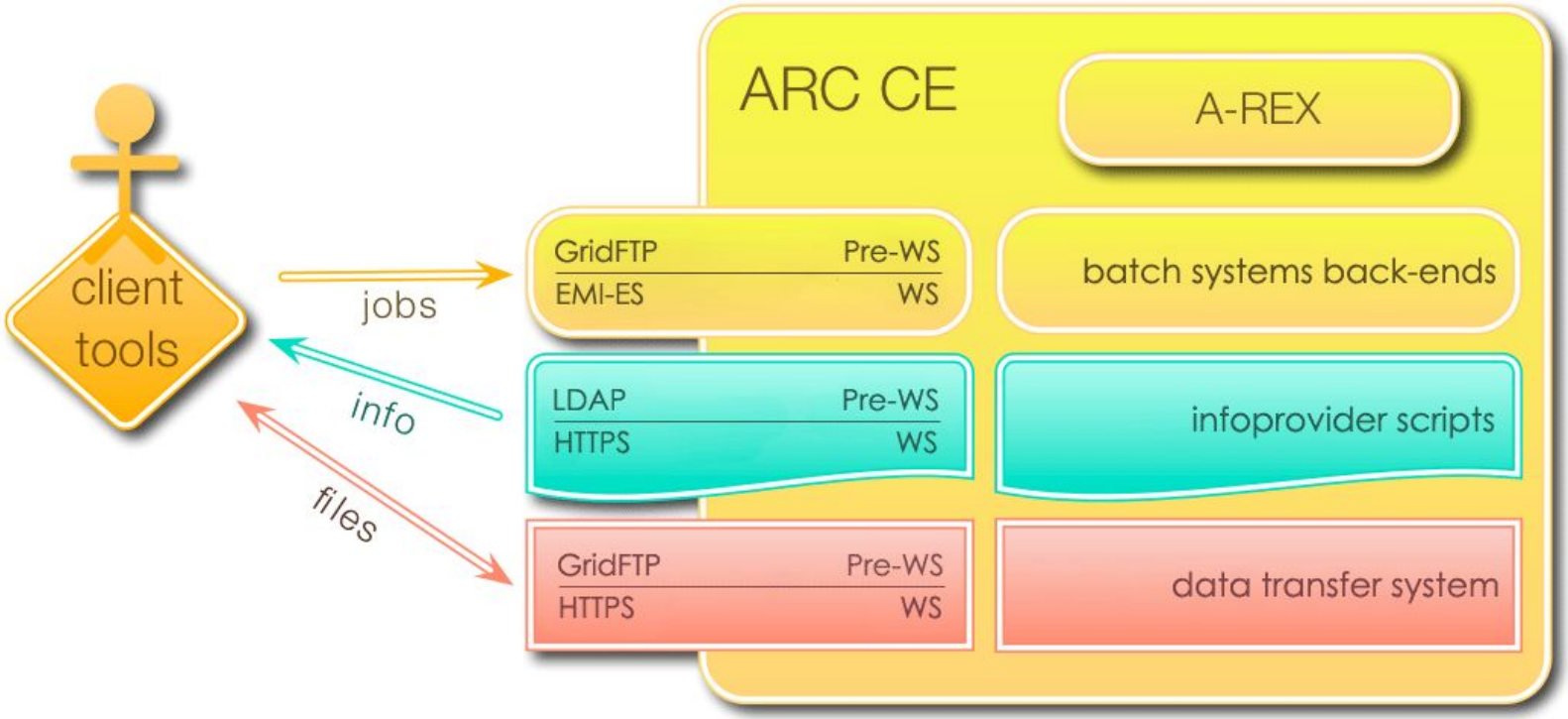
Roadmap towards WM standardization for HL-LHC: ARC perspective

Andrej Filipcic

What is Advanced Resource Connector

- Grid middleware stack developed by Nordugrid collaboration since 2002
  - Open Source, mostly volunteer contributors
- Key components:
  - ARC Compute Element with support for most of the batch system, Boinc, SCEAPI
    - Job control
    - Data transfers and shared cache management
  - Client tools:
    - Jobs submission
    - Proxy management
    - File copy
  - API: python libraries providing interface to full software stack
    - Custom services
    - Custom clients, eg arcControlTower

ARC-CE services and access



Using ARC on different architectures

- Pull mode - standard WLCG way of job submission
- Push mode - submit predefined payload with resource specification and input files to be prestaged
- Caching - using shared posix file system to store and auto-manage inputs
- HPCs - push mode + input caching, no node connectivity (CSCS, SuperMUC, CN HPCs...)
- Boinc - ARC-CE with Boinc service backend (just another batch system)
- Clouds - automatic bootstrapping of batch cluster, cache FS, ARC-CE and supporting services (eg squid, nfs server), tested on Amazon EC2 and academic clouds
- In all use cases, the clients can use ARC-CE transparently - automatic integration in ATLAS production system

