# Harvester for Grid/Cloud

Tadashi Maeno (BNL)
on behalf of Harvester team

ATLAS TIM,
18-22 September 2017, CERN, Switzerland

# Current Status for Grid and Cloud

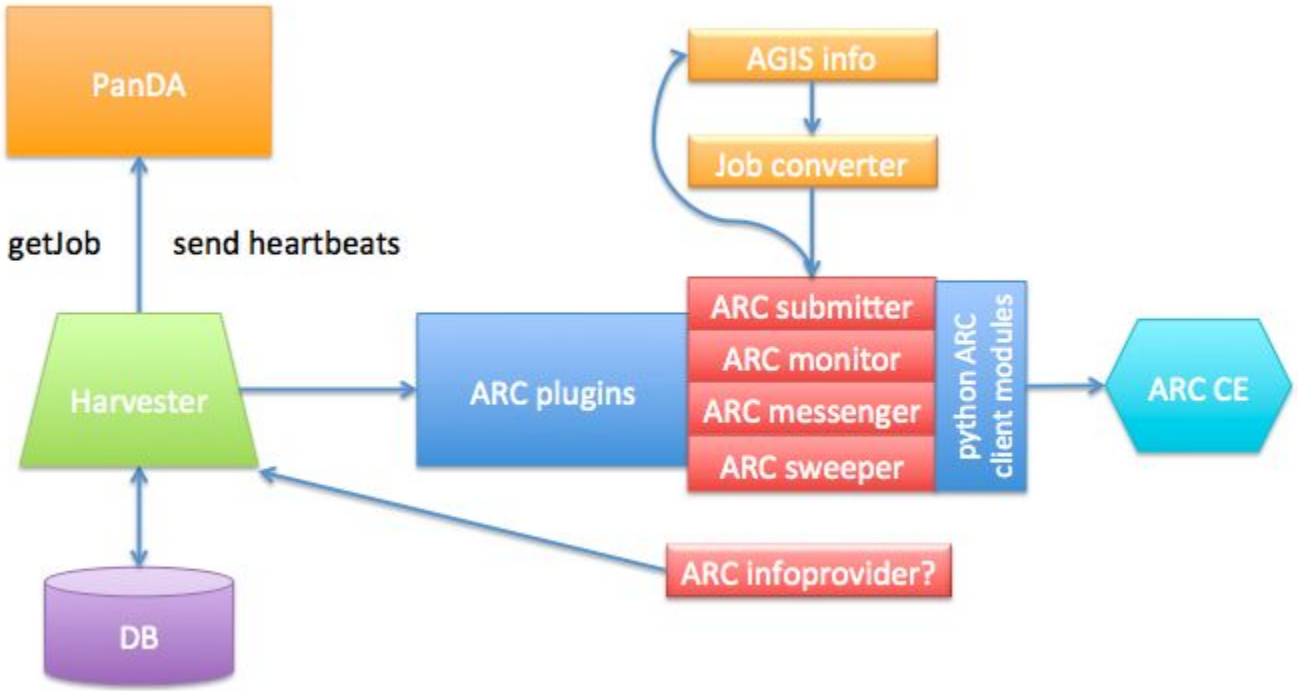
# ARC plugins 1/3

- All ARC modules are fully implemented
  - Submitter, monitor, messenger, sweeper
- Required only very minor modifications in core harvester to work
- Test instance successfully running HC jobs for test queue LUNARC_TEST

https://bigpanda.cern.ch/jobs/?schedulerid=test_dcameron&days=3

| | |
|---|---|
| **computingsite (1)** | LUNARC_TEST (120) |
| **corecount (1)** | 1 (120) |
| **eventservice (1)** | ordinary (120) |
| **gshare (1)** | Test (120) |
| **homepackage (3)** | AtlasOffline/21.0.14 (33) AtlasProduction/19.2.3.6 (30) AtlasProduction/20.7.5.1 (57) |
| **inputfileproject (1)** | mc15_13TeV (120) |
| **inputfiletype (1)** | EVNT (120) |
| **jobstatus (1)** | finished (120) |
| **jobsubstatus (1)** | staged (120) |
| **outputfiletype (1)** | LUNARC_TEST (120) |
| **priorityrange (1)** | 10000:10099 (120) |
| **processingtype (2)** | gangarobot-celpft (23) gangarobot-pft (97) |

