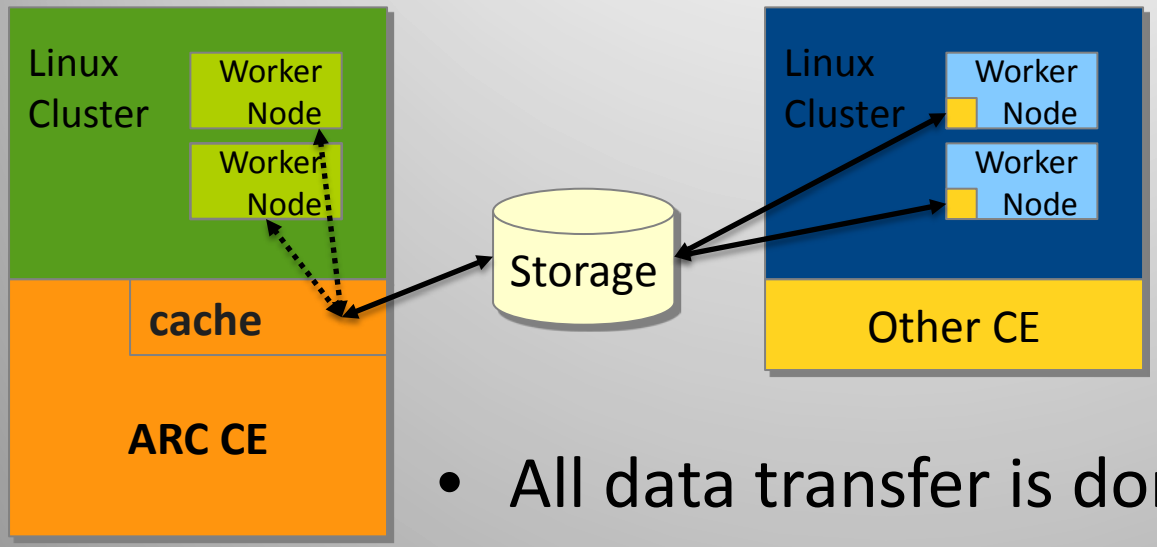
ARC CE

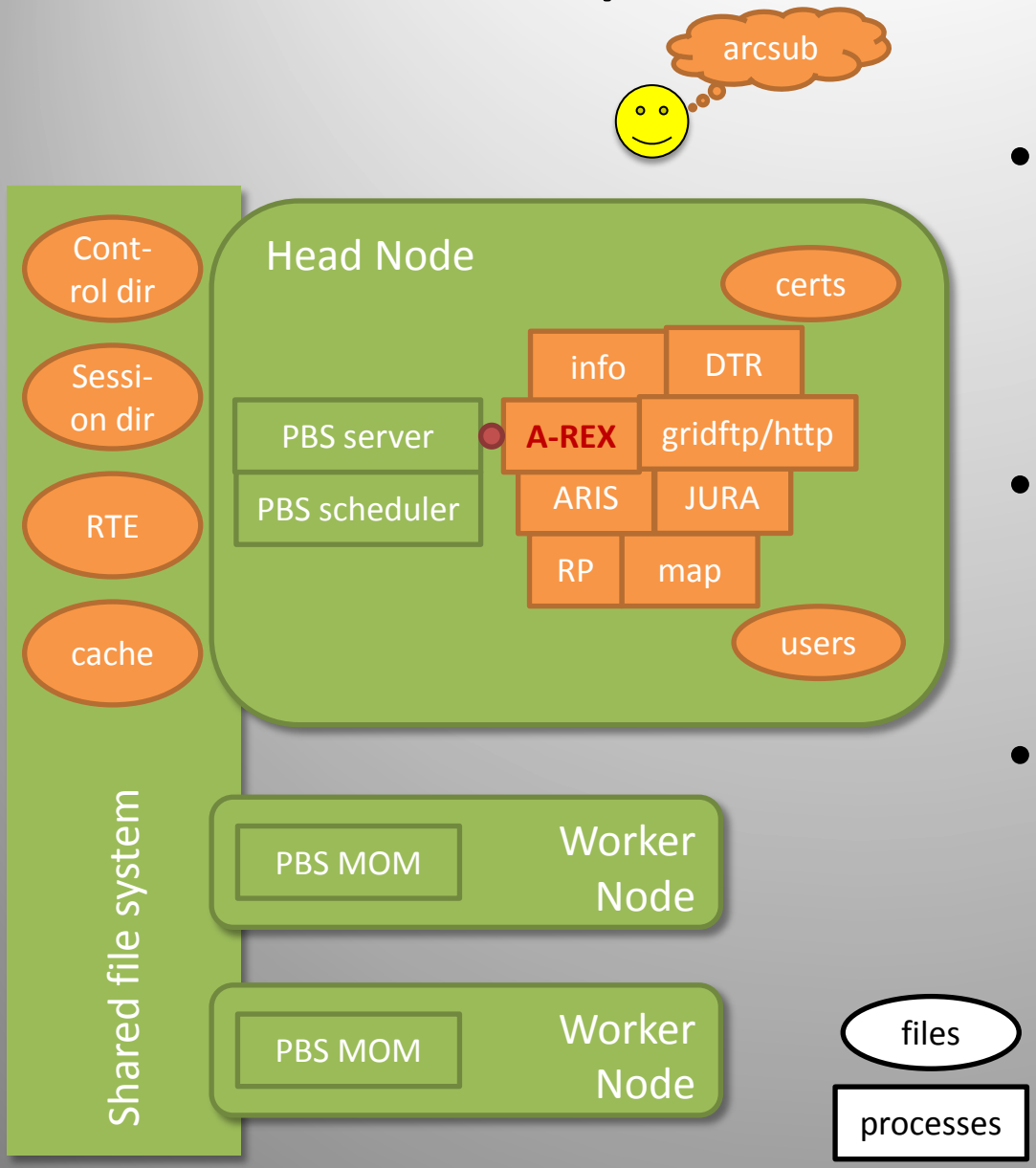
Oxana Smirnova

ARC CE key concept: optimized for data-intensive jobs



- All data transfer is done by the CE itself
  - Allows to cache frequently used files
  - Minimizes bandwidth
  - Maximizes WN usage efficiency
  - ARC CE itself is a very complex service
  - Requires high-end storage for cache

ARC CE components on a (PBS) cluster



- Interfaced to a number of batch systems
  - SLURM, SGE, PBS, LSF, LL, Condor
- ARC CE is a uniform interface: batch system specifics are not exposed
- No ARC component is installed on worker nodes
  - No need, because the CE handles transfers

Step 3: formulate job description

- 37 attributes to find best matching cluster
- Job description language: whatever users like, internally ADL-like

