ARC Data Staging and Cache, Integration with Rucio

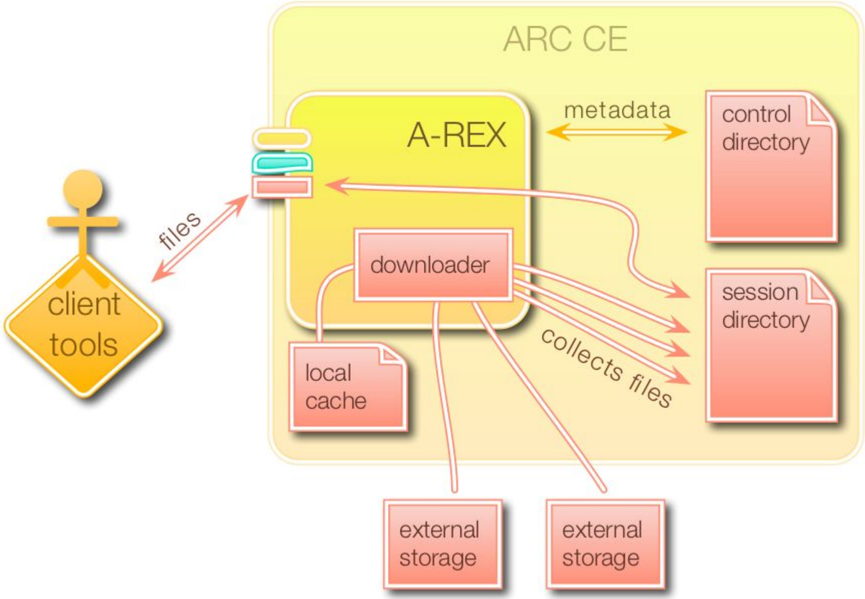
David Cameron, Vincent Garonne
University of Oslo

pre-GDB, 19 Nov 2019

(some slides stolen from Mattias Wadenstein and Maiken Pedersen)

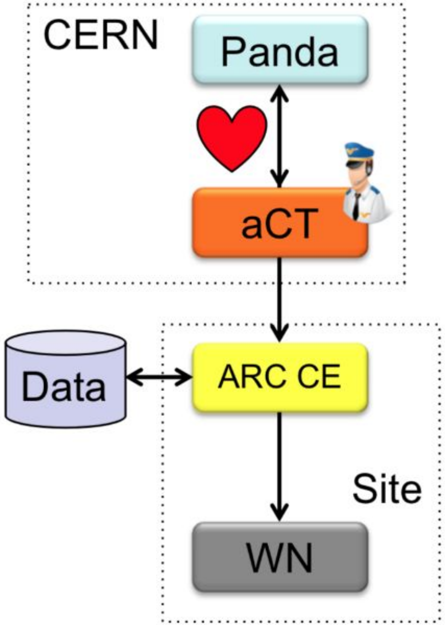
Data Staging

- ARC can do data staging
- Prepares all input files needed by the job before submission to batch system
- Saves all requested outputs to remote storage afterwards
- Cache for reuse of input files between jobs



Data Staging

- ARC in data caching mode requires the NorduGrid mode used by ATLAS
  - Payload is picked from central WMS then pushed to ARC CE
  - The job description sent to ARC CE has a list of input and output files
  - The CE stages all these files to local cache and links them in the session directory
  - The job is submitted to batch system and runs on local files only
  - Afterwards the listed output files are uploaded to SEs
- Caches are normal shared filesystems
  - NFS, CephFS, GPFS, Lustre, etc



Advantages

- Overall efficiency
  - Data access is on low-latency local filesystems
  - Download before submission to batch system → better CPU efficiency
- Non-local storage
  - Like NDGF with distributed storage
  - Or a “compute only” site
- Limited external connectivity
  - Like HPC sites where external connectivity might be blocked or only available through a slow NAT
- No need for grid-aware computational software

