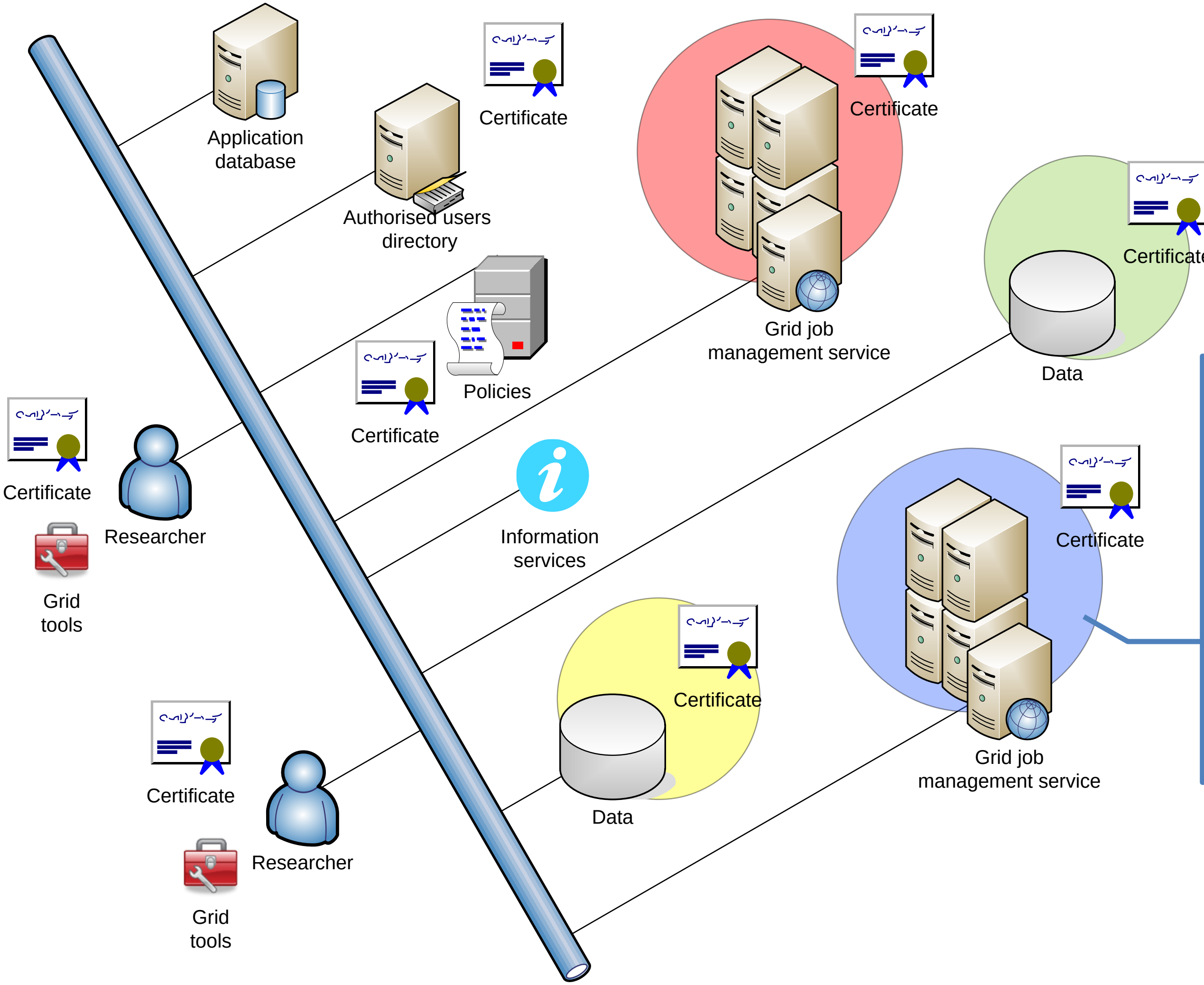
Computing services

Overview of generic Grid components



Computing service deals with:

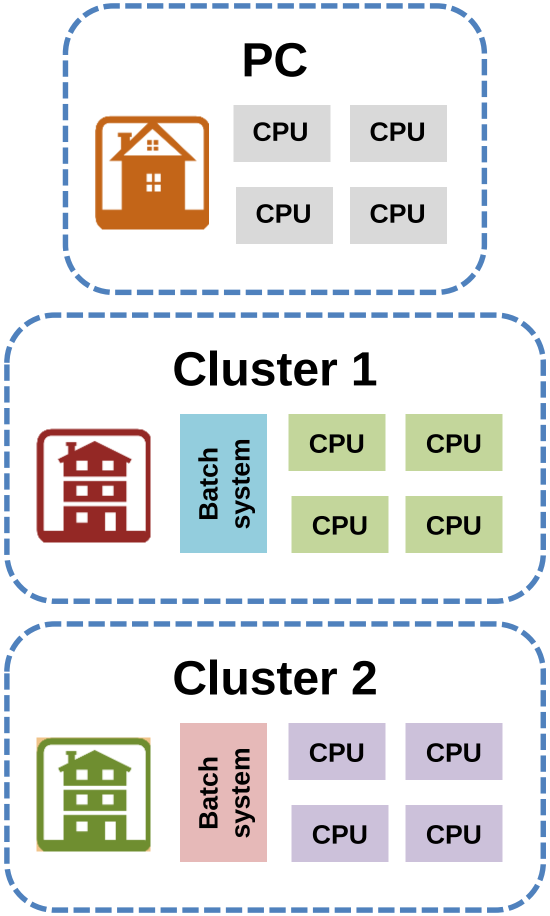
- **Access control**
- **Job management**
- **Data input/output**
- **Accounting**
- **Information**

The core of the Grid

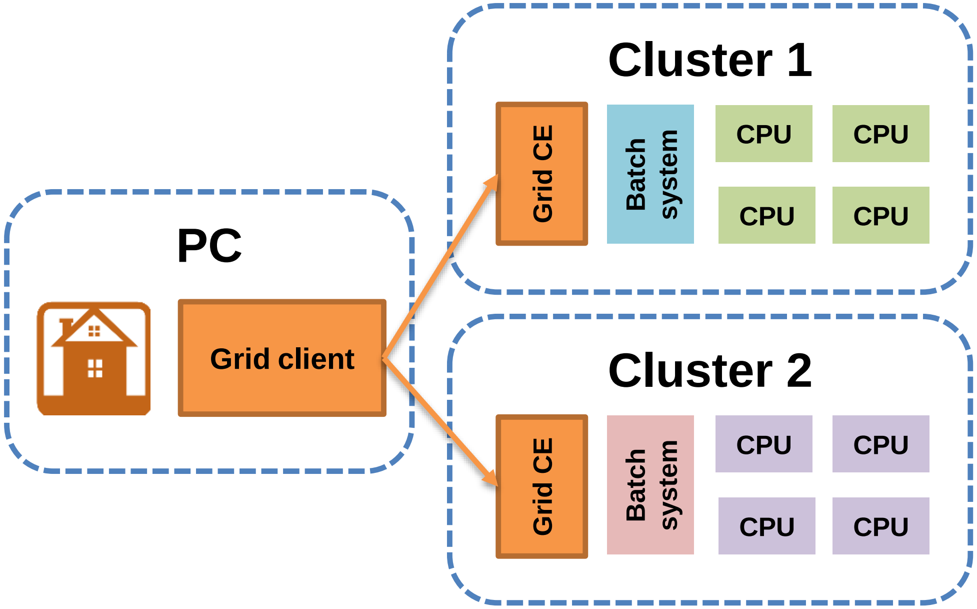
- Grid is a distributed **computing** technology
  - It is particularly useful when data is distributed
- The main goal of Grid is to provide common layer on top of different computing resources
  - Common authorization, single sign-on
  - Common task specification (job description)
  - Common protocols and interfaces for job management
  - Common accounting and monitoring
- All this is provided by Grid Computing Services
  - A single instance of such service is called a **Computing Element (CE)**
  - You also need a Grid client to communicate to CEs

Grid as abstraction layer for computing

PC/cluster world


PC
CPU CPU
CPU CPU
Cluster 1
Batch system
CPU CPU
CPU CPU
Cluster 2
Batch system
CPU CPU
CPU CPU

Grid world


PC
Grid client
Cluster 1
Grid CE
Batch system
CPU CPU
CPU CPU
Cluster 2
Grid CE
Batch system
CPU CPU
CPU CPU

- Grid software is called **middleware**
  - Layer between system and application software

Workflow: Grid vs PC/cluster

PC/cluster

Log in via SSH	• Different logins on different machines
Familiarize with the environment	• OS, installed software, storage space, batch system, sysadmin etc
Customize the environment	• Pathes, environment variables, own software, scripts, data files etc
Prepare for batch submission	• Interactive execution of short jobs, optimization of batch scripts
Submit jobs to the batch system, check their status	• Different batch systems (or none) on different machines
Log out	
Log in later to fetch the output	

Grid

Create proxy	• One for all machines
Create a Grid job description document	• Generalization of batch scripts, plus input/output data location etc
Test a couple of jobs, fix job description	
Submit jobs to the Grid, check their status	• Same commands for all machines
Watch output appearing in the desired location	• Or fetch it manually

