which allows getting info about Harvester instances, basic parameters and error and warning messages which appears
- Harvester is easy to manage as service: stop, start restart and configure.
  - Configuration quite wide, and some of the parameters requires an expert knowledge. But, these parameters are not required to be changed somehow frequently
- Harvester instance at OLCF associated with the OLCF user account, so it will not be easy to manage one instance by different users.
  - This can be solved by ATLAS operational policies/agreement
- Unfortunately, we have a lack of OLCF infrastructure monitoring
  - For example degradation of FS can be reported a few hours later than it appears and began to be observed
  - It's not easy to estimate even own impact to the system: used space in project working directory, number of files, average IO an CPU loading on service nodes (DTN)
  - Limits of system
- Despite issues with infrastructure monitoring we will continue the transition of daily operations to ATLAS operation team
  - Basic parameters which need to be monitored was already described, more detailed documentation/“how to” will be prepared

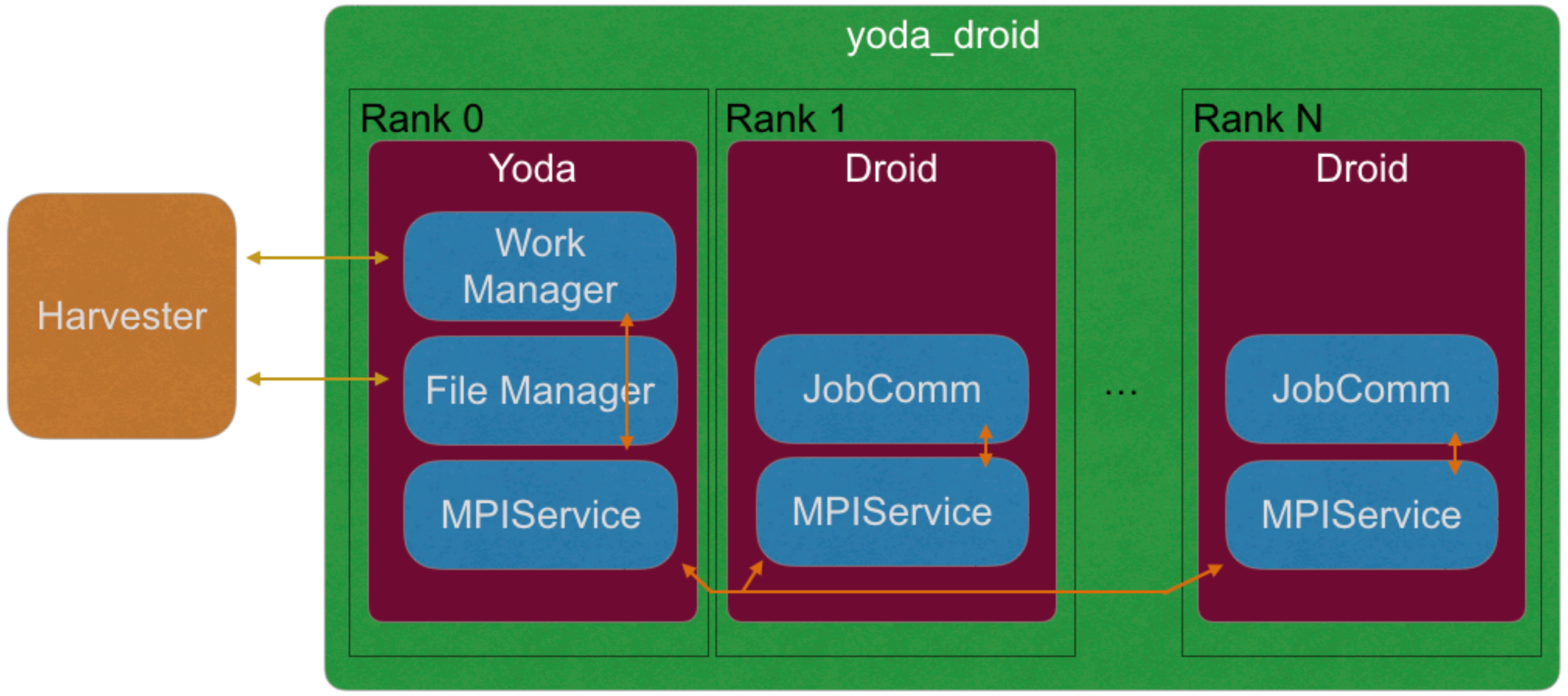
Summary

- Stable ATLAS production at OLCF in both modes: backfill and allocation
- Validation of Yoda: AES for HPC at OLCF forced set of improvements and bug fixes
  - Still some work to be done before going to production scale
- Harvester was improved to cope scalability issue and now thousands of PanDA jobs can be managed
  - More improvements on the way: backfill compatibility, Pilot2 and Yoda final integration
- Nearest plan: get rid from MultoJob pilot in next 4-6 weeks
- Continue transition of day by day operation efforts to ATLAS operation team