# ARC plugins 2/3



# ARC plugins 3/3

- These are jobs running in “full NorduGrid” mode
  - I.e. the most difficult mode where harvester has to handle all Panda communication, download, massage and propagate pilot info etc
  - Now this works, true pilot or pull mode will be easy
- TODO for harvester on grid:
  - Pilot logs, easy configuration, monitoring, ...

https://bigpanda.cern.ch/job?pandaid=3610542656

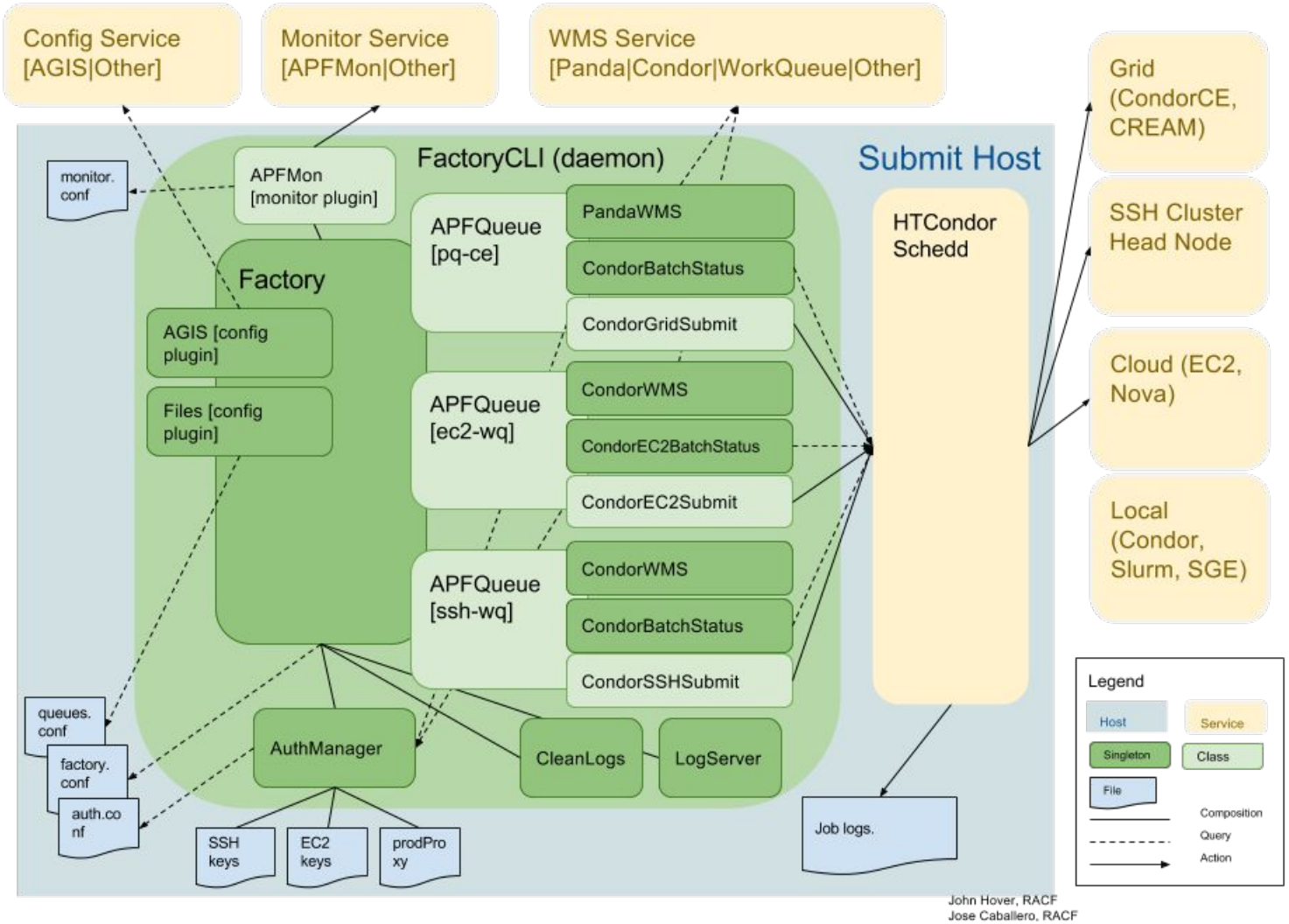
To fix:

| | |
|---|---|
| **pilotid** | http://nowhere/\|NEWMOVER-ON\|PR\|PICARD 70.4 |

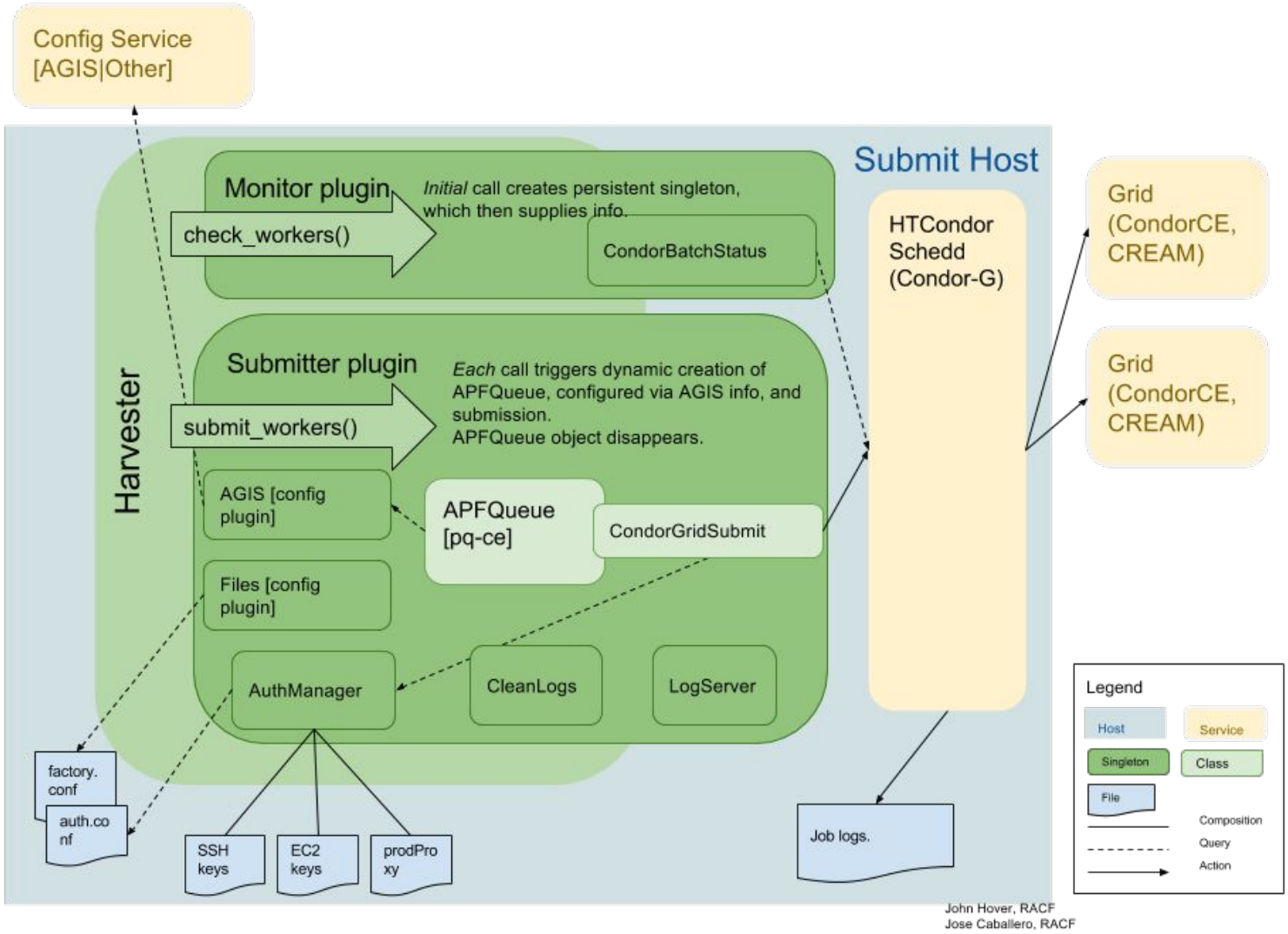
# APF plugins - Status

- Code in fork at: https://github.com/bnl-sdcc/panda-harvester.git
- Deployment scripts/configs/wrapper at: https://github.com/bnl-sdcc/autopyfactory-harvester.git
  - This is very ad-hoc to ease rapid development.
  - Still need a deployment model for real usage.
- Instance running (off and on) at BNL submitting to a single CE at BNL for PQ HARVESTER_APF_BNL_TEST
  - Running HC jobs via standard modular wrapper.
  - NEEDS periodic Panda log rotation/deletion. Where to config?

# APF plugins - Standard APF Layout (standalone)



# APF plugins - Harvester APF Layout (as plugin/library)



# APF plugins - Misc (1)

- Authmanager.py at:
  - https://github.com/PanDAWMS/autopyfactory/raw/master/autopyfactory/authmanager.py
  - Conf file example: https://github.com/bnl-sdcc/autopyfactory-harvester/raw/master/configs/jrh-auth.conf
  - Can be imported and path to proxies retrieved dynamically, OR it can be run as a stand-alone daemon that will maintain a set of proxies with arbitrary VOMS attributes.
  - Can be configured to use multiple certs for one profile, such that if the first cert tried fails (e.g. is expired) it will use the next (and send an alert email).
- Agis.py at:
  https://github.com/PanDAWMS/autopyfactory/raw/master/autopyfactory/plugins/factory/config/queues/Agis.py
  - If desired, I can make the internal processed information ( a Tree of PandaQueues containing CEQueue objects) accessible.

