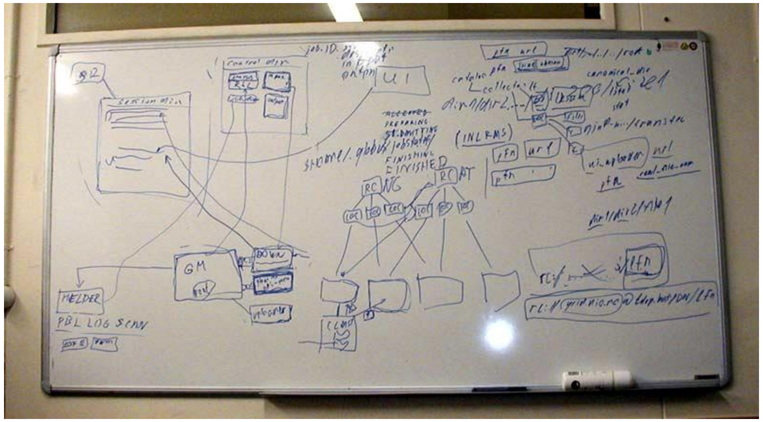
NORDUGRID

Grid Solution for Wide Area
Computing and Data Handling

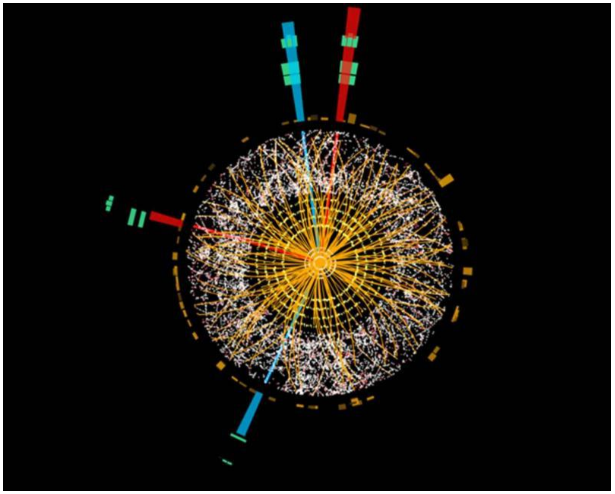
ARC for ATLAS computing

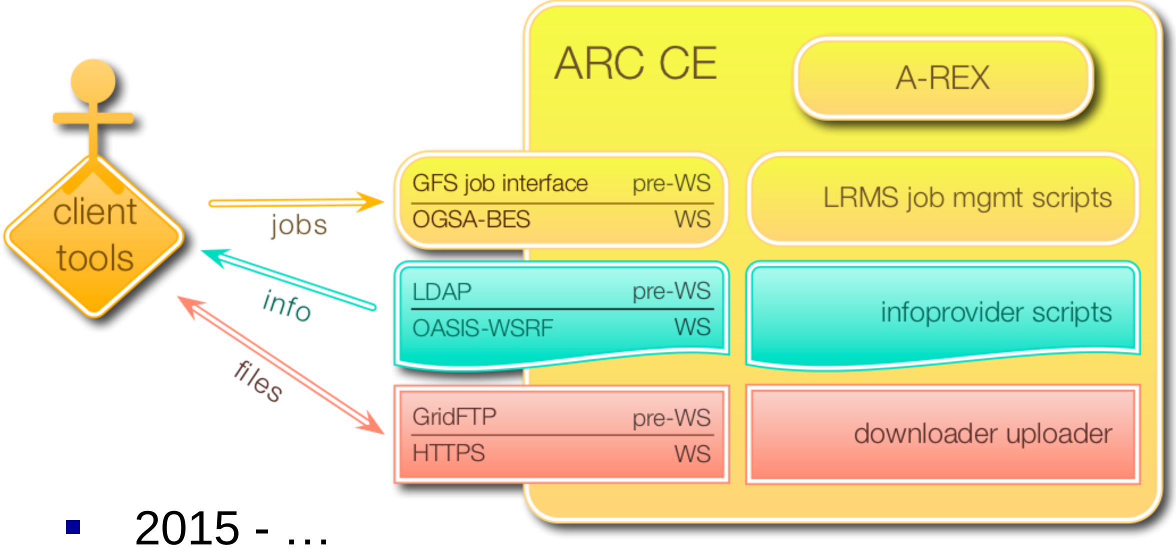
Jon Kerr Nilsen,
ARC Release Manager

ARC Highlights



- 2001- 2002
  - "Transverse momentum distribution" of pre-LHC data simulated on HPC resources …
- 2012 - 2013
  - Higgs discovery, Nobel prize



