Direct I/O vs copy to scratch, WAN vs LAN

Thomas Maier
LMU

Nicolò Magini
INFN Genova

2018-03-06

ATLAS Sites Jamboree, March 2018

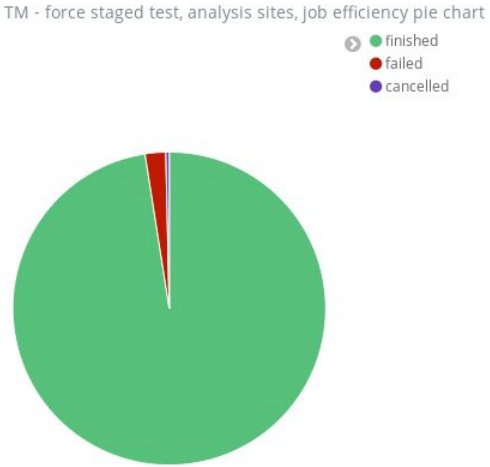
Introduction

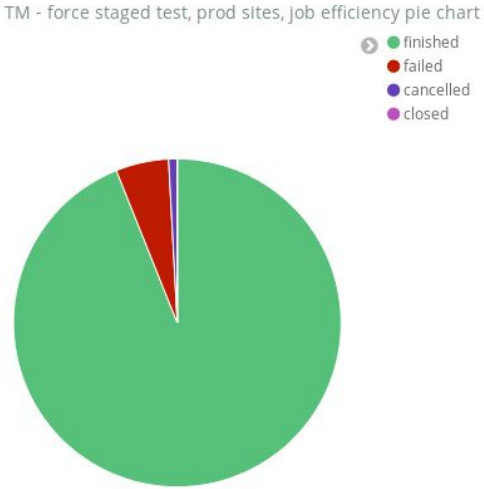
- Started to look into alternatives to traditional copy-to-scratch for more workflows
  - Direct I/O over LAN vs. copy-to-scratch
  - (WAN access vs. LAN access)
  - Streaming events over WAN → ESS
- Will present some test results in the following

Direct I/O vs. Copy-to-Scratch

- Test of direct I/O vs. force staged via HammerCloud test setups (provided by Tomas Javurek)
- Enforcing direct I/O or staged in by overwriting AGIS config:
  - overwriteQueuedata={direct access lan=True,direct access wan=True}
  - transferType = direct
- Dedicated HC templates, except for force staged on production sites
  - Used existing PFT
- Direct I/O in this case is reading from local SEs

