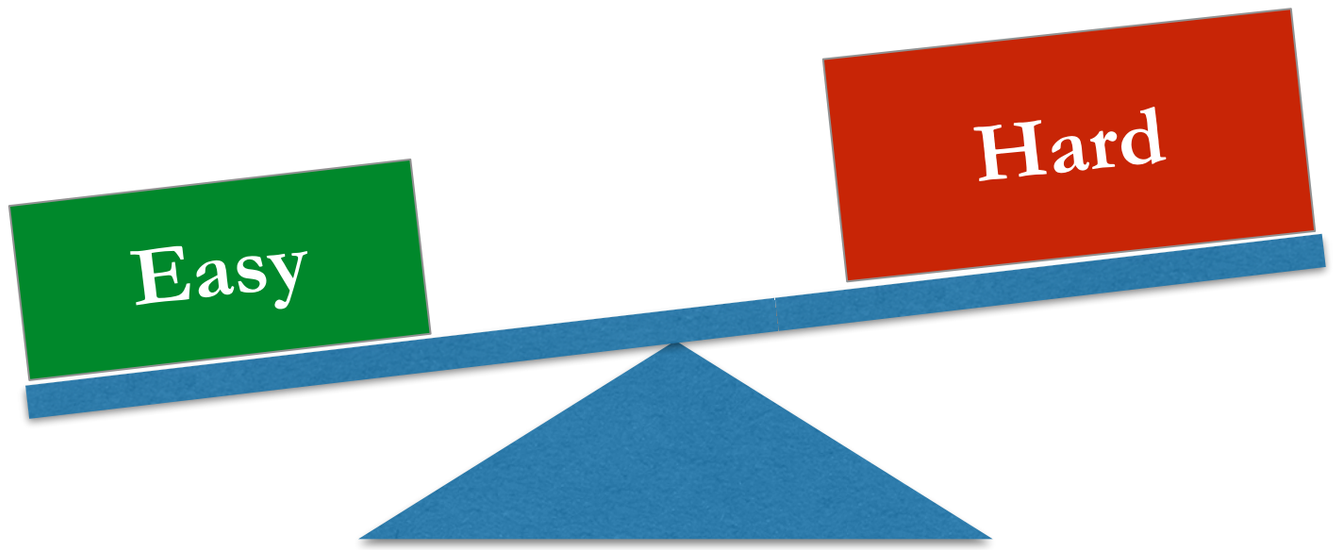
High Performance Computers Working for ATLAS

Taylor Childers (Argonne)
with Tom LeCompte (Argonne),
Tom Uram (Argonne), and
Doug Benjamin(Duke)

Putting HPCs to Work for ATLAS

- HPCs are not grid machines
  - Custom Hardware, Custom Compilers, Custom Environments, Custom OS
- This makes running Athena out of the box a daunting task
  - cannot use standard libraries and binaries included in CVMFS
  - recompilation necessary, as well as external libraries
- Running Event Generation can be done in a standalone way
  - Allows production to move forward while we work out the more difficult problem