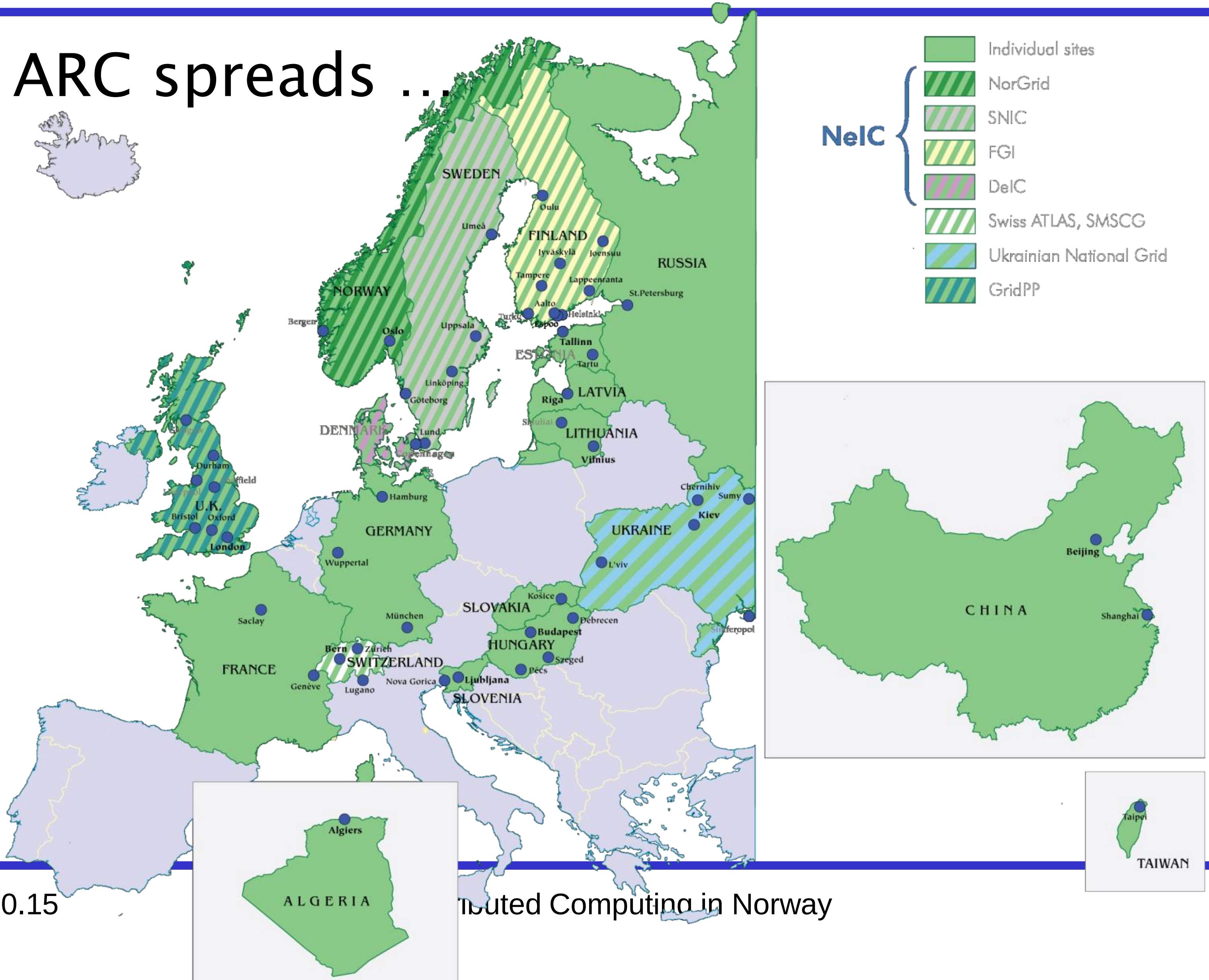


- 2015 - …
  - 13 TeV collisions, Dark Matter?

NORDUGRID
Grid Solution for Wide Area Computing and Data Handling

ARC Deployment Map

- ARC spreads …

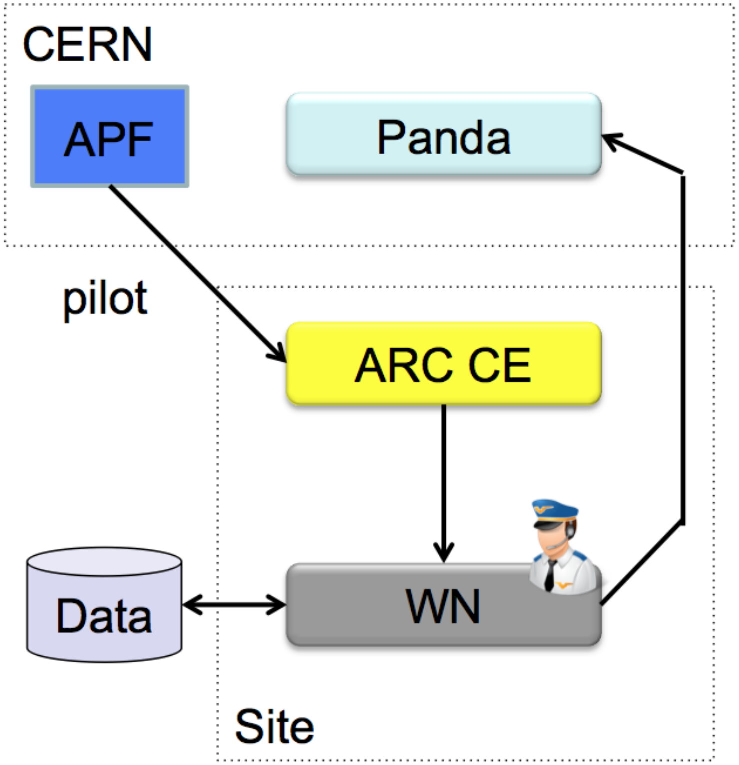


ATLAS ARC workflows

- APF gateway
- Truepilot
- ATLAS@HOME
- Distributed T1
- Chinese style
- Storage free sites