Backup

Yoda Overview

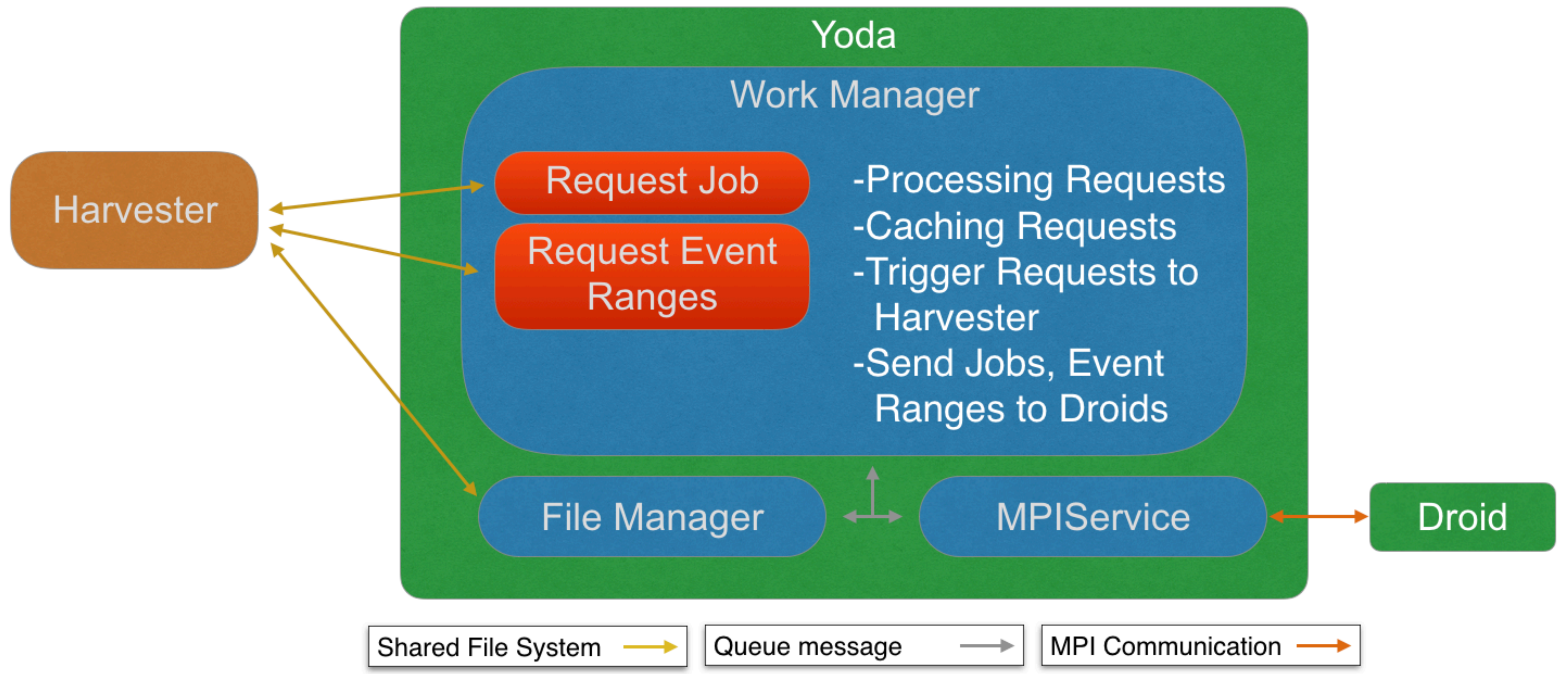
- Yoda is an MPI-enabled job payload manager for the ATLAS experiment which plugs into the PanDA Event Service model for running work on HPCs under the supervision of the Harvester service.