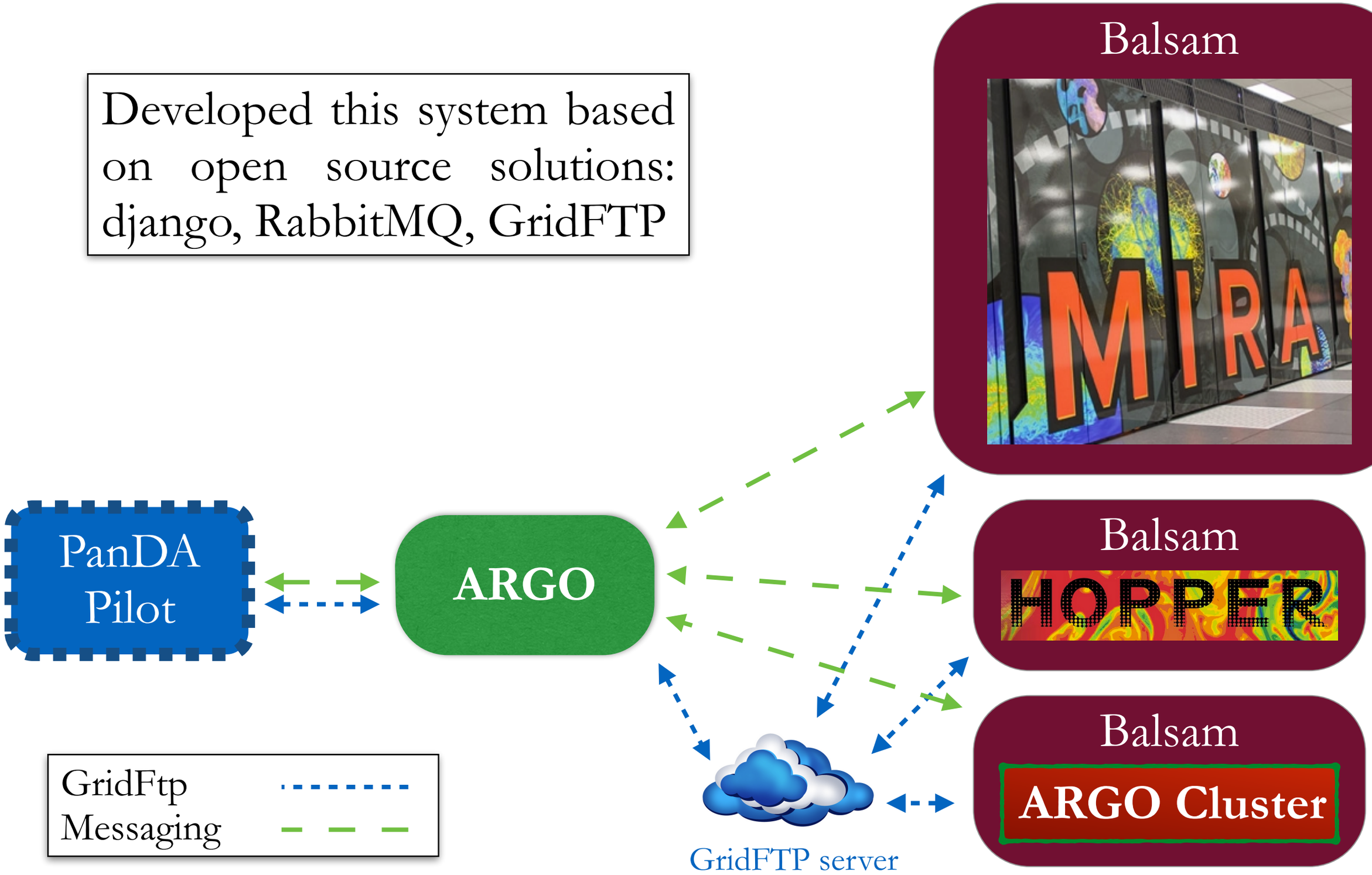


Putting HPCs to Work for ATLAS

- Grid jobs run serial processes on one core of a multi-core processor.
- We could just submit pilot jobs to HPC queues, but this ignores the benefits of a highly parallel HPC.

ARGO-Balsam HPC workflow system

Developed this system based on open source solutions: django, RabbitMQ, GridFTP



ARGO-Balsam HPC workflow system

ARGO is a python-based workflow manager implemented in the django framework.

ARGO-Balsam HPC workflow system

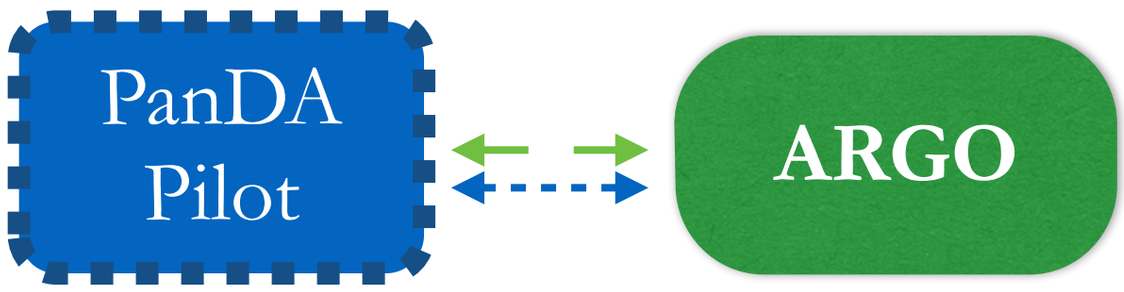
Balsam is a batch scheduler interface that can be run on a variety of systems. For instance:

- Mira (HPC@Argonne)
- Hopper (HPC@NERSC)
- ARGO Cluster (local cluster)

ARGO-Balsam HPC workflow system

An ARGO specific PanDA Pilot parses the job, sends it to ARGO, and waits for status updates via the RabbitMQ message queue system. Any data transfer is handled by GridFTP.