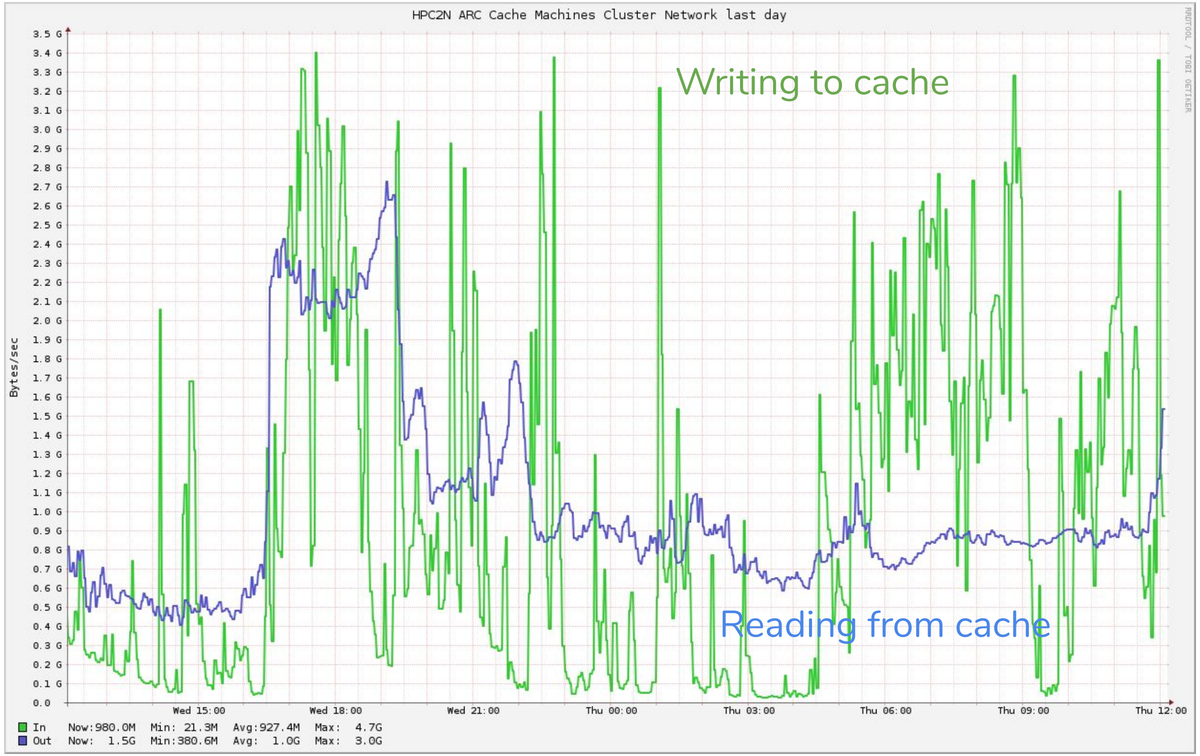
ARC on HPC

- ARC CE + staging has been used not only for NDGF (see next talk) but many HPC centres:
- Piz Daint (CSCS): See previous talk
- SuperMUC (Munich) and IT4I (Prague): No CVMFS, no network, ATLAS jobs are preempted by higher priority users
  - Run ARC CE outside the HPC, use ssh/sshfs to access login node with batch commands and shared filesystem
  - Run event service - after a job is finished or preempted, ARC CE takes care of uploading what was produced
- Chinese HPCs: a “grid” of HPCs accessible through a REST API
  - An ARC batch system plugin for this API was written
  - ATLAS has run on ERA-II, PI, TIANHE-1A
- Also used at Marenostrum (Barcelona), Toronto HPC, Tokyo HPC, DRACO (MPPMU)
  - All have restricted network access from worker nodes
- ATLAS@Home on BOINC (not HPC!)
  - No credentials on volunteer hosts

Cache Experience

- From NDGF ATLAS usage (other communities might have different IO patterns)
- About 100TiB is sufficient cache space to support a few thousand cores of ATLAS compute
    - This is for production + analysis
    - In case of running only simulation (like some HPC), small input files are shared by ~10 jobs, only a few TB cache is necessary
- Bigger will have better cache reuse
    - Sample point, a 204TiB cache for ~4k ATLAS cores:
    - 50% of files accessed within 24h
    - 90% of files accessed within 48h