APF mode – ARC as a Gateway

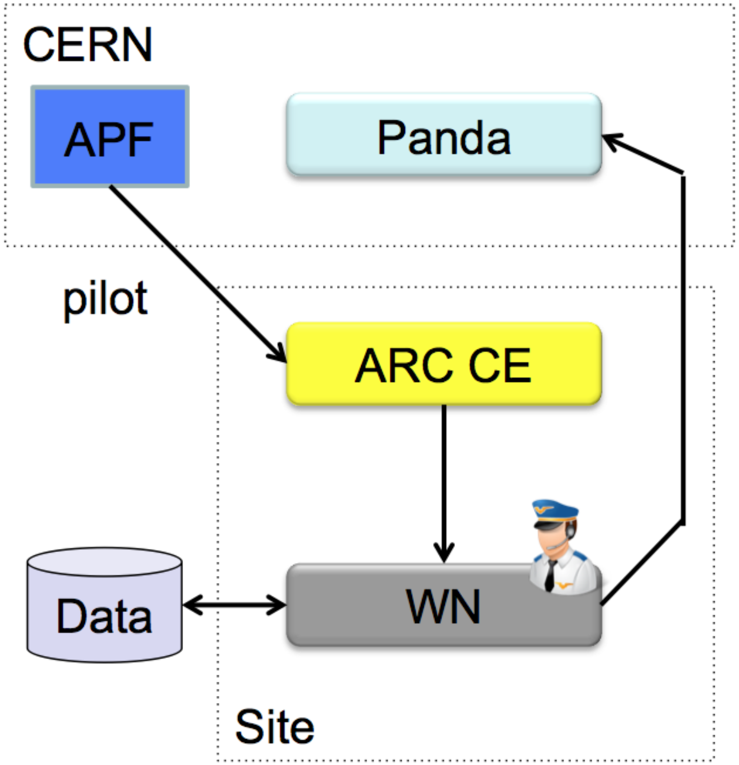
- Simplest case
- APF sends jobs to ARC CE
- ARC CE submits job to Worker Node
- Worker Node pulls PanDA for payload
- All data traffic directly to Worker Node



Worker Node pulls PanDA for payload

APF mode – ARC as a Gateway

- Works nicely for smaller WLCG specific sites
- If you just need to replace CREAM, this is it
- ARC in APF mode is supported by all LHC experiments



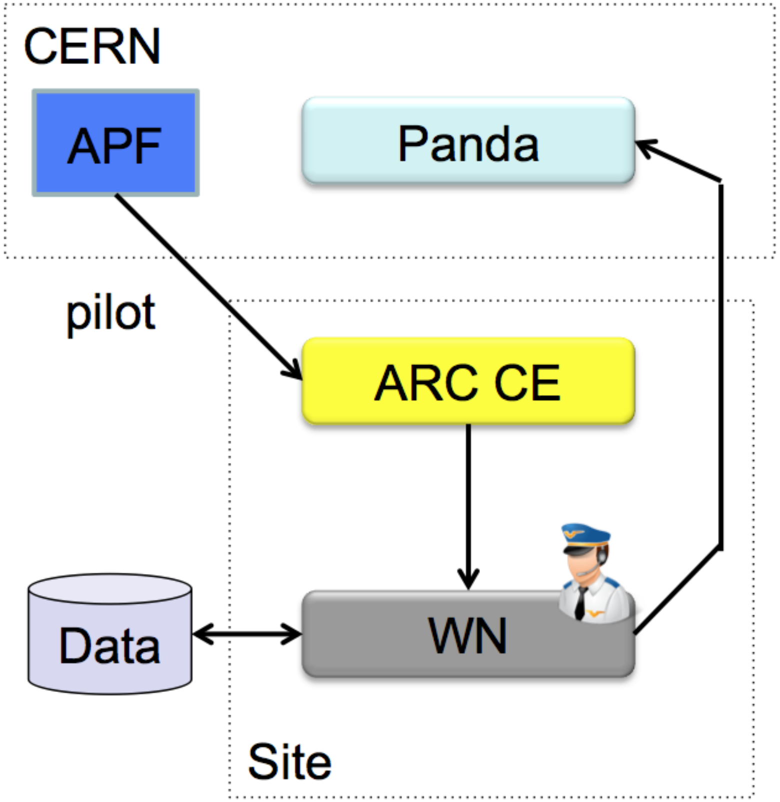
Worker Node pulls PanDA for payload

Payload submission practice

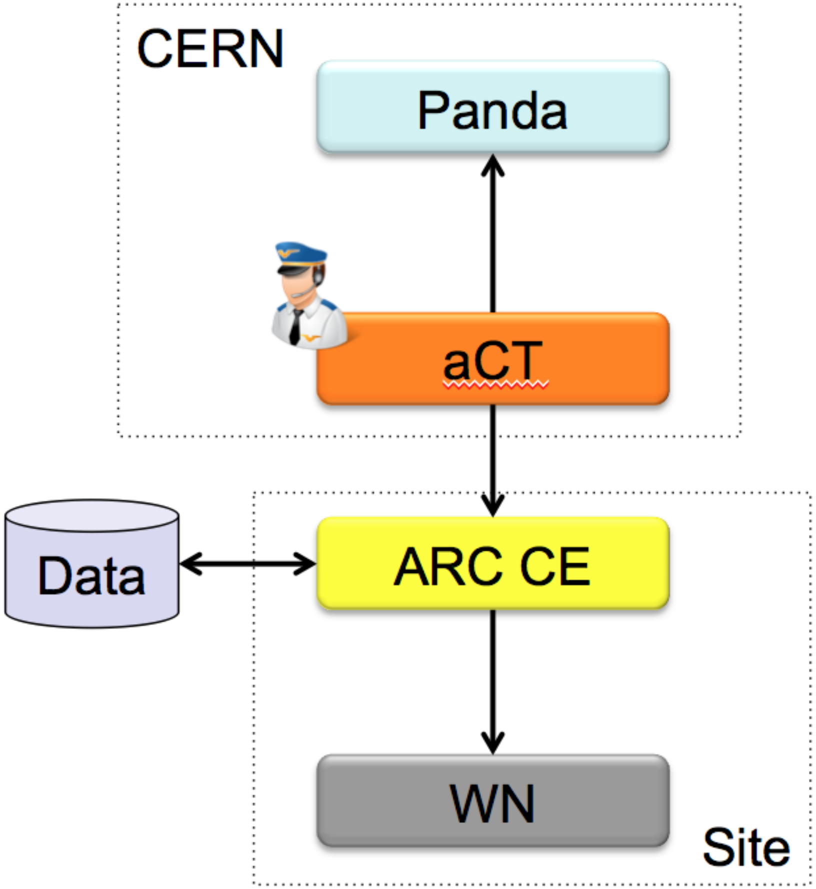
- Push model – direct (full) job submission to grid sites was terribly inefficient and unreliable 10 years ago:
  - failure rates were exceeding 50%
  - Workload management systems could not cope with the submission rate and complexity
- Pull model gained on popularity
  - Dummy batch jobs – pilots – pull the payload from central services
  - Local site instabilities have less impact on central submission service
  - But all the pilot jobs are the same – uniform memory, walltime and cpu requirements