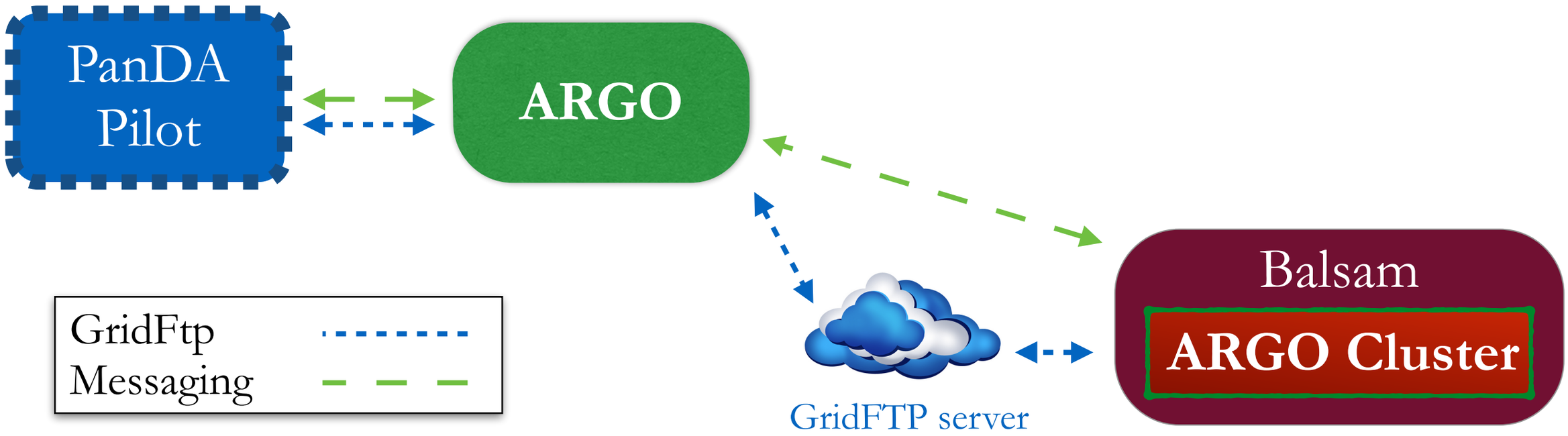
Let's use Alpgen as an example.



GridFtp
Messaging

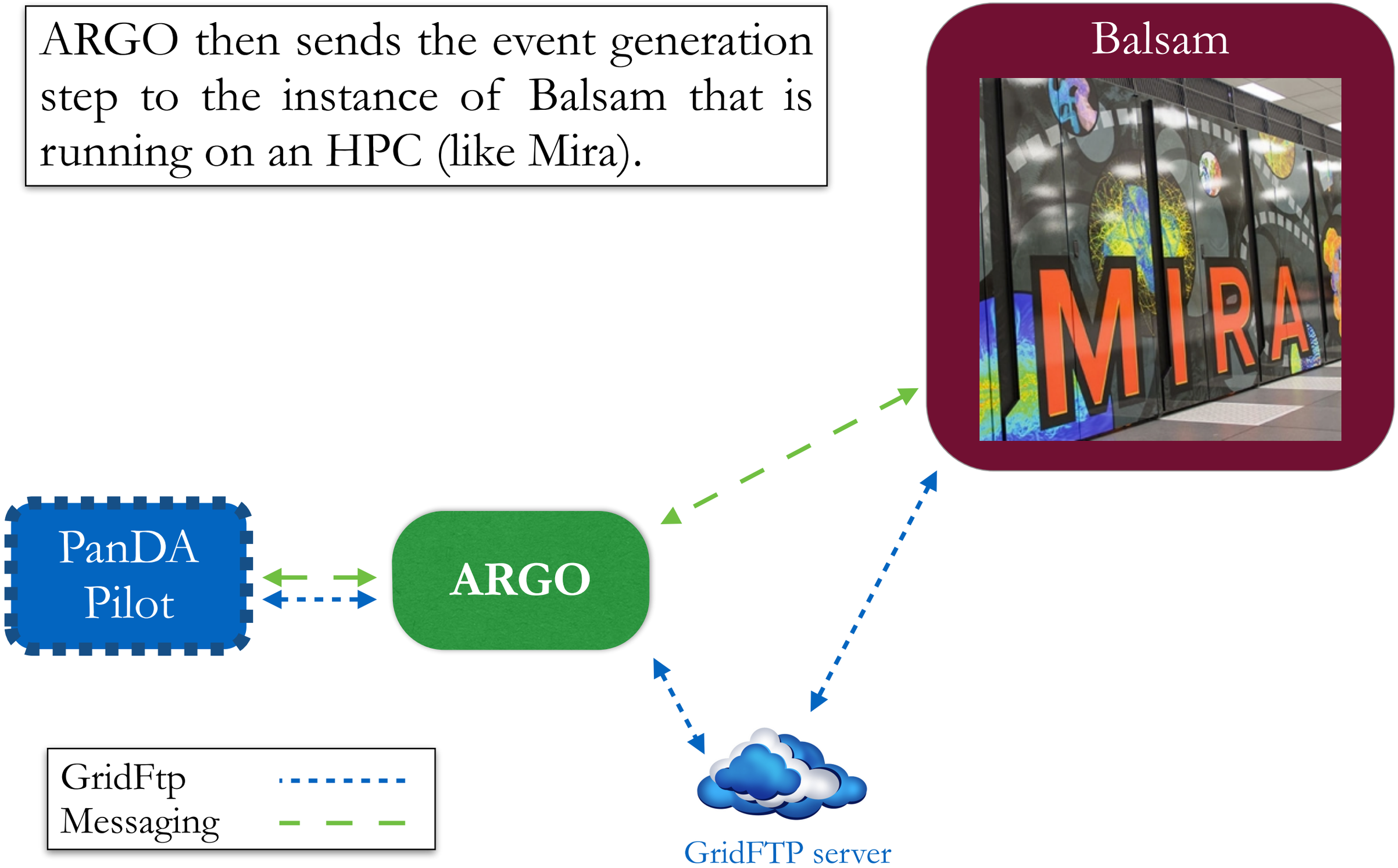
ARGO-Balsam HPC workflow system

- Alpgen has a serial integration step at the start. We run that on the local ARGO Cluster.
- This cluster is useful for steps that are not optimal for an HPC.
- It is SLC6 (grid-like) so we can also run Athena here when needed.
- ARGO sends the instance of Balsam running on the ARGO Cluster the integration job, then retrieves the output when it is completed.