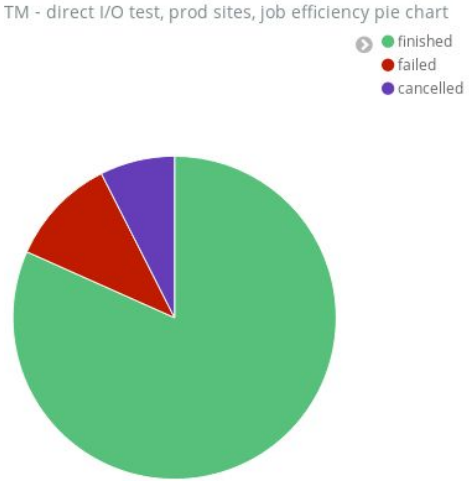
ANALY Sites
Force Staged





Prod Sites
Force Staged

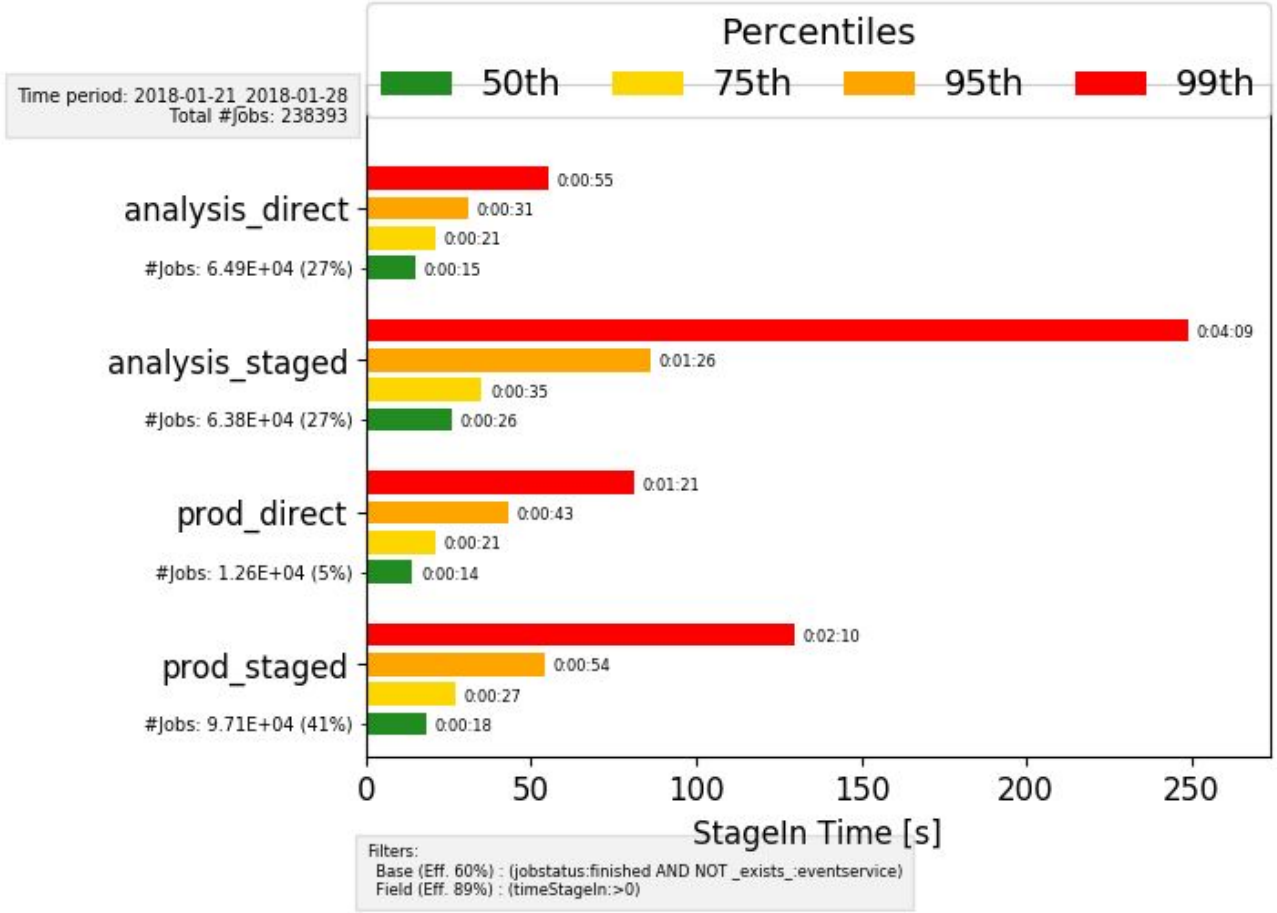
ANALY Sites
Direct I/O



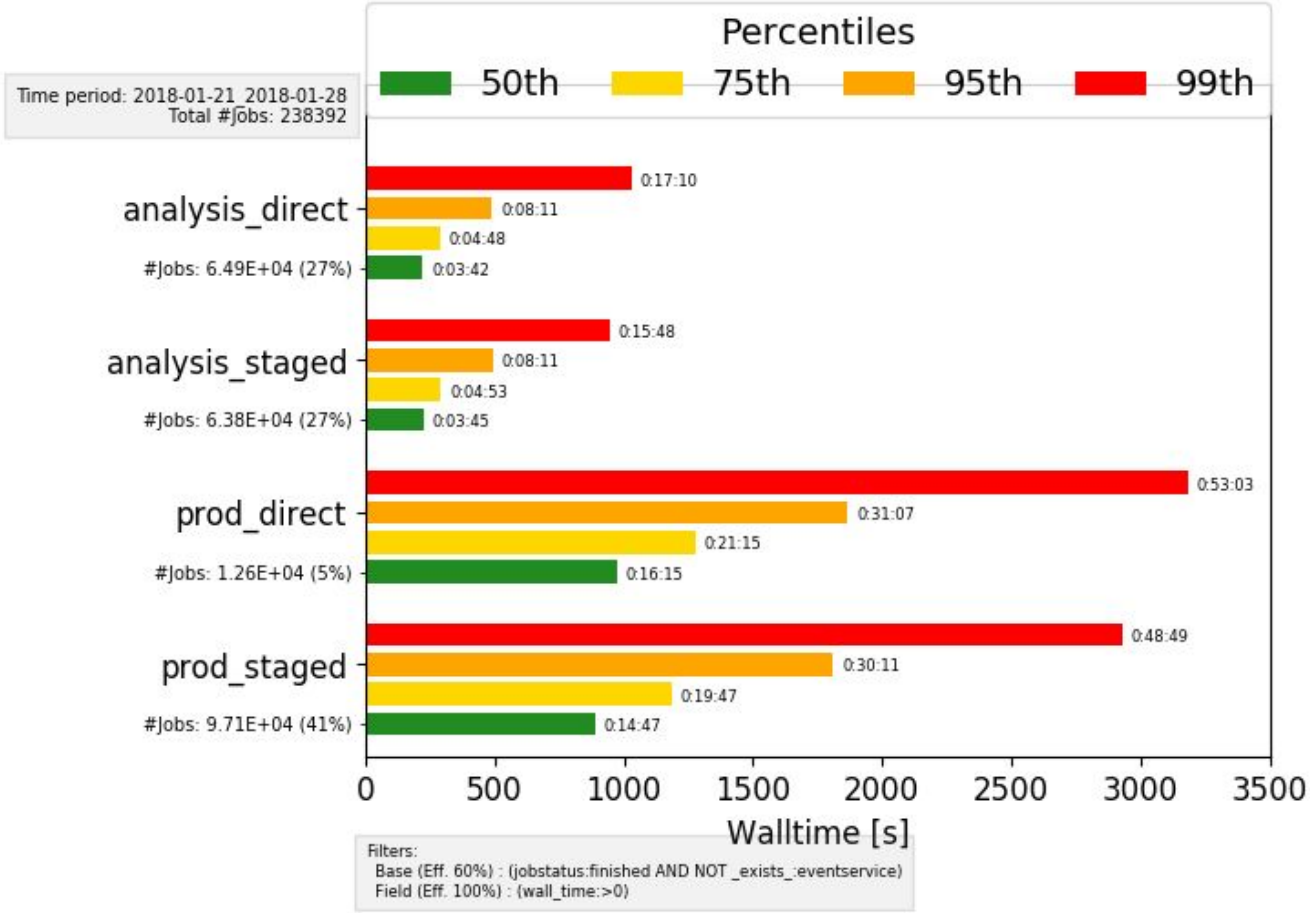
Prod Sites
Direct I/O

*1 week of HC jobs

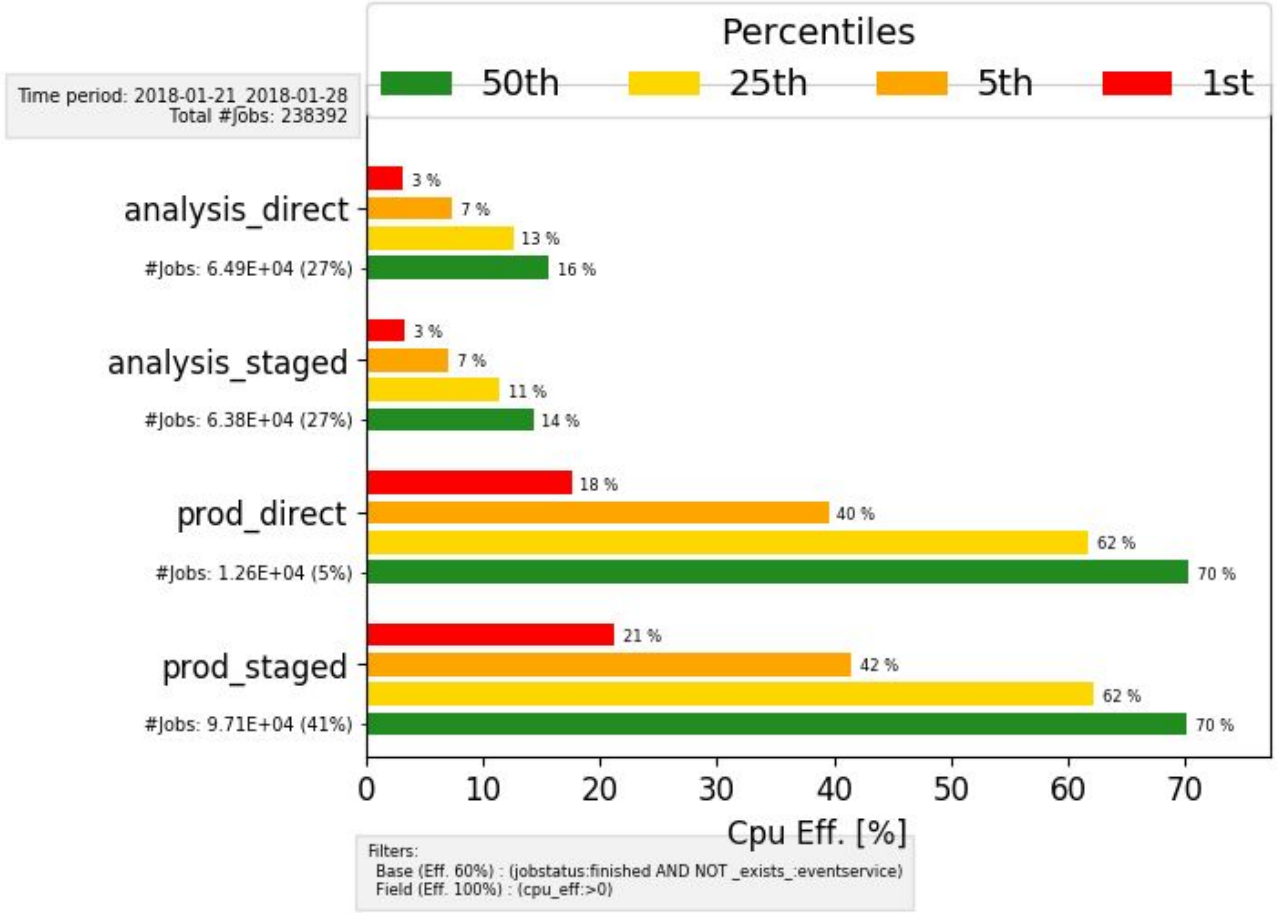
Comparison: StageIn Time



Comparison: Walltime



Comparison: CPU Eff.



Direct I/O Summary

- First direct I/O vs. copy-to-scratch comparisons with HammerCloud tests
  - ANALY_* sites show very similar job efficiency between the two
  - Prod sites significantly less efficient ⇒ underlying technical issue
  - Finished job performance very similar for direct I/O and copy-to-scratch
- Direct I/O jobs are reading from local SEs
  - Need test setups where file access is done over WAN incrementally further away from site
- Tests are copies of PFT and AFT
  - By design short, functional tests
  - For proper direct I/O performance measurement, require longer running jobs