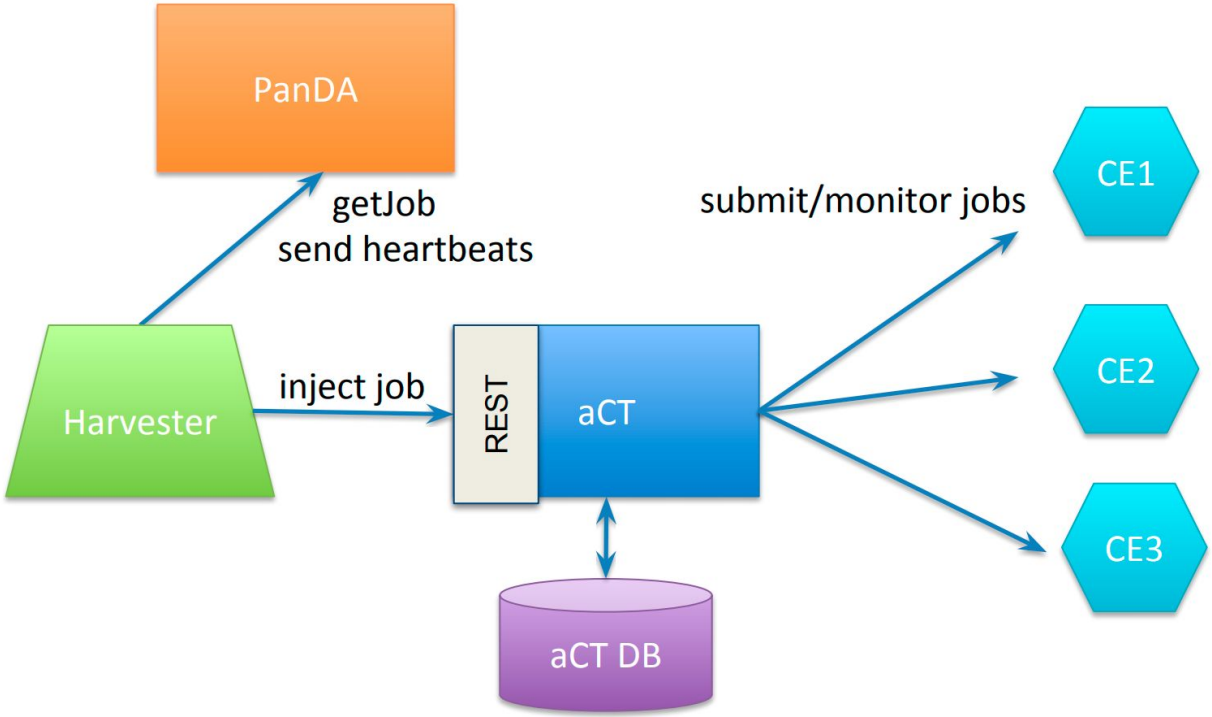
Resource vs Workload management

- ARC-CE
  - interface to the batch system
  - Used by different tools, eg Dirac, Condor-C, ...
- arcControlTower (aCT)
  - Workload management system interfacing to workflow management system (PanDA)
  - Submission to a (large) set of CEs (similar to Condor-C + schedd)
  - Part of harvester effort

ARC-CE - general purpose

aCT - tuned for ATLAS, but can be adapted to any WFMS

Harvester + aCT submission backend

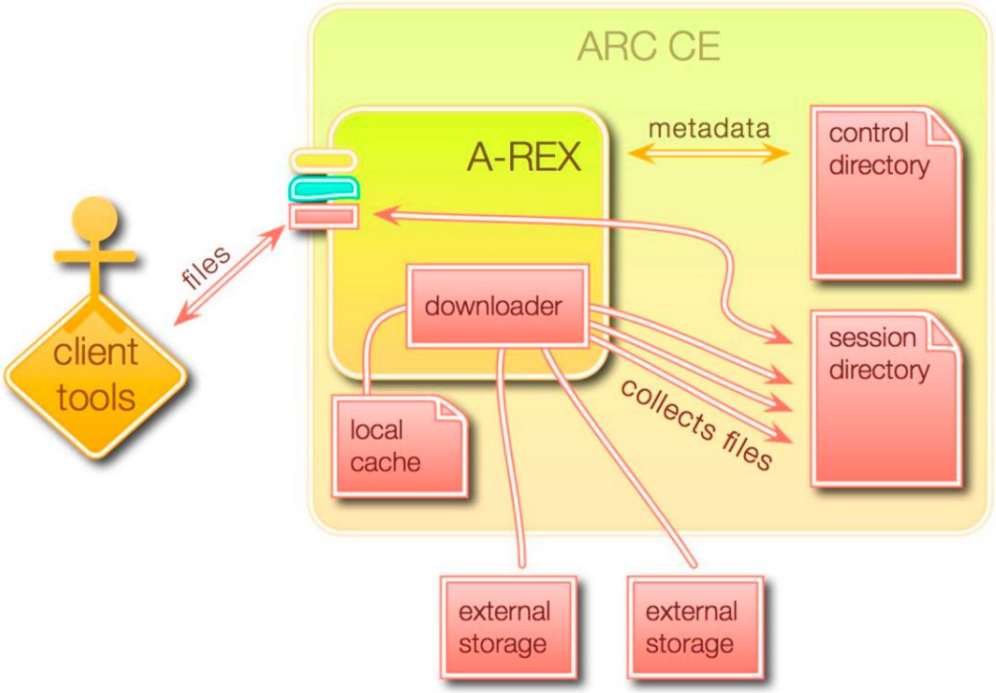


- Scalable to ~100 sites and 100k cores on single server
- Submission to diverse resources:
  - Grid sites
  - HPC with CE
  - ATLAS@ Home
- Direct CE submission, or through schedd