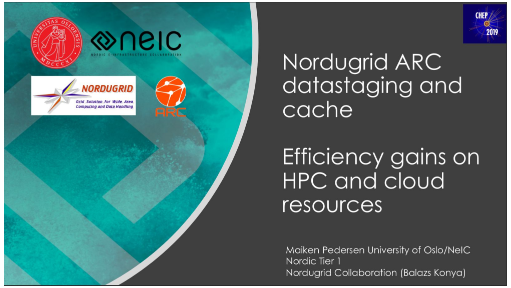
Cache traffic (4k cores ATLAS)



Writing to cache

Reading from cache

Efficiency study

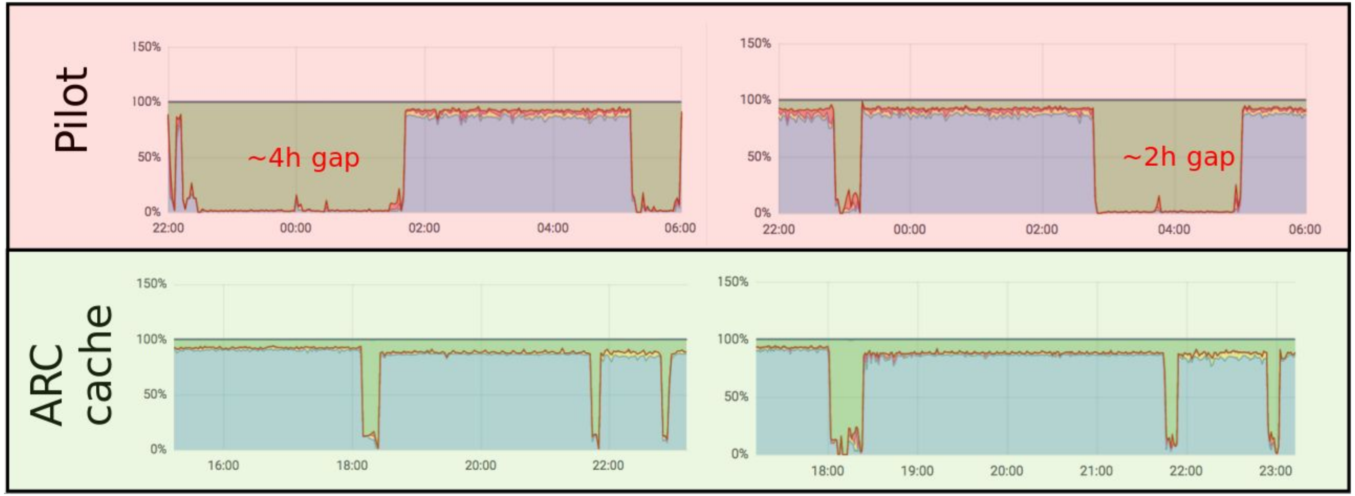


Nordugrid ARC datastaging and cache

Efficiency gains on HPC and cloud resources

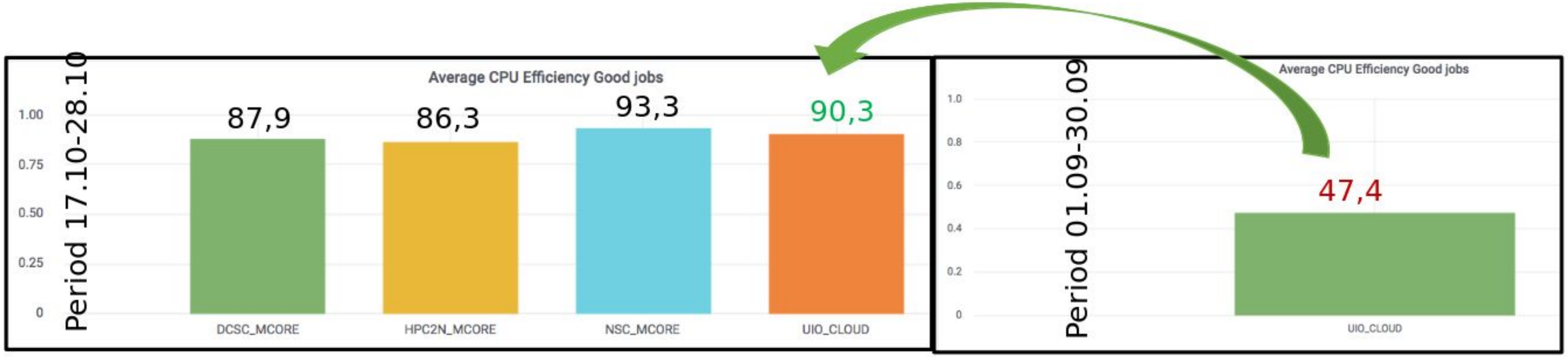
Maiken Pedersen University of Oslo/NeIC
Nordic Tier 1
Nordugrid Collaboration (Balazs Konya)

- Comparison of pilot jobs and ARC caching presented by Maiken at CHEP
- Pilot staging from remote storage lowers CPU efficiency
- Asynchronous staging by ARC CE eliminates the holes in CPU usage



Efficiency study

Comparison of NDGF sites CPU efficiency

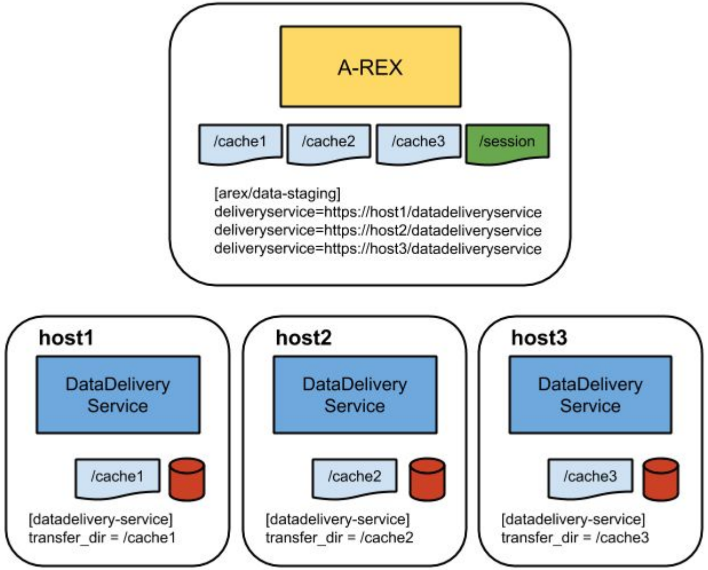


NDGF ARC cache MODE – NDGF sites

UIO_CLOUD Pilot mode

Data Staging Design

- Caches and session directories are placed on shared filesystems between CE and WNs
- In a basic setup, the CE host performs all data transfers
- To scale up, DataDeliveryService nodes can be added
  - They transfer data in and out from the session directory and caches
  - Can be one or several, depending on the data rates you want to support
  - One common deployment is to have 5-15 NFS servers all running a DDS for the local filesystem
- Caches are automatically cleaned using LRU



```
[arex/data-staging]
deliveryservice=https://host1/datadeliveryservice
deliveryservice=https://host2/datadeliveryservice
deliveryservice=https://host3/datadeliveryservice
```

```
[datadelivery-service]
transfer_dir = /cache1
```

```
[datadelivery-service]
transfer_dir = /cache2
```