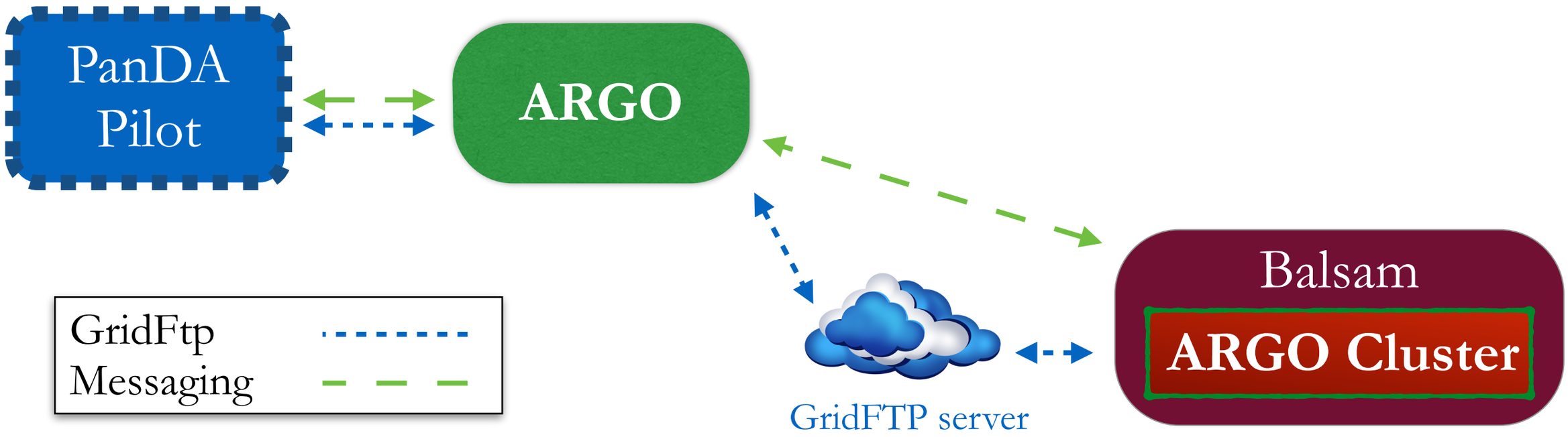
ARGO-Balsam HPC workflow system

ARGO then sends the event generation step to the instance of Balsam that is running on an HPC (like Mira).