ARC success stories

- Using diverse resource architectures
  - HPCs, ATLAS@Home, Cloud on demand
  - Some ARC sites are fully on IaaS (e.g. in Finland)
- Using ARC clients for massive job submission by individuals, either directly or through:
  - arcrunner - job submission loop scripts
  - arcControlTower - stripped down ATLAS aCT without PanDA interface
- Using national facilities
  - E.g. ARNES HPC does not provide login access, all submission and data handling is done by ARC clients

Using the ARC cache



- Volatile storage of input files
- Up to 1PB cache in production
- Suitable for:
  - Sites with shared filesystems
  - Sites with distributed storage such as NDGF-T1
  - HPCs
  - Lightweight sites without permanent storage
- Can be used as volatile Rucio storage element

ARC Roadmap

- ARC release 6 coming out in few months - major revision
  - Simplifying the configuration
  - REST submission interface
  - Event driven engine
  - ARCHERY - dns based hierarchical endpoint registry, resource discovery
  - Transition to git
- Future development
  - Extending caching to support other technologies, eg xrootd/xcache
  - Native container handling
  - Alternatives to X509 (scitokens prototype in development)
  - aCT: extending to other user communities

What is wrong with WLCG computing in general?

- Custom WLCG stack
- x509 authentication
- Tuned for pull-model workflow and uniform resource requirements
- Tuned for serial workloads
- Complex clients, for big communities and custom workflow management systems
- Difficulties if the computing resources deviate from WLCG-standard architecture, eg
  - HPCs with limited connectivity and no node-local disk
  - Complex resource requirements (40GB mem, 8 cores, GPU, 250GB scratch disk, 1.5Gb/s LAN bandwidth...)

Points to consider

- Information system
  - bdii going away
  - But we need to know site capabilities, both static (eg no of cores), and dynamic (site is full with high-memory jobs, how many 1GB mem job slots we can backfill?)
  - Infosys should be flexible enough to provide additional information (plugins, eg as done for json storage reporting)
  - pragmatic approach rather than long standardization process - evaluate what information is needed and try to implement it in a simple but extensible way
- Authorization:
  - X509 not enough any more - difficult for users and not supported by popular services
  - oauth2, eduGain, other technologies - common development/effort for WLCG middleware stack

Advanced usage of resources

- As opposed to classic wlcg workflow
  - Allocate core, grab a job from central service, stagein inputs, run payload, stageout outputs
- More complex scenario:
  - Push input dataset to a site
  - Allocate large number of cores
  - Process data on dynamic/available amount of resources
  - Access/stream remote inputs if not locally available (eg event streaming)
  - Stream the outputs as produced to a non-volatile storage
  - Optimize the workflow mixture for a given site
  - Act progressively if bottlenecks appear, eg
    - Reduce cores if I/O limits affect the efficiency
    - Reduce heavy I/O payloads if disk, LAN or WAN is saturated

cntd.

- There are no common tools to handle such complex workflows
  - Batch systems (SLURM) are only recently introducing dynamic resource shaping during the job execution (eg reducing used cores for the job last step...)
- Experiments are testing and implementing their own solutions
  - Job monitoring: memory usage, disk I/O metrics, network merics
  - Whole-node job slot micro-management
  - Input data caching
- Time to consider
  - Common tools, eg Harvester
  - Common extensions can be added on top of existing middleware
  - Middleware API can be used to build additional services needed by experiments

Resource diversity

- Not only HPC!
- WLCG pledged sites differ a lot:
  - Different storage technologies
  - Caches, local disks, shared filesystems
  - Fat nodes vs slim nodes
  - Local scratch SSD vs HDD
  - 10Gb/s v 1Gb/s NICs (and others...)
- Different sites perform differently for specific workloads
- These days we tune it manually according to site performance - still manageable with long campaigns
  - eg , Tier-0, HLT farm, sites with low WAN, ...
- A systematic approach to describe and collect information and metrics is needed

Workflows

- Necessary to improve the event processing throughput and optimize for given resources
  - Work for both software and computing, eg computing-aware software
  - Focus on (sub)event-based processing on datasets
- Overall guideline - process as many events as possible balanced per processing step on a given set of resources
  - cpu efficiency is not the primary measure, but it’s stupid not to use all the cpu power, eg run low I/O payload to leverage the idle cycles

Summary

- ARC provides solid service and API for current usage and building blocks for future development of custom services and tools
- Way of using the resources for HL-LHC needs to change a lot, a simple scaling will not work efficiently
- Common approach to design and architect components for the future computing is mandatory, we need some sort of (loose-enough) standardization