Pilot mode works well only if everybody is happy with equal job resources

Push vs Pull Model



Worker Node pulls PanDA for payload



Payload is pushed to the Worker Node by intermediate service

Ideal distributed model

- An extended/distributed “batch” system
  - Worker nodes – full nodes allocated to external “batch” scheduler (PanDA)
  - Permanent pilots – “batch daemon slaves” – ask for payload
  - Central scheduling system (PanDA) distributes job to the pilots according to priorities and job requirements for resources
- Central scheduling system would manage all users (VOs)
  - Fair-sharing between VOs
  - Common job priority treatment

Was not even planned at the start-up of the grid computing

Distributed Reality

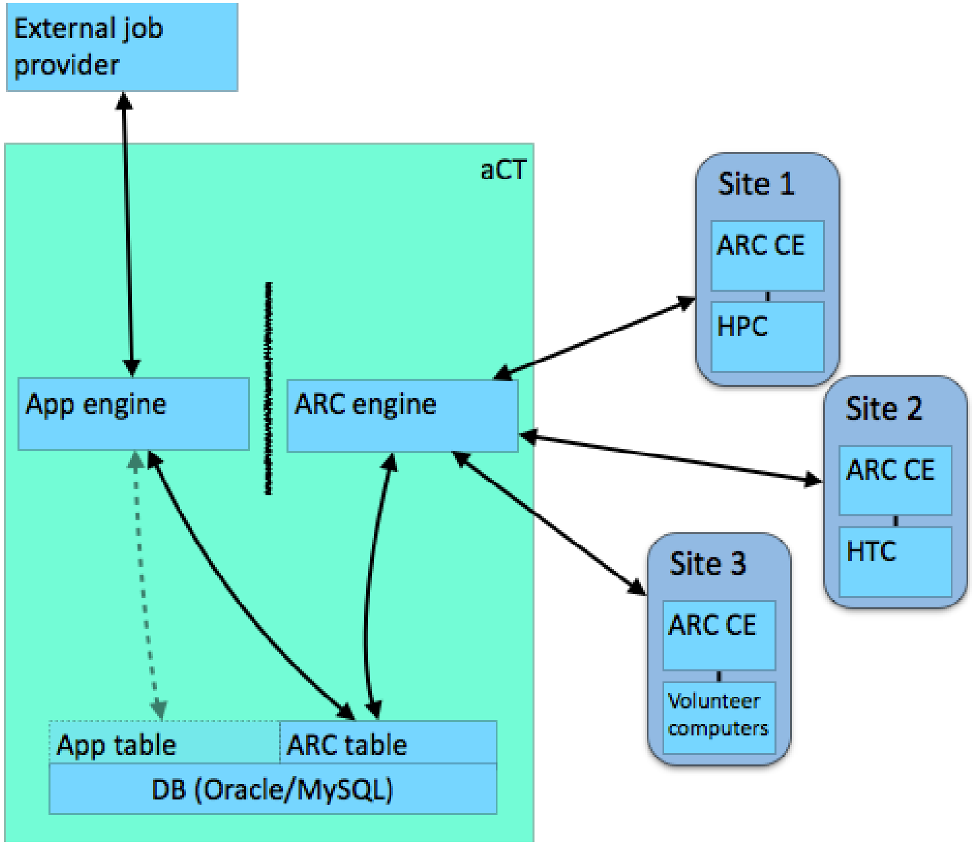
- Sites are still using the conventional batch systems to submit the jobs to clusters
  - We need to deal with multi-level scheduling
  - Central scheduling system and sites need to adapt to each other
- Pilots with uniform resource requirements not good enough any more:
  - ATLAS uses different workloads by memory, cputime, corecount requirements
  - Even worse if other VOs use completely different requirements – simple batch system configuration is not sufficient any more
- Workaround for ATLAS PanDA:
  - Each site has many custom queues, corresponding to different workload requirements:
    - RAL-LCG2_SL6 – default queue
    - RAL-LCG2_MCORE – 8-core
    - RAL-LCG2_HIMEM_SL6 – more memory
    - RAL-LCG2_VHIMEM – even more memory
    - ANALY_RAL_SL6 – analysis
  - When the tasks with new requirements are to be launched (“insane memory”) a new PanDA queue needs to be defined for each site
  - Difficult to maintain long term – after two years of multicore life, there are still sites without multicore support

Issues with uniform payloads

- Some sites are shared with other VOs, or are general purpose clusters (e.g. supercomputers)
  - Fixed partition allocation does not make sense
  - Shorter jobs would get more cpu resources – backfilling
  - Long (2 day) jobs cannot start on empty extra worker nodes – draining is too expensive for sites
- ATLAS job resource requirements – wide spectrum:
  - 0.5GB to 6GB of memory
  - Minutes to 4 days of walltime
  - 1 to 32 cores
  - Massively parallel jobs coming into ATLAS production – AthenaMP spanning several nodes (Yoda)
- Static PanDA queues are becoming difficult to maintain and use