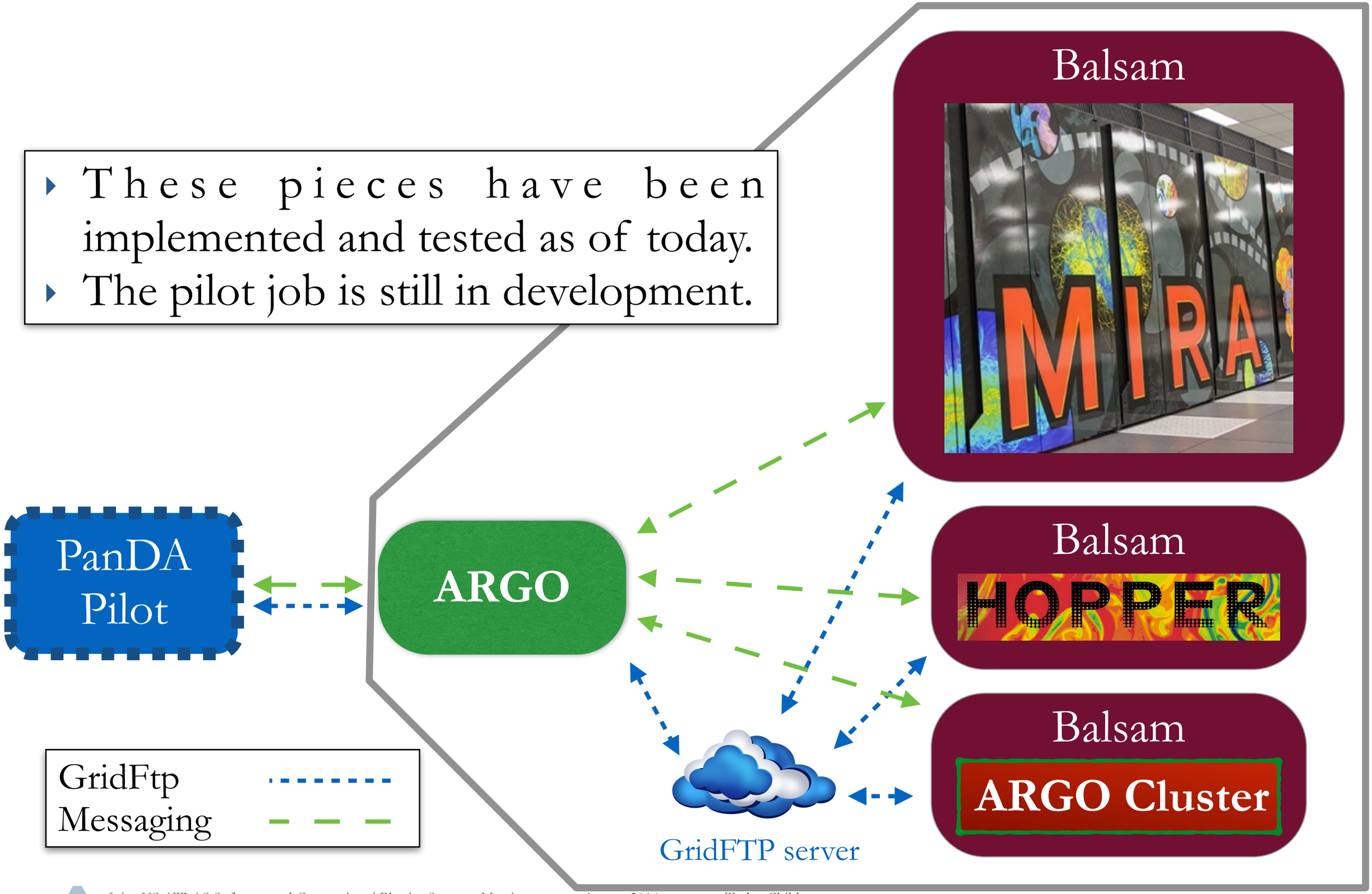
ARGO-Balsam HPC workflow system

- The ARGO Cluster performs any post processing requiring Athena, which in the case of Alpgen is running the showering.
- Since this is run in Athena, using the original job options from the job received from the pilot, the output is identical to a grid job.
- The resulting data is retrieved by ARGO and sent to the Pilot for registration on the Grid.



ARGO-Balsam HPC workflow system

- These pieces have been implemented and tested as of today.
- The pilot job is still in development.