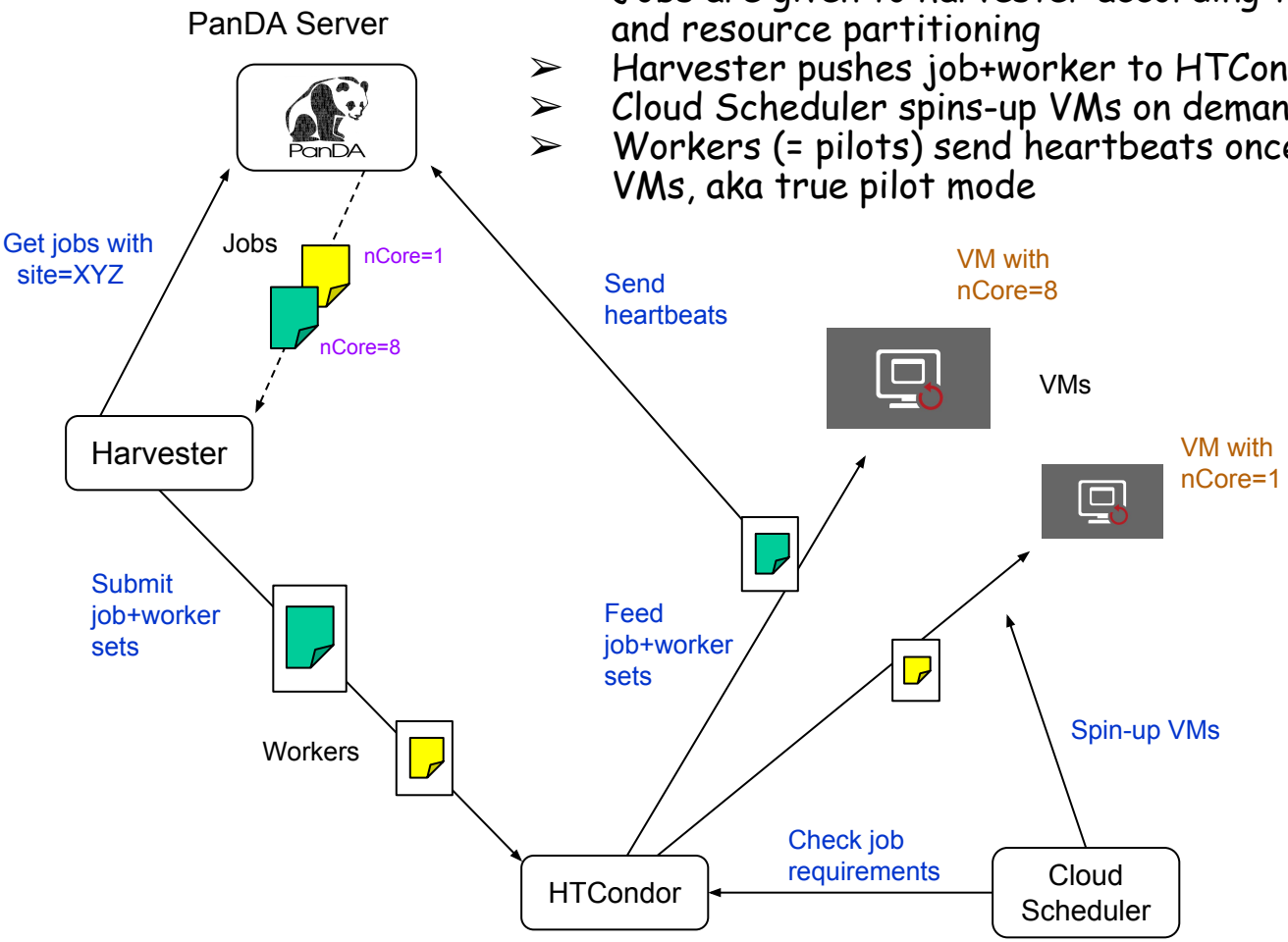
# APF plugins - To Do

- Implement kill_workers() in an APFGrid_sweeper.py
- CE fanout.
  - Plan to submit to a randomly-selected CE from set provided by AGIS for each submit_workers() call. OK? Distributing over multiple CEs from one call will be complicated. (Confirmed that random-per-call is OK).
- max_at_once: Will this be implemented prior to submit_workers() or will it need to be in the submitter plugin as a parameter?
- Two more APF subcomponents to import and use in the plugins (should be easy).
  - Pilot log cleanup (CleanLogs)
  - Pilot log export via HTTP (LogServer)
- Pull request to upstream project.

# Harvester for Cloud

- Good progress thanks to Taiwan and Canadian colleagues
- Full production at CERN OpenStack cloud with 800 CPU cores
- Trying high performance harvester config (MariaDB + multi-processing + apache + multiple harvester nodes) with thousands of nodes in TW for non-ATLAS VOs

- Jobs are given to harvester according to global share and resource partitioning
- Harvester pushes job+worker to HTCondor
- Cloud Scheduler spins-up VMs on demand
- Workers (= pilots) send heartbeats once they land on VMs, aka true pilot mode