Key differences

Operation	PC/Cluster	Grid
Log in	Interactive SSH session	No actual log in
	Different passwords	Single sign-on
Job description	Shell script with batch-system-specific variables	Specialized language
	Different scripts for different batch systems	Same document for all systems
Environment	Can be personalized	Pre-defined, generic
	Can be explored in detail	All details can not be known
Job monitoring and management	Requires log in	Remote
Data management	Manual	Can be automatic

Tasks of a Grid client

Security

- Create proxy certificates
  - We discussed it earlier

Information

- Discover Grid resources
  - Will be addressed in the next-to-next lesson

Computing

- Interpret job description and submit it to a matching resource
  - **Topic of today**

Data handling

- Copy files to/from the Grid
  - To be addressed in the next lesson

Grid job description

- For the purposes of this course, Grid job description is a document prepared by the user
  - This document can be modified by Grid tools
    - » Job description received by a cluster may be quite different from that prepared by the user
- Job description has a twofold purpose:
  - Specify the workflow
    - » Executable, input/output files, notifications etc
  - Express job requirements such that a matching resource can be found
- Job description can express requirements as a range, or as a condition
  - E.g., at least 1 GB of memory, or use different input if there is little disk space
    - » Description received by batch systems must be deterministic, no ambiguities
- Challenge: encapsulate features of all batch systems, while adding ranges and conditions

Main attributes of job description

Job attribute	Example
Main executable (binary or script)	MyAnalysis.py
Arguments of the executable	-i input.dat -o output.dat
Input files	https://store.lu.se/physlab/2012/file1.dat
Output files	https://store.lu.se/physlab/2014/file1.dat
Standard input/output/error	stdout.txt
Time, memory, disk	numbers
Job name	My data analysis
Number of slots per job	36
and many others: ARC job description language has 37 attributes	

ARC job description language: xRSL

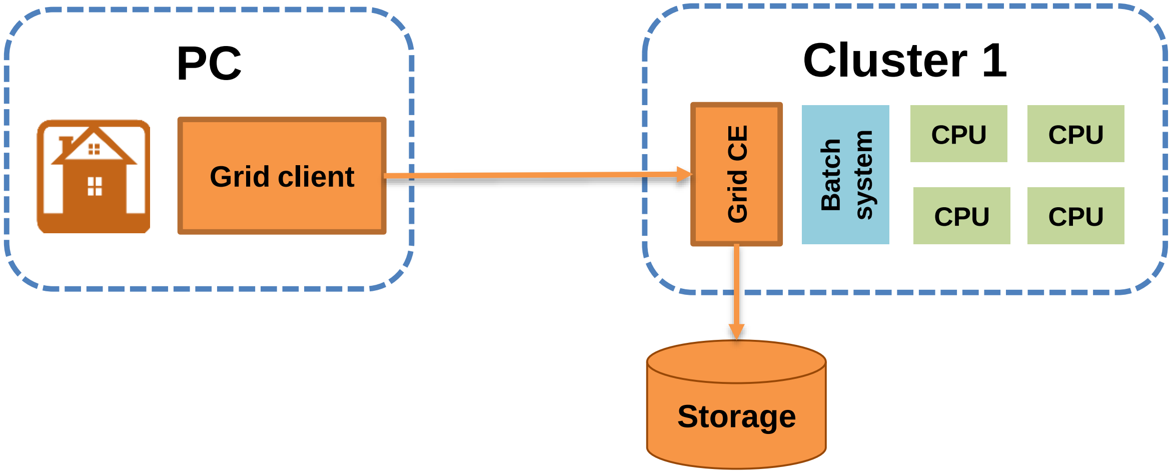
- Based on Globus’ RSL
  - Resource Specification Language
  - In turn is based on LDAP
- The xRSL document is a string, being a concatenation of several attribute assignment statements
- One document can contain several job descriptions
  - Useful for speeding up submission
- Google for “arc xrsl manual” or get the manual here: http://www.nordugrid.org/documents/xrsl.pdf

myjob.xrsl

```
&
(jobname=“My data analysis”)
(executable=“MyAnalysis.py")
(arguments=“-i input.dat -o output.dat”)
(inputfiles=
 (“input.dat” “https://store.lu.se/physlab/2012/file1.dat”)
)
(outputfiles=
 (“output.dat” “https://store.lu.se/physlab/2014/file1.dat”)
)
(stdout=“stdout.txt”)
(join=“yes”)
(count=“36”)
(cputime=“2 hours”)
(memory=“2000”)
(disk=“500”)
(gmlog=“gmlog”)
```