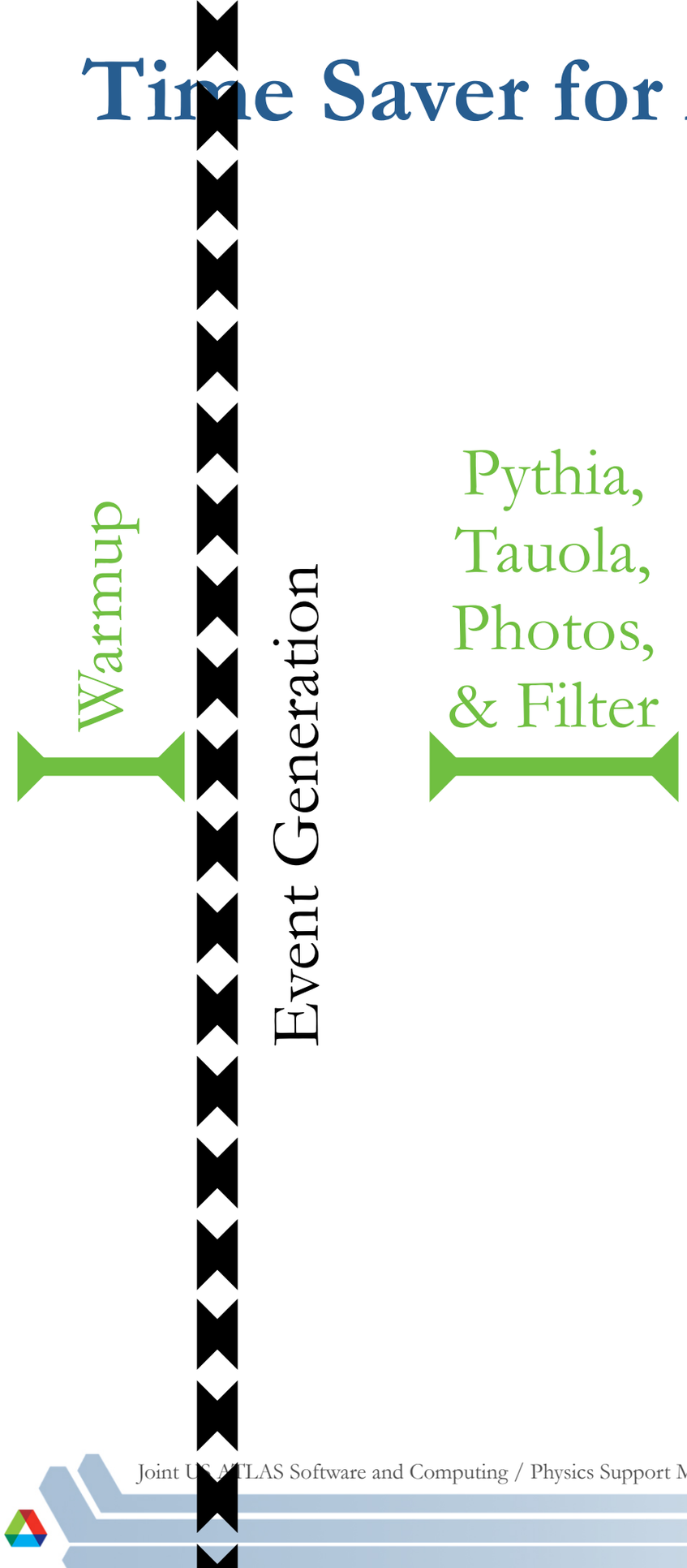


Time Saver for Alpgen

- On the grid, each job runs an identical warmup
- Warmup for multi-leg processes can be many hours long and grid jobs are limited to 24 hours in length
- This means a lot of duplicated work on the grid

Event Generation

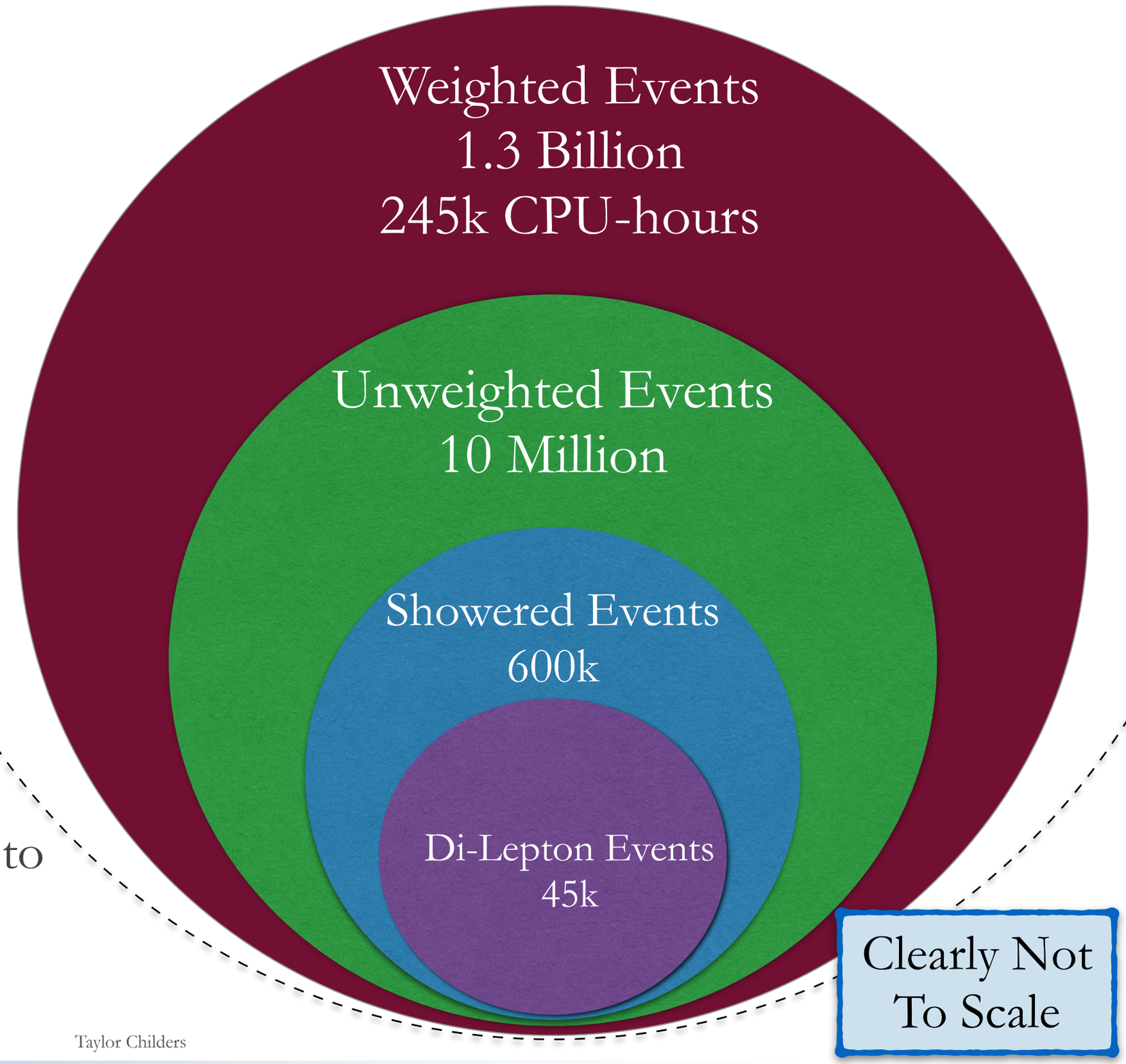
Time Saver for Alpgen



- In the ARGO-Balsam framework, we save time.
- Integration step is done once.
- This is an important part of properly using an HPC since time is competitively allocated.
- Could also imagine saving time by showering with different simulations at the same location.