ARC Control Tower

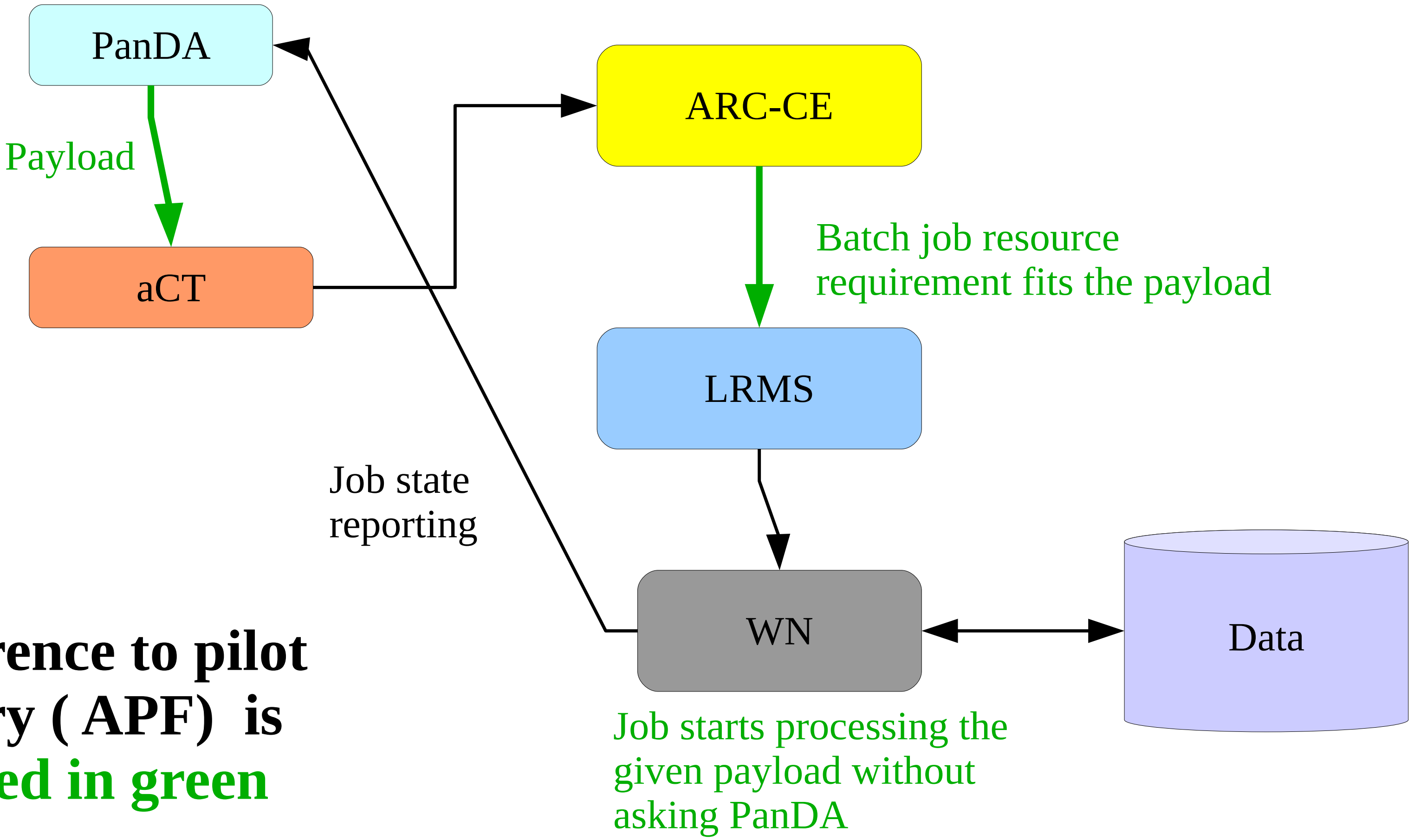
- aCT is a job management layer in front of ARC CEs
  - Picks up job descriptions from external job provider
  - Converts them to XRSL job description
  - Submits and manages jobs on ARC CEs
  - Fetches logfiles, validates output, handles common failures and updates job status



Modes of aCT job submission

- ARC native mode:
  - aCT communicates with PanDA and submits predefined payload to ARC-CE
  - ARC-CE transfers input and output files and submits to the batch
  - Pilot wrapper on worker nodes only executes the payload without accessing the external network
    - Outbound connectivity still used by CVMFS and Frontier
  - Worker nodes do not use grid middleware
  - Good for sites with capable shared filesystem with caching of input files, as well as HPC sites
- Truepilot mode:
  - aCT fetches the payload and submits it to the ARC-CE
  - ARC-CE submits the batch job with predefined payload
  - Pilot on the worker node does the same as on the conventional pilot sites, but skips the fetching of payload from PanDA
  - Good for worker node centric sites with capable local disk space and fast transfers to close storage site