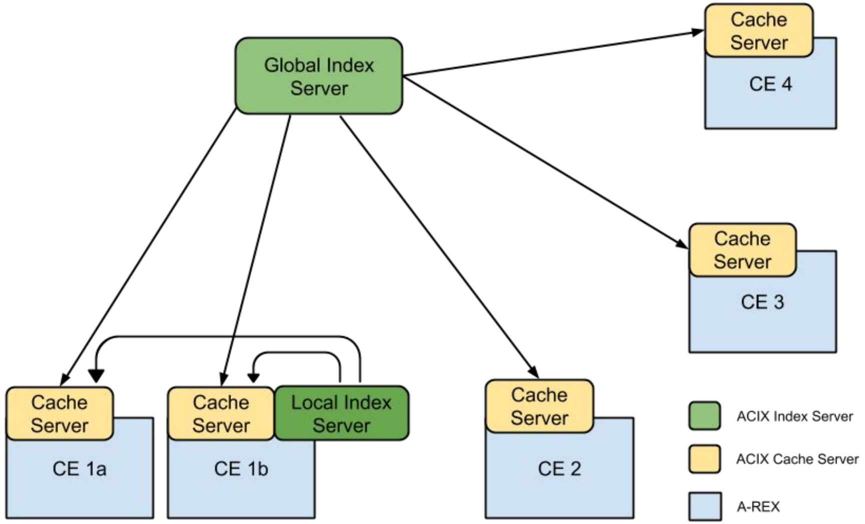
```
[datadelivery-service]
transfer_dir = /cache3
```

Data staging protocols

- Largely influenced by WLCG evolution, the current data transfer protocols supported by ARC are:
    - ACIX (ARC Cache Index)
    - File
    - GridFTP
    - HTTP(S)
    - LDAP
    - Rucio (ATLAS data management system)
    - SRM (Meta-protocol for access to WLCG storage, now deprecated)
    - S3
    - Xrootd (Native protocol to access files stored in ROOT format)
    - LFC, dcap, rfio, … (legacy WLCG protocols supported through gfal2 library)
- Note that ARC CE does not do 3rd party transfer, all data is transferred to or from a local file system

More on cache, ACIX and Candypond

- Several cache-related services exist:
    - CandyPond: extension of A-REX service allowing on-demand caching of files by a running job
    - CacheAccess: extension of A-REX service allowing the cache to be exposed to the outside
    - ACIX: A catalog of cache content - useful for brokering jobs to CEs where data is already cached
    - Whistleblower: Publication of cache content to an external service through message queues



Possible ACIX deployment, with one global Index Server and a local Index Server for CE 1a and CE 1b

ARC and Rucio

- Rucio was added in 2014 just before Rucio went into production for ATLAS
- Implemented using REST calls with native ARC HTTP client (no dependency on Rucio clients)

```
> arcls -L rucio://rucio-lb-prod.cern.ch/replicas/mc16_13TeV/EVNT.12714678._001433.pool.root.1
        srm://srmv2.ific.uv.es:8443/srm/managerv2?SFN=/lustre/ific.uv.es/grid/atlas/atlasdatadisk/rucio/mc16_13TeV/e5/fd/EVNT.12714678._001433.pool.
root.1
srm://grid002.ft.uam.es:8443/srm/managerv2?SFN=/pnfs/ft.uam.es/data/atlas/atlasdatadisk/rucio/mc16_13TeV/e5/fd/EVNT.12714678._001433.pool.root.1
srm://lapp-se01.in2p3.fr:8446/srm/managerv2?SFN=/dpm/in2p3.fr/home/atlas/atlasdatadisk/rucio/mc16_13TeV/e5/fd/EVNT.12714678._001433.pool.root.1
srm://sdrm.t1.grid.kiae.ru:8443/srm/managerv2?SFN=/t1.grid.kiae.ru/data/atlas/atlasdatadisk/rucio/mc16_13TeV/e5/fd/EVNT.12714678._001433.pool.root.1
srm://dcsrm.usatlas.bnl.gov:8443/srm/managerv2?SFN=/pnfs/usatlas.bnl.gov/BNLT0D1/rucio/mc16_13TeV/e5/fd/EVNT.12714678._001433.pool.root.1
```