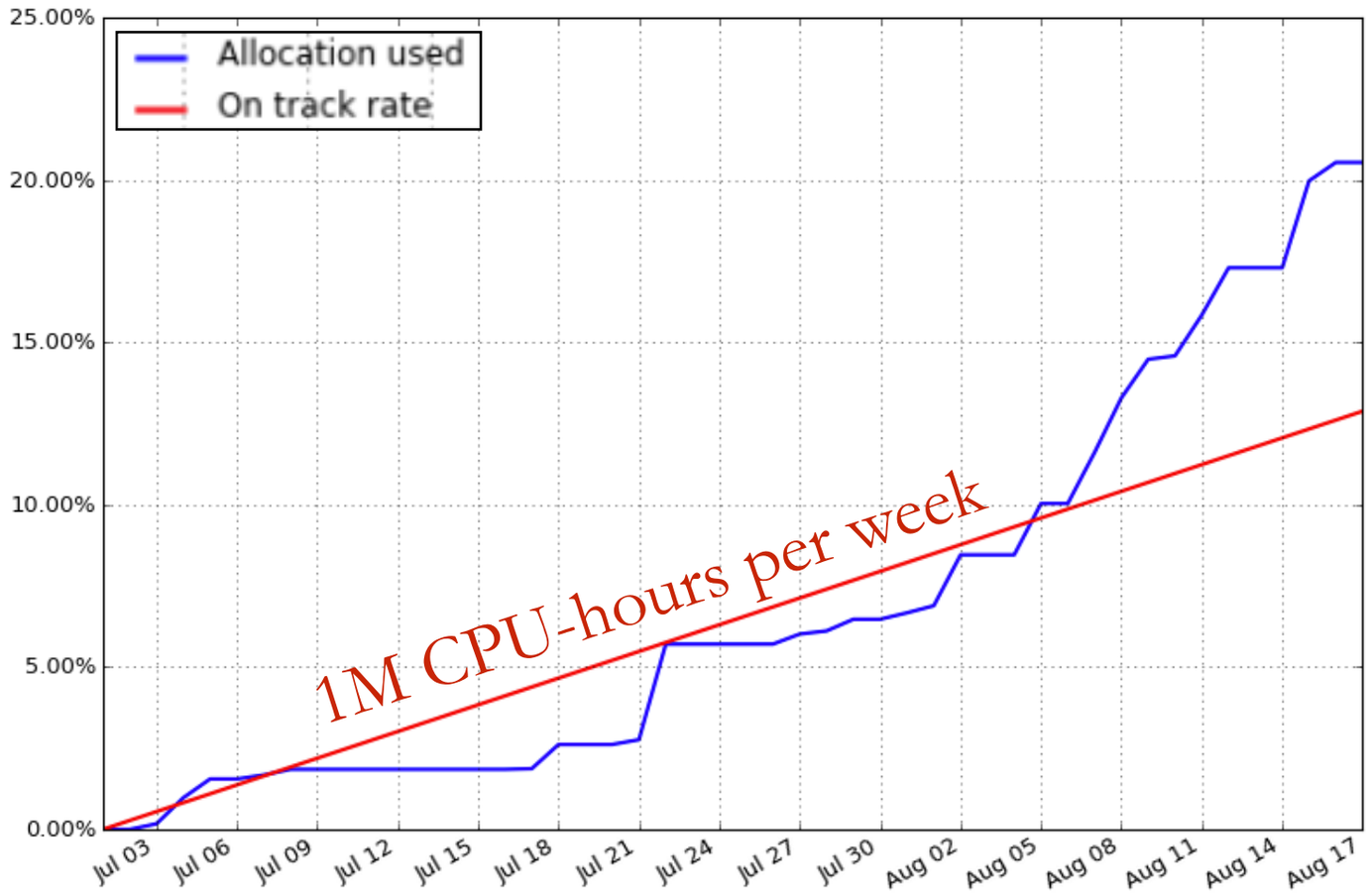
Example Job: Z+5jets

- **45,000 $Z \rightarrow \tau\tau$+5jet** Alpgen+Pythia events
- Would have required **12,250 24hr Grid jobs**
- **Saved 50k CPU-hours** compared to the Grid due to work duplication in grid workflow