aCT Truepilot

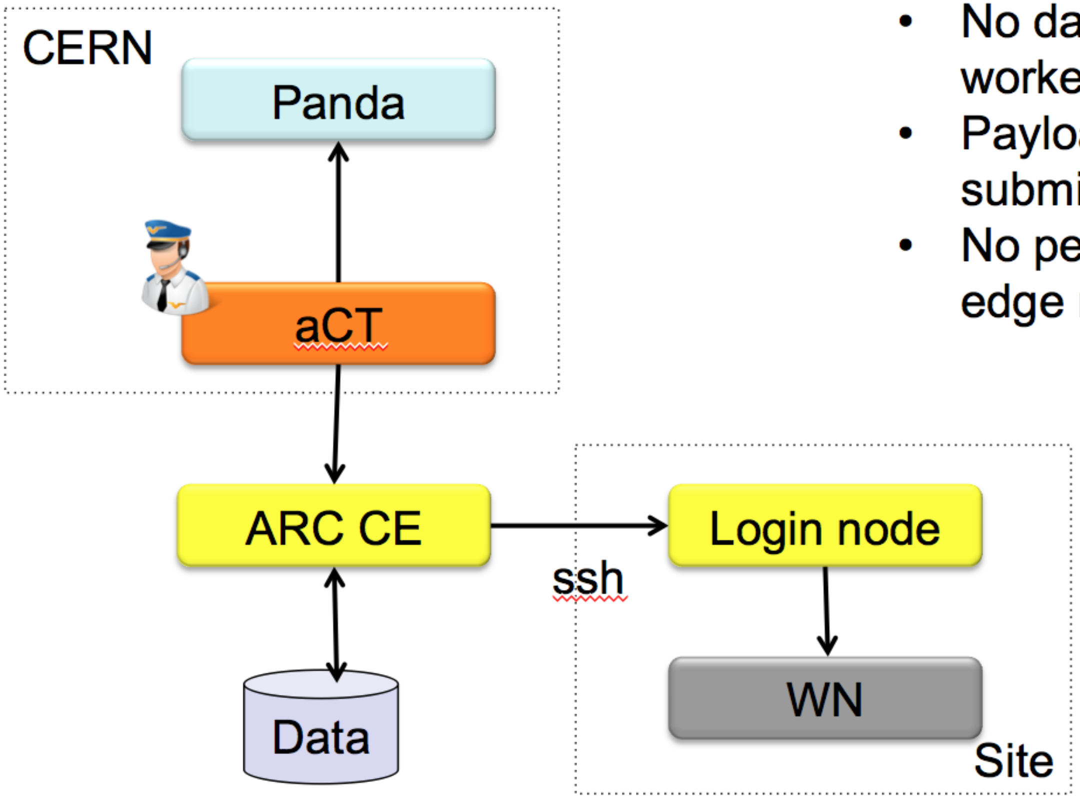

PanDA
Payload
aCT
ARC-CE
Batch job resource requirement fits the payload
LRMS
Job state reporting
WN
Data
Job starts processing the given payload without asking PanDA

Difference to pilot factory (APF) is marked in green

Pilot factory vs aCT Truepilot

- Pilot factory:
  - Highest priority jobs start running first
  - But the batch jobs have all the same resources
- aCT truepilot:
  - Payload known in advance – the batch job has the resource requirements fit to the job
  - Payload can request any memory, cputime, corecount, of course in agreement with site capabilities
  - But the late-binding is partially lost – highest priority jobs need to wait some time in the batch
  - Bad worker nodes can cause black holes – fast resubmission cycle

aCT and Supercomputers - HPCs



- No data access from worker node ✔
- Payload known at job submission ✔
- No persistent service on edge node or open ports ✔

Using aCT native node

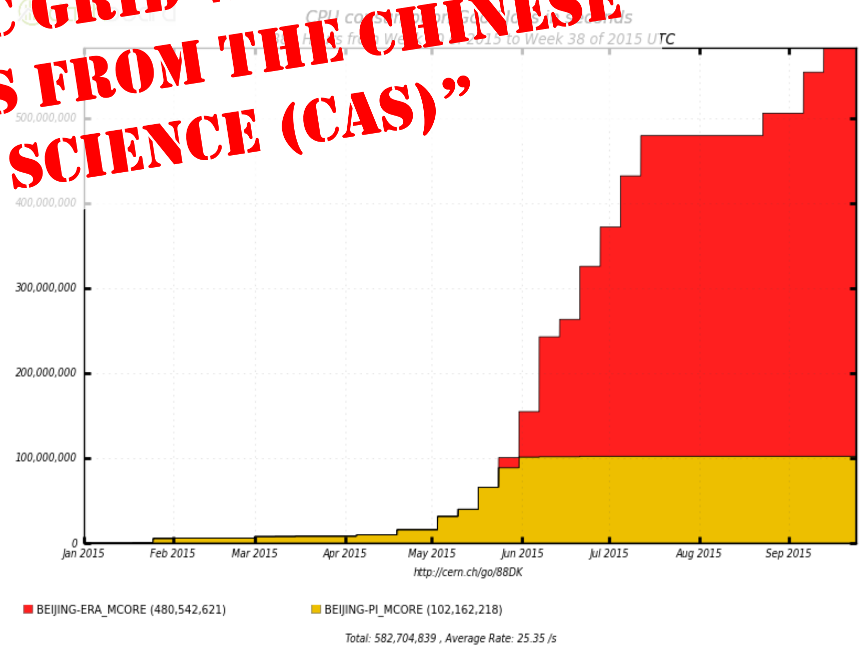
ATLAS+ARC in China

- Remote ARC CE
- New Python backend
- Team of 7 developers sited in China, Switzerland and Norway

“THE INTERFACE BETWEEN ARC-CE AND THE CHINESE HPC GRID WON ONE OF THE 4 HPC AWARDS FROM THE CHINESE ACADEMY OF SCIENCE (CAS)”

Two systems:

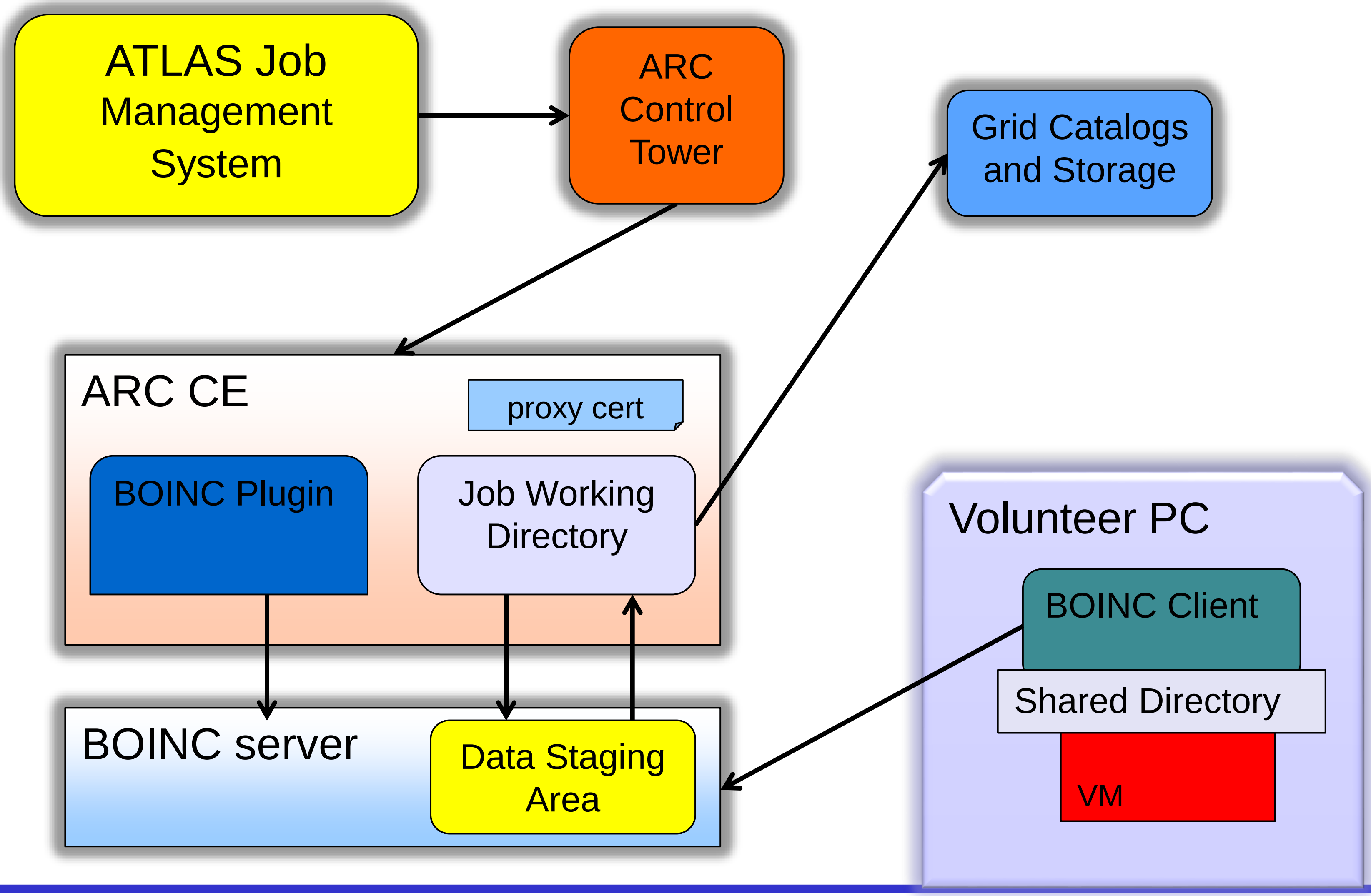
- Pi – CE in Beijing, jobs through ssh to Shanghai
- ERA – CE in Beijing connects to Chinese HPC Grid