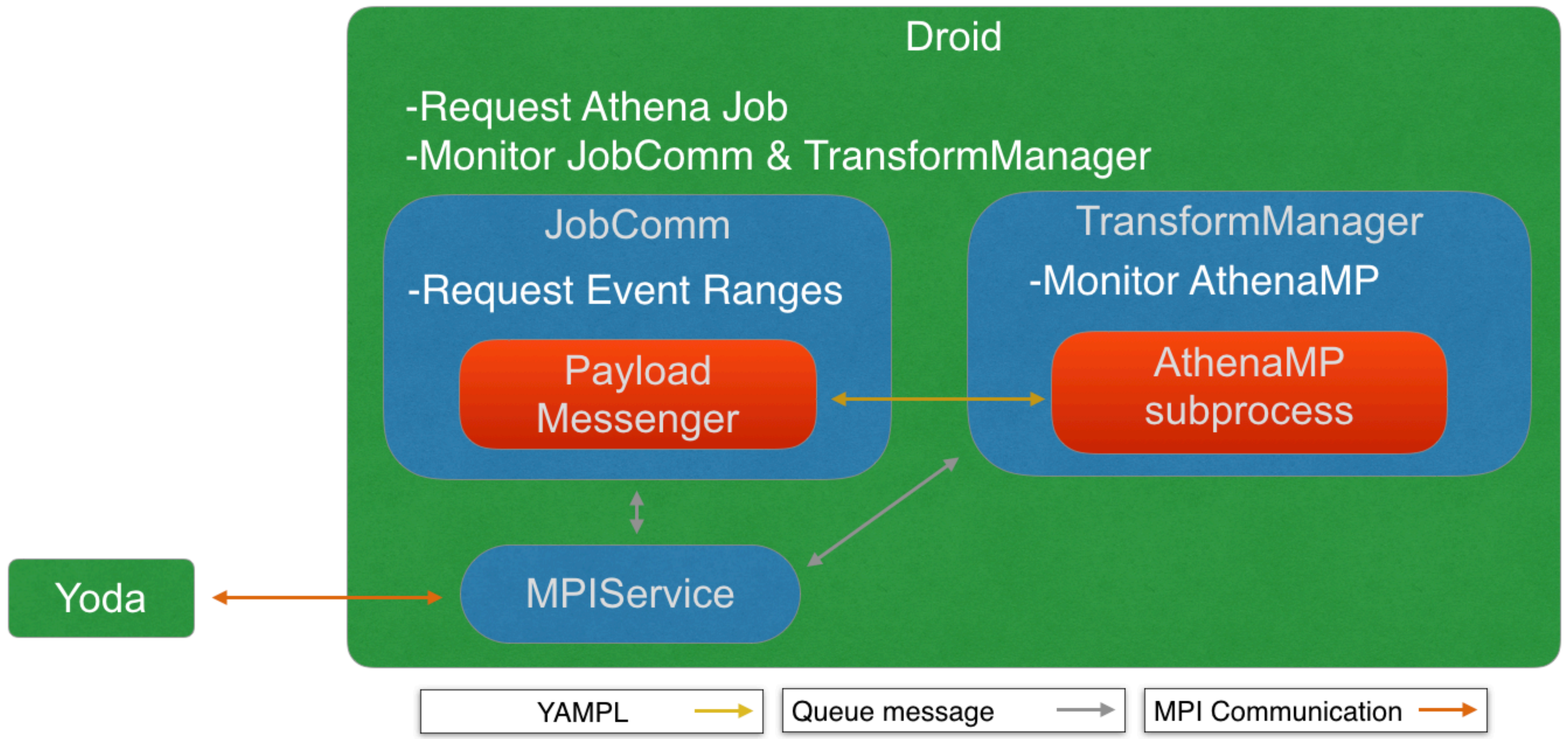
- Each Droid is responsible for starting an ATLAS Athena application as described in jobs provided by the primary yoda thread running on Rank 0. Yoda communicates with Harvester to receive work descriptions and then passes that information to droids upon request. Yoda & Droid use MPI to communicate. Droid communicates with AthenaMP using Yampl.

Yoda Process Description

- The Yoda process is responsible for communicating with Harvester and responding to requests from Droid processes. Yoda keeps track of work needing to be done. When Droid sends a request for work, Yoda provides it or responds that no work remains. This model ensures that Yoda does not need to keep track of all the droid ranks and what work they are each doing. This reduces the memory overhead and logical complexity.



Droid Process Description



- The Droid process runs on each node of the HPC and launches the actual Athena payload. Droid initiates a Job request to Yoda when it first begins. The job description will be provided to both the TransformManager and the JobComm.