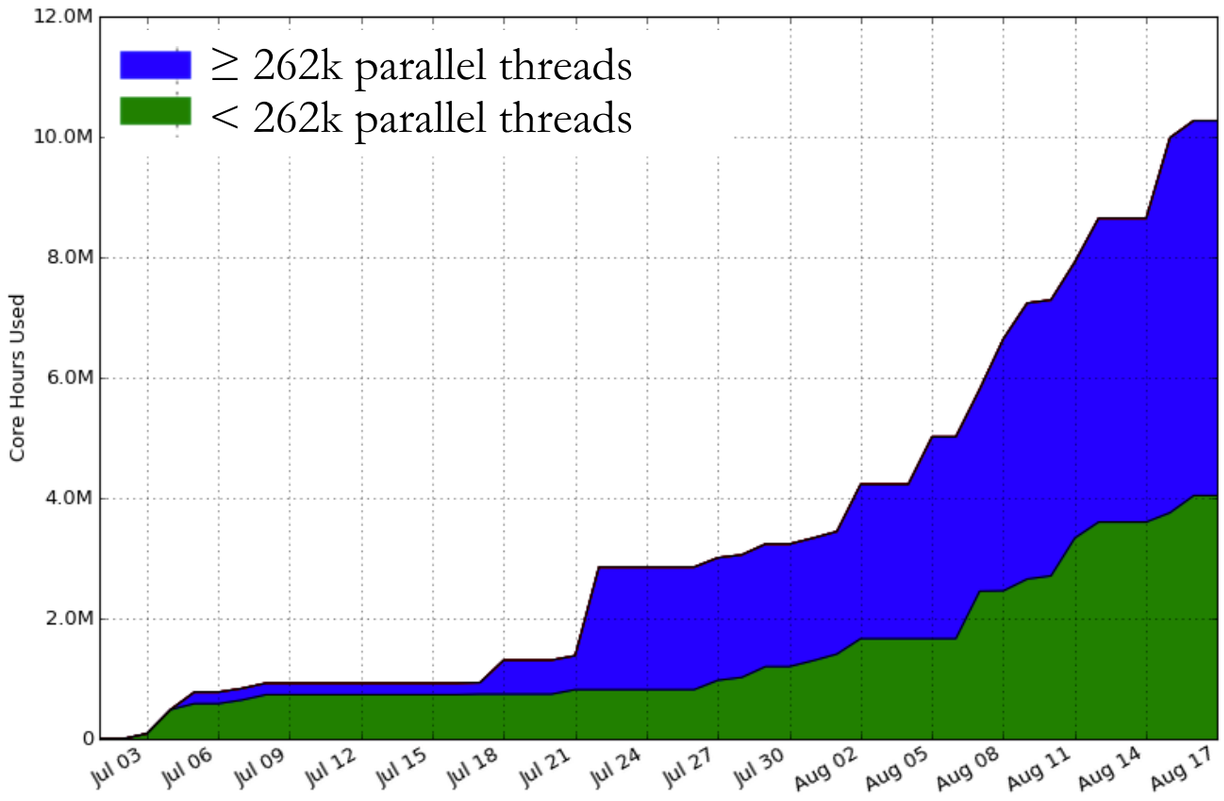
ASCR Awarded ATLAS with an allocation

- ASCR has awarded us with an ALCC (ASCR Leadership Computing Challenge) award:
  - 50 million CPU-hours at Argonne LCF
  - 2 million CPU-hours at NERSC
  - Started July 1st
  - Simultaneously recognizes the success we've have, while encouraging more success
  - ASCR wants HEP on HPCs
- Have done quite well at using it
- It's important for us to use this completely or it could jeopardize future applications.



Learning about growing job sizes

- One of the goals of this project is to **learn about how to grow jobs** to massive parallel scales.
- If we can't **FILL** current HPCs, we won't be able to use the next generation machines at the EXA-SCALE expected in 2021 or so.
- Currently have run Alpgen with up to **262,144 MPI-aware threads**
- Generated so far:
  - **105k Alpgen+Pythia Z(tautau)+4j @ 8TeV**
  - **40k Alpgen + Pythia Z(tautau)+5j @ 8TeV**
  - **6.5M Alpgen + Pythia Z+4j @ 13TeV**
  - **5.1M Alpgen + Pythia Z+5j @ 13TeV**
  - **860k Alpgen + Pythia Z+6j @ 13TeV**
  - **500k Alpgen W(heavy flav.)+4jet @ 13TeV**
  - **1.2M Alpgen W+5jets @ 13TeV**
- **200k each of Sherpa W+jets, Z+jets, multijets, and Drell-Yan for the Sherpa 2.1 validation effort**



Discussion Points

- We've developed a platform-independent framework based on open source software that can submit jobs to multiple HPCs.
- This platform will soon be integrated with the PanDA Pilot job.
  - Still need to define what signals PanDA would like us to send concerning the job status
    - Job status on HPC? Queued, Running, Post-processing…
  - Still need to define how PanDA knows what can be run using ARGO because this will not be a standard grid site
    - Can only run Alpgen, Pythia, Sherpa at first… Geant4 sometime later.
  - With running Athena on HPCs as a goal, it would make sense to use a **common deployment method** for getting Athena to these sites and compiled. CVMFS?
  - How to fill these machines? **Yoda/Jedi? Opportunistic Backfill & Big Job Allocations**
- When the ALCC applications are due in Jan/Feb 2015, we should know **what kind of proposal we want to submit**.
  - In 2014, 42 ALCC awards totaling 3B CPU-hours (~70M CPU-hours per award)
- ALCCs are a gateway drug to **INCITE awards which are O(100M+)** CPU-hour proposals