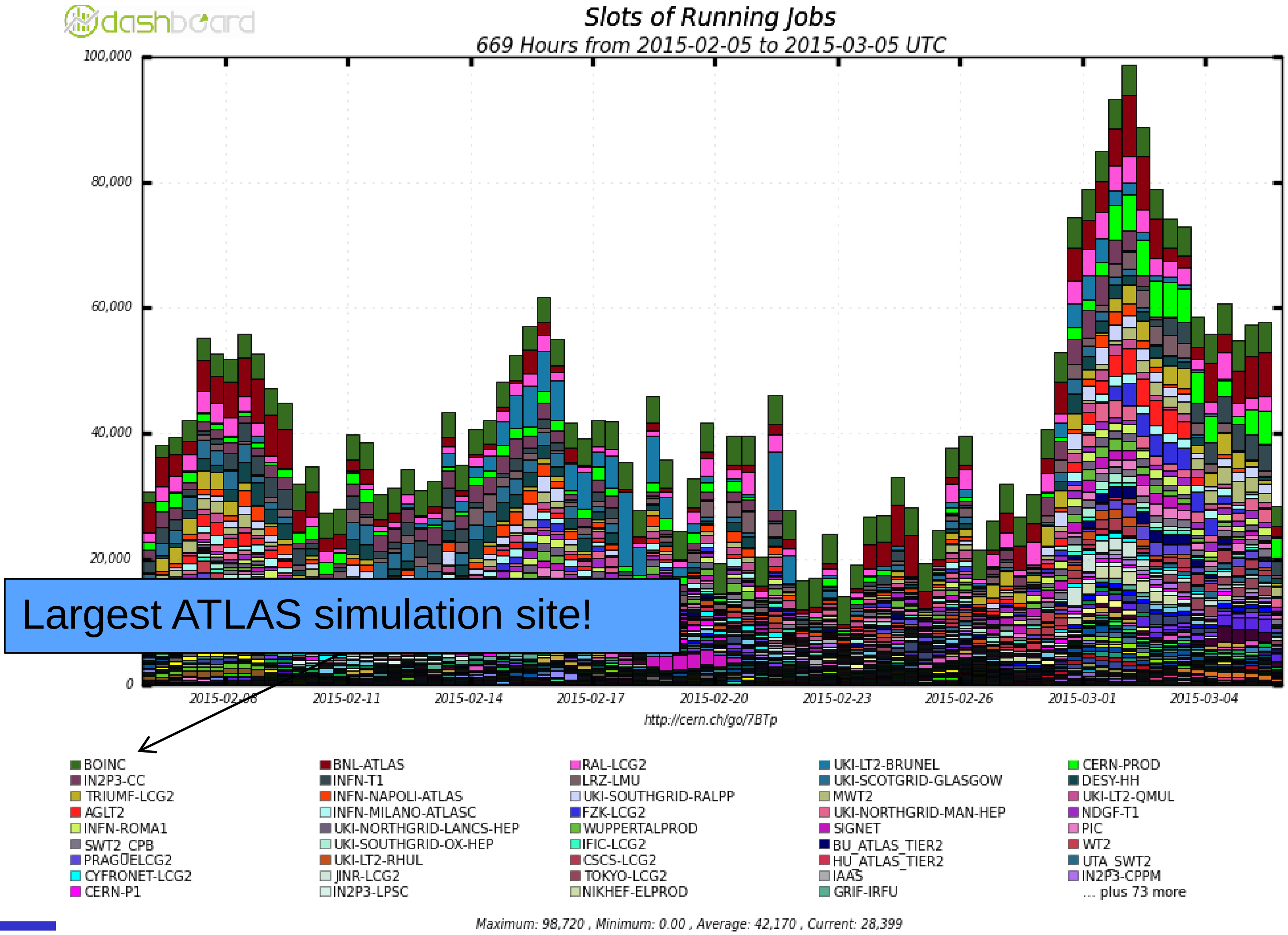
Volunteer computing

- Why use volunteer computing for ATLAS?
  - It’s free! (almost)
  - Public outreach
- Considerations
  - Low priority jobs with high CPU-I/O ratio
    - Non-urgent Monte Carlo simulation or specific tasks
  - Need virtualisation for ATLAS sw environment
    - CERNVM image and CVMFS
  - No grid credentials or access on volunteer hosts
    - ARC middleware for data staging
  - The resources should look like a regular ATLAS computing resource
    - ARC Control Tower

Basic ATLAS@Home Architecture



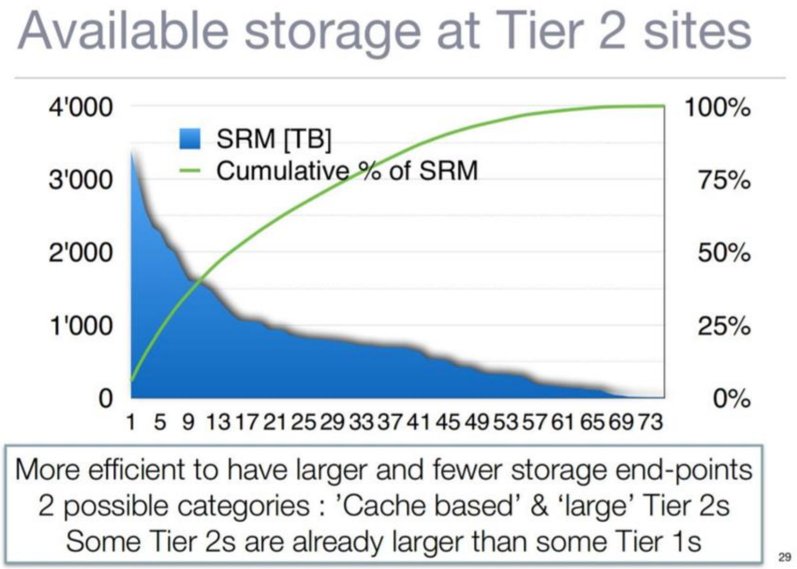
Scale of ATLAS@Home



Largest ATLAS simulation site!

Storage-less sites - why?

- Some sites have very limited storage/manpower
- From operational experience, the small sites are the ones that generate most of the problems/operational load:
  - Lost files
  - More Dark Data than bigger sites
  - Sometimes have to reduce space, change SE hostname/path...
  - Sometimes not very responsive



Available storage at Tier 2 sites

More efficient to have larger and fewer storage end-points
2 possible categories : 'Cache based' & 'large' Tier 2s
Some Tier 2s are already larger than some Tier 1s

29

Solution: Use cache instead

- Different cache types
  - Secondary files :
    - Files residing on normal Rucio Storage Element but can be deleted whenever space is needed
    - Deletion based on LRU logic done by Rucio
    - Not the purpose of this talk
  - “Internal cache”, i.e. cache that is only accessible from the site.
    - For local jobs
    - Not registered in Rucio
  - Cache site
    - Can be accessed from the WAN
    - Needs to have the data registered in Rucio. Will help for brokering
    - There might be some inconsistencies between the cache and the catalog

Cache in ARC

- Cache is used for a long time in Nordugrid
- ARC CE has a built-in cache
  - It stores files that are used as input on a given CE
  - The cache is usually on a shared file-system accessible from the WN
  - The space is managed by ARC CE, deleting least recently accessed files when space is needed
  - Typical size for an NDGF-T1 site: 100TB
- By default “internal cache”, i.e. not accessible from the outside world

Cache in ARC

- This cache model can only be used with the “full” ARC CE setup
  - i.e. aCT submitting pre-defined payloads and ARC CE doing data staging
- Not appropriate for “ARC as a pilot gateway”
- ARC cache recently integrated in Rucio
  - Will allow brokering of panda jobs to sites where files are cached
  - No problem if files are deleted by the time the job gets there, ARC will download

Conclusions

- ARC connects resources
  - Pilot gateway, HPC frontend, remote frontend, volunteer computing service…
- Excellent option for smaller sites that don’t want to maintain their own storage (or a CREAM CE ☺)
- Works on all resources
  - Only compute element needed
  - No need for software or external network access on worker nodes
  - No storage required