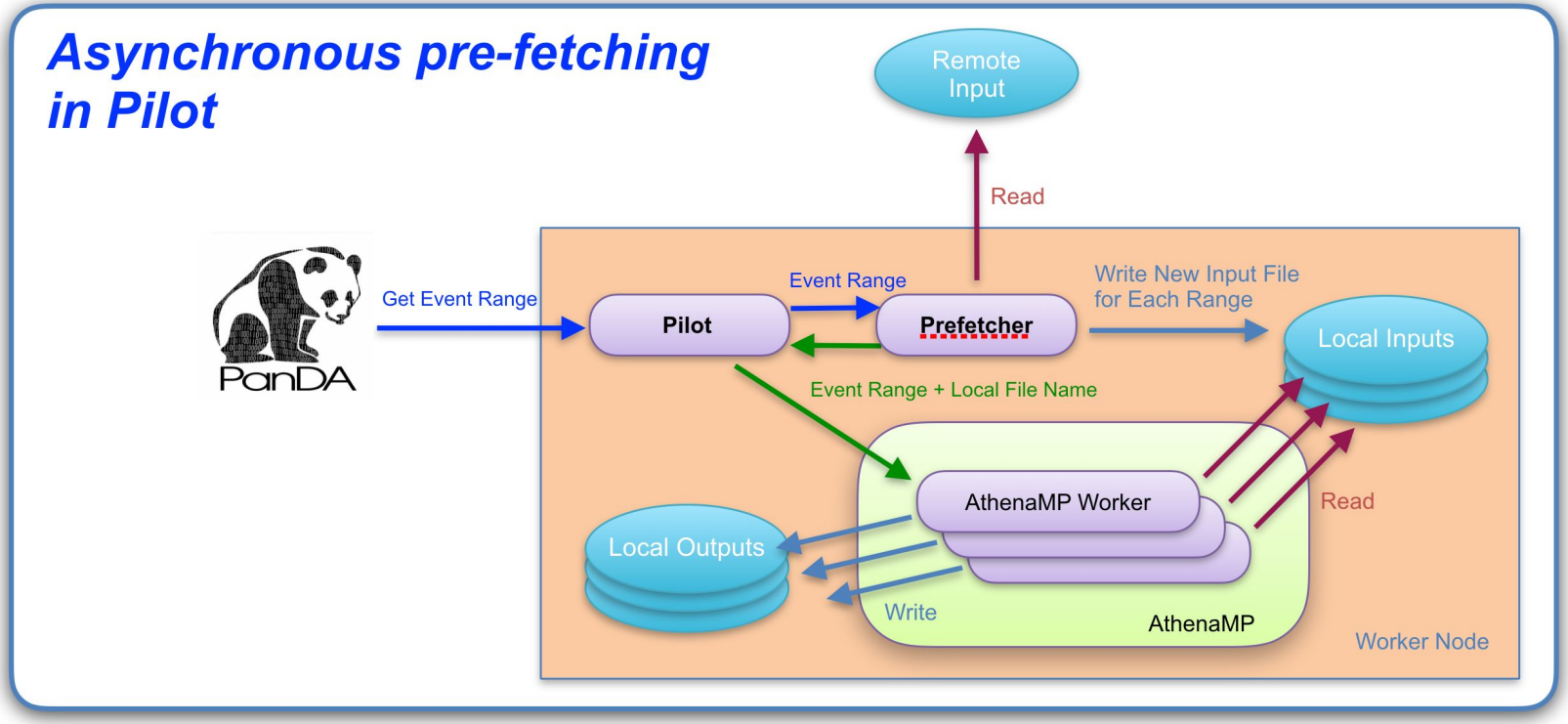
Event Streaming Service

- **Event Streaming Service (ESS)**: delivers input data to compute nodes over the network at fine granularity
  - Complements the fine granularity in work assignment and output delivery of the Event Service

- The goal is to benefit from the **good** of remote WAN access - *less input replicas, faster task start* - but protect applications from the **bad** - *inefficiency with high latency*

- For further details, see the talks by Torre and Vakho in the ESS session of the March S&C week

ESSv1 prototype



- Details in https://twiki.cern.ch/twiki/bin/view/PanDA/EventServiceDataPrefetching

Prefetcher commissioning

- ESSv1 implemented as a **Prefetcher** process started by the pilot in parallel to the main AthenaMP application
  - Prefetcher reads the remote input file over WAN, and duplicates each event range to a small input file on local scratch disk
- Able to run successfully over WAN since December
- Created 100k-event SIM tasks in PanDA for validation and ran them on AGLT2 reading input from BNL
- Compared some metrics for tasks running over WAN with Prefetcher vs without Prefetcher

Prefetcher task performance

- No significant difference observed in task running time (not expected for SIM)
  - see backup slides for details

- Observed ~10% of jobs failing with Prefetcher timeout
  - Occasionally stuck/slow opening files over WAN
    - FIXED: Prefetcher will not close&reopen input file after every event
    - FIXED: Pilot will restart the Prefetcher instead of aborting the job
  - Causes waste of events produced since the last stageout cycle (2-4 hours)
    - TODO: Pilot could stage them out before cleanup

ESS Summary

- Progressing towards a working ESS prototype
- Ready to start to use Prefetcher also for other workflows
  - Starting to submit test Derivation tasks
- Will also check performance at different WAN distances
  - Feel free to volunteer your site to get validation tasks

Backup Slides

ESSv1 implementation

- ESSv1 implemented as a **Prefetcher** process started by the pilot in parallel to the main AthenaMP application
  - Just another AthenaMP process which reads the remote input file, and duplicates each event range to a small input file on local scratch disk

- Relies on Event Service to split the input data into event ranges
  - Side effect: currently supports only G4 sim

- Actively working on integration of the Prefetcher prototype with PanDA since October

Prefetcher commissioning

Prefetcher over WAN				
Task ID	Done evts	Failed jobs	Completion time - h	Completion time (no queue) - h
12858157	100%	70	6:00	4:30
12868946	100%	56	8:30	7:45
12944727	100%	45	13:45	10:45
12959866	98%	196	9:00	8:30
13048011	100%	22	6:45	6:45

Prefetcher commissioning

Direct access over WAN				
Task ID	Done evts	Failed jobs	Completion time - h	Completion time (no queue) - h
12858152	100%	36	4:15	4:15
12868956	100%	8	8:00	8:00
12944729	51%	3091	16:06	7:00
13048014	100%	3	7:13	5:15
Default (Rucio transfer + copy-to-scratch)				
Task ID	Done evts	Failed jobs	Completion time - h	Completion time (no queue) - h
12935663	100%	5	7:00	5:00