ARC Cache Integration with Rucio

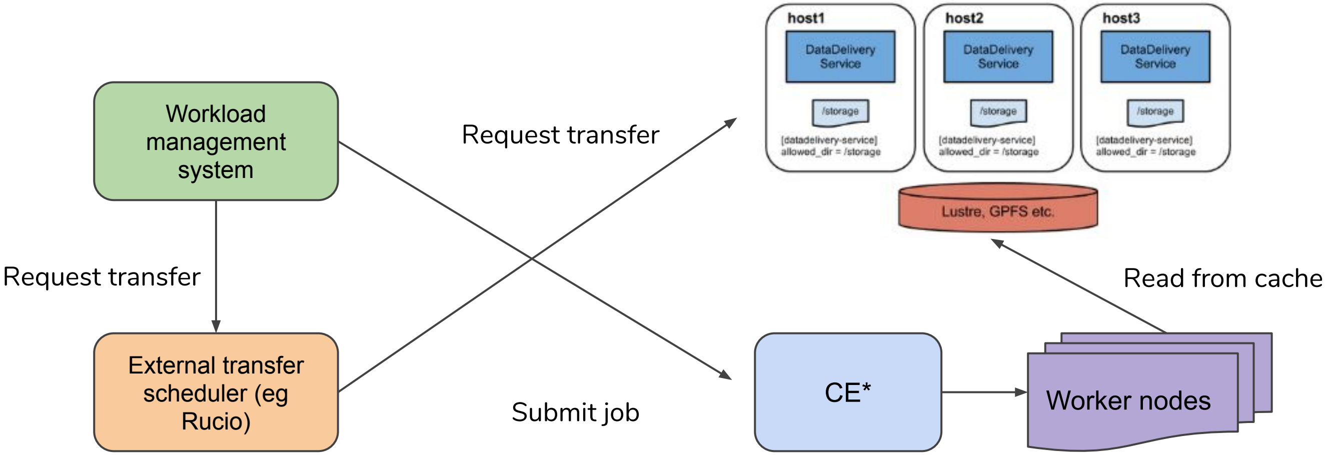
- Cache data can be registered in Rucio on a volatile RSE
  - i.e. Rucio does not manage the data on the RSE but can index it
- Useful for brokering jobs to where data is already cached
  - It is not guaranteed that the data is still in the cache when the job gets there, but not a problem - ARC will download it again
- A probe on the CE (the “whistle-blower”) periodically sends lists of files added to and deleted from the cache

```
# python whistle-blower.py
usage: whistle-blower.py [-h] --cache-dir CACHE_DIR --rse RSE
                         [--broker BROKER] [--port PORT] [--topic TOPIC]
                         [--timeout TIMEOUT] [--chunk-size CHUNK_SIZE]
                         --username USERNAME --password PASSWORD
```

  - ARC CE is configured to periodically dump the cache content to files
  - The whistle blower compares the dumps and looks at the difference
  - ActiveMQ messages with add/delete replica are sent to the Rucio message brokers
- The mechanism is not ARC-specific, can be used for any cache or non-Rucio-managed storage

ARC as a Data Transfer Service

- A stand-alone delivery server could provide a mechanism for pre-placing data in the cache without the need for ARC CE
    - i.e. alternative transfer tool to FTS in Rucio
- Useful in cases like HPC which have performant shared filesystem but no grid storage interface



Workload management system
Request transfer
External transfer scheduler (eg Rucio)
Request transfer
Submit job
host1
DataDelivery Service
/storage
[datadelivery-service] allowed_dir = /storage
host2
DataDelivery Service
/storage
[datadelivery-service] allowed_dir = /storage
host3
DataDelivery Service
/storage
[datadelivery-service] allowed_dir = /storage
Lustre, GPFS etc.
Read from cache
CE*
Worker nodes

* Any kind of gateway for scheduling payloads, vacuum model etc

Conclusions

- ARC’s data staging features have been an essential part of the NDGF model since the beginning
    - Asynchronous data transfers hide the latency of a distributed storage
- ARC CE with data staging is the most common solution for integrating HPC sites in ATLAS
- ARC data staging features can be deployed as an independent service and expose a compatible transfer tool interface to Rucio

- Links:
    - ARC data services technical description: http://www.nordugrid.org/documents/arc6/tech/data/index.html
    - CHEP presentation on ARC efficiency gains: https://indico.cern.ch/event/773049/contributions/3473375/