# Recent and/or on-going Developments in Harvester Core

# Unification of Production and Analysis PQs

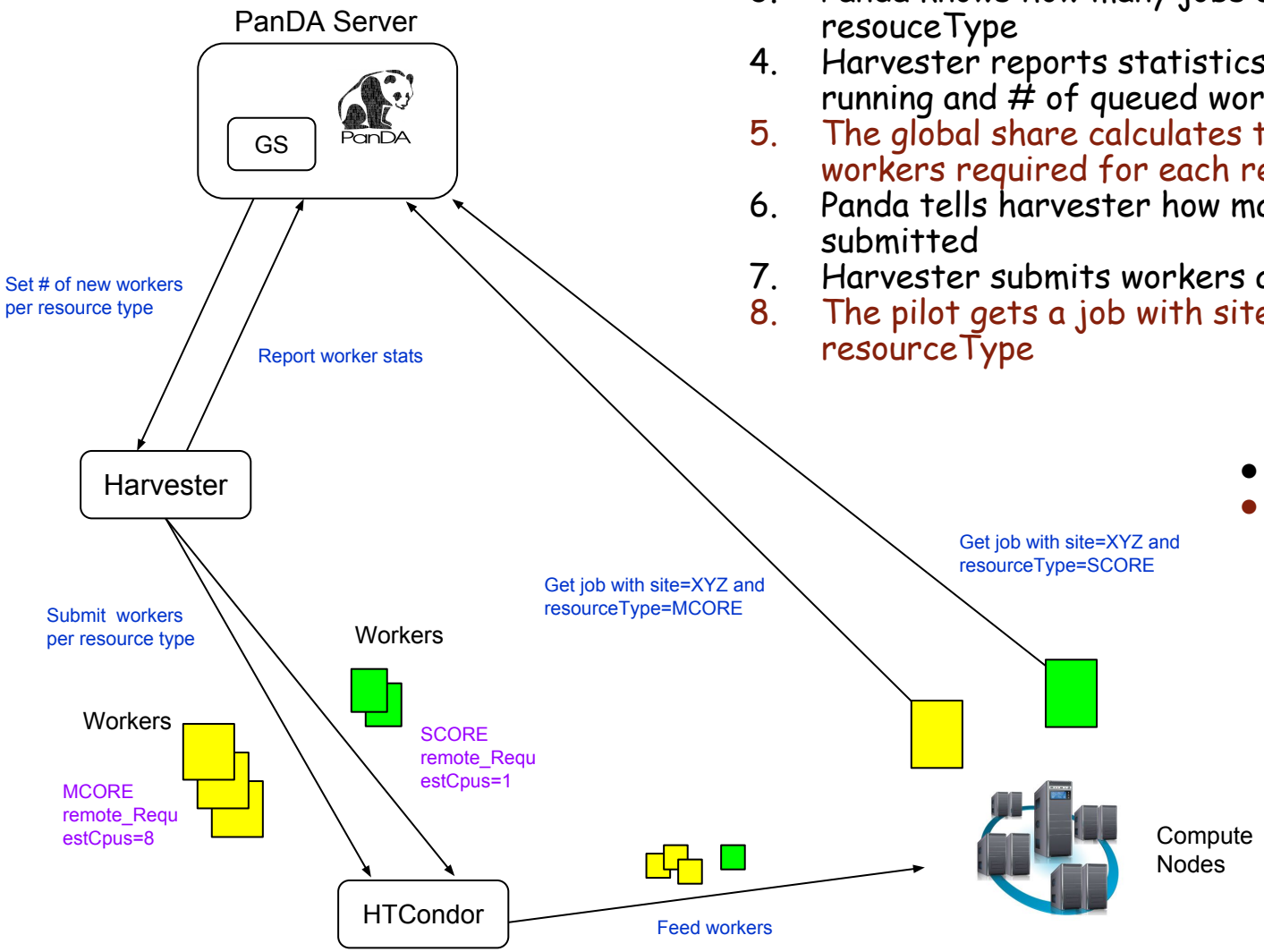
- Unified PQ to consolidate resource-specific PQs to a single PQ
- Production PQs have been unified at LRZ-LMU_MCORE
  - SCORE, SCORE_HIMEM, MCORE, MCORE_HIMEM jobs can run there
  - Full production needs pilot stream control for pull and enhanced global share for push
- The next step is to unify production and analysis PQs
  - Sending pilots with a single credential
  - New resource type ANALY?
- JIRA : https://its.cern.ch/jira/browse/ADCDPA-49
- Issues
  - Analysis and production have different default DDM endpoint
  - Analysis uses direct IO while production uses copy-to-scratch at some sites

  Some queue attributes are different between production and analysis → Not so simple to instantiate pseudo resource-specific PQs from a single unified PQ
  - For default DDM endpoint, new table to associate resource type and endpoint type?
  - For direct IO, unified PQs could specify all attributes for directIO and the pilot would change behaviour depending on a job parameter, e.g. job.transferType=c_t_s for production by default, job.transferType=direct for analysis → Easy to enable directIO for production
- The pilot may need a mechanism to strip prod/pilot role when executing analysis payload
  - Executing voms-proxy-init on WN
  - Downloading cached proxies from Panda
    - Used by harvester to renew proxies on HPCs