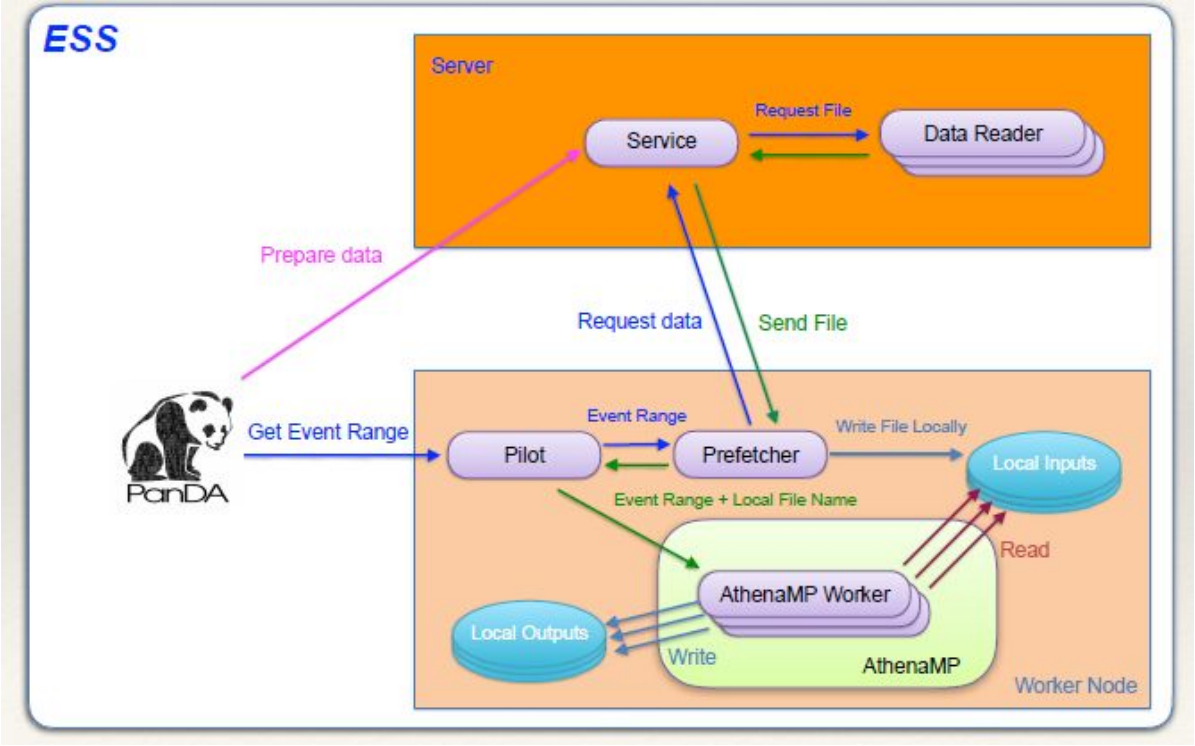
Configure a Prefetcher task in ProdSys

- **project_mode: usePrefetcher=yes**
- Requires Athena >= 21.0.21
  - E.g. AMI TAG s3209 (21.0.38), project_mode: cmtconfig=x86_64-slc6-gcc62-opt
- Enable EventService
  - project_mode: esFraction=1.0; registerEsFiles=yes; skipScout=yes; isSmallEvents=yes; corecount=0; ramCount=0
- Enable direct access over WAN/LAN
  - project_mode: allowInputWAN=yes; allowInputLAN=only
- Parameters needed for proper treatment of event ranges spanning more than one file or less than one file
  - project_mode: inFilePosEvtNum=yes; ignoreTrfParams=skipEvents

Prefetcher next steps

- Extend Prefetcher to different workflows that could benefit from ESS more than G4 Sim, e.g. derivation
  - Many inputs, potentially distributed, expensive in (re)brokering
  - But not validated with ES yet
  - Started to send validation tasks, looking into results
- Optimize Prefetcher to prefetch further
  - Currently it prefetches only one event at a time, and only just in time
  - Discussion started in https://its.cern.ch/jira/browse/ATLASPANDA-410
- Look into caching Prefetched events
  - Successfully prefetched files through XCache hacking the Pilot
- Look into decoupling Prefetcher from ES

ESSv2



- Process preparing the reduced input files is moved into a Server close to the data
- Prefetcher process in pilot becomes a client asking for event files from the Server
- Design/implementation/ prototype to start based on results with ESSv1