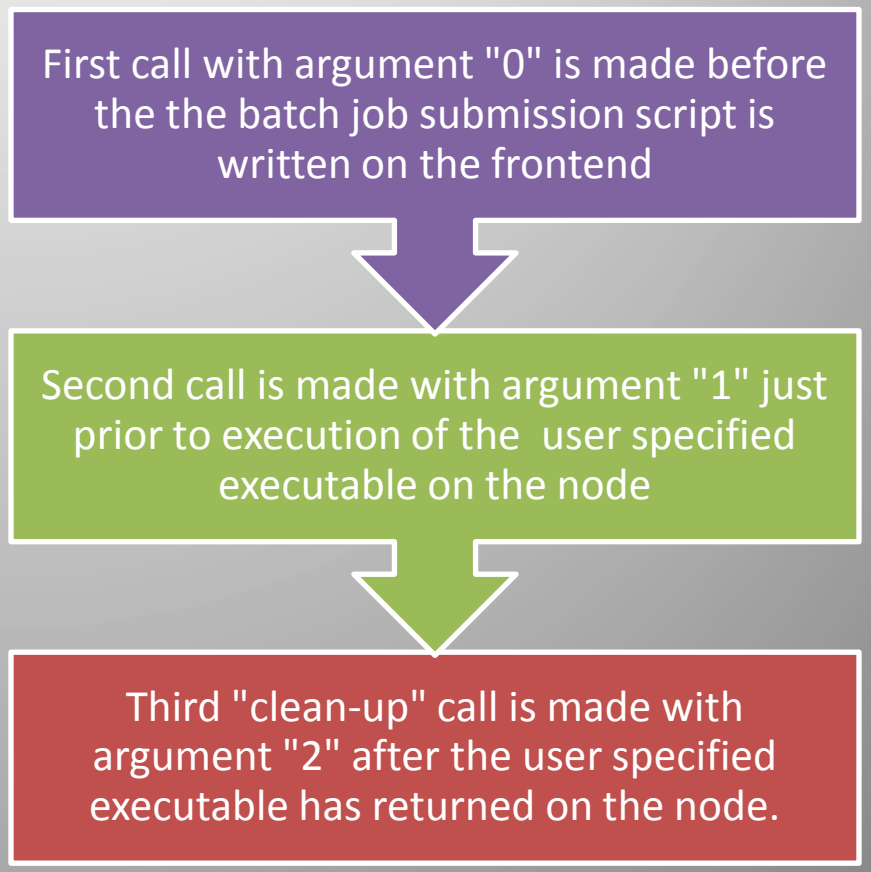
Job attribute	Example
Application environments	/APPS/HEP/ATLAS-20.1.0.1 /ENV/GLITE /ENV/FULLNODE
Main executable (binary or script)	Findhiggs.py
Arguments of executable	-i input.root -o output.root
Input files	srm://srm.infn.it/atlas/2012/file1.root
Output files	srm://srm.ndgf.org/atlas/ulf/higgs.root
Number of slots per job*	36
Queue name (**bad** practice)	mpi_jobs
Time (or benchmark), memory, disk	*numbers (or benchmark name, e.g. HS06)*
Standard input/output/error	stdout.txt
and many others	

** Interpretation of "slot" depends on batch system !*

Step 3a: understand Runtime Environment (RTE) concept

- Application environment is formalised as “Runtime Environment” (similar concepts exist in other CEs)
- Runtime Environment can encapsulate not just application software, but also:
  - Batch system peculiarities
  - Hardware aspects
  - Can even emulate gLite WN
- It is just a shell script
  - For many RTEs, namespaces are handy

Requested RTE script is called by A-REX 3 times:



Example: how to make jobs to use 4 cores per node (for PBS)

Good way: RTE script

```
user@host# cat RESERVE_4_CORES
#!/bin/sh
case "$1" in
0)
      export joboption_nodeproperty_0="ppn=4"
      ;;
1)
      # Nothing to do
      ;;
2)
      # Nothing to do
      ;;
*)
      return 1;;
esac
```

- Add line to job description file: (runtimeenvironment= "RESERVE_4_CORES")
- As many RTE scripts as needed

Bad way: special queue

```
user@host# cat arc.conf
.....
[queue/mpi_jobs]
.....
queue_node_string="ppn=4"
```

- Add line to job description file: (queue=mpi_jobs)
- And so on, introduce a new queue for every imaginable configuration

Sysadmins' opinion of RTEs

- In practice is the only way to make parallel/multi-core applications work in heterogeneous clusters
- No need to transfer executable, loader or libraries
- Possibility to build clusters which allow execution of specified applications only
- Better application performance with architecture specific optimizations
- Initialization of environment variables and paths, i.e. providing standard environment for executables submitted by user
- Version management

- Logistical problem:
  - Who/how keeps track of all RTEs?
    - Currently, just a Web page

Multi-core jobs in ARC outlook

- Internal capability to support multi-core/whole-nodes exists (RTE)
  - External interfaces used in WLCG production are still pre-EMI
- EMI-ES is the future interface
  - Service information must expose EMI-ES capabilities (GLUE2)
  - Job description must match EMI-ES capabilities (ADL)
  - All EMI CE flavors are expected to implement EMI-ES, GLUE2 and ADL
- ADL is reach enough and includes attributes sufficient to request whole-node/multi-core configurations
  - ARC is yet to implement ADL server-side, expected this fall
    - Priority: SLURM, SGE, PBS, LSF
  - ADL is not really user-friendly, good for internal communication. User-end languages (XRSL, JDL) may need to be extended to map to ADL fully
- Accounting for multi-core jobs: basic attributes exist in EMI CAR, more may be needed