# Unified PQ + Pull

## Pilot stream control

1. All jobs have the same computingSite=XYZ (not XYZ_SCORE, XYZ_MCORE, XYZ_HIMEM, …)
2. Each job defines resourceType (e.g., SCORE, MCORE, HIMEM, …)
3. Panda knows how many jobs are activated for each resouceType
4. Harvester reports statistics of workers (# of running and # of queued workers) to panda
5. The global share calculates the number of new workers required for each resource type
6. Panda tells harvester how many workers should be submitted
7. Harvester submits workers accordingly
8. The pilot gets a job with site=XYZ and resourceType




- Available
- To be implemented

# OneBox

- To run CPU-intensive jobs with low priorities on a WN when IO intensive jobs are running on the WN
- Implementation
  - Panda side
    - Jobs are flagged if they are good for background execution based on priorities and IO intensity measured by scouts
    - Flagged jobs are dispatched when getJob requests come with backGround=True or when there are no high priority jobs waiting
  - Harvester side
    - Two internal queues with the same PQ
    - Getting normal jobs for one queue and background jobs for the other
    - Sending pilots+normal jobs for one queue with normal ClassAd and pilots+background jobs for the other queue with special ClassAd
  - Batch system side
    - Condor slots overcommitting resources
    - 1/2 slots match with the special ClassAd and execute payloads with nice
- Tried at CERN-EXTENSION_HARVESTER but not very successful due to bad configuration in condor
  - Not sure if the idea is still relevant, but better to retry?

# Removal of Empty Pilots

- Two main causes
  - Attracting jobs at free PQs
    - Useful to find faulty PQs which are not blacklisted but pilot cannot run there
    - Expensive to run empty pilots at special PQs like MCORE and HIMEM
  - Over submission of pilots from multiple schedulers due to lack of the system overview
    - Large sites need multiple schedules to feed enough pilots
    - Each scheduler doesn't know what others are doing
- Solution
  - To unify PQs and send empty pilots only to SCORE
    - Negligible as one empty pilot per 30min
  - To use the slave mode of harvester for pull
    - Panda has central knowledge on all harvester instances, e.g., how many instances are working for each PQ, how many pilots are waiting in the batch queue, …
    - Panda tells harvester instances how many new pilots they should submit

# Resource Usage of Harvester

- Heavy CPU usage and large memory consumption were reported at Theta/ALCF (~200k cores)
- CPU issue solved
  - Actions
    - Code refactoring
    - Optimization of sqlite database
- Memory issue is still being investigated
  - Actions
    - Improved to work both with python2 and python3
      - Heap memory is returned to OS in python3
    - Added a new capability for memory usage monitoring which appends to each logging message the information of memory usage at that time
    - Slimming jobReport.json?
      - Sometimes too large. e.g. 3597120705 → 8MB with many "message": "-------- WWWW ------- G4Exception-START -------- …" which is loaded into RAM
      - d = json.load(old, old_file); for e in d["executor"]: e["logfileReport"] = {}; json.dump(d, new_file) can reduce the 8MB to 3kB
      - Better to put the code in (mini) pilot until trfs are fixed?

# Plans for the grid and cloud

- Commissioning for the grid with APF and ARC plugins
- New agent to collect realtime resource information (details in Alessandro's talk)
- Adding larger cloud resources in production
  - TBD with cloud community
    - E.g. UVic cloud in CA, HLT farm at CERN, ...
- Unification of PQs
- Pilot stream control for pull and enhanced global share for push
  - E.g. SCORE vs MCORE
- Removal of empty pilots at expensive PQs
- Protection against over-submission of pilots leading to empty pilots
- Integrated worker and resource monitoring
  - JIRA ticket https://its.cern.ch/jira/browse/ATLASPANDA-395 for the worker status summary as the first step
- Direct diagnostics for lost-heartbeat jobs
- Adding a mechanism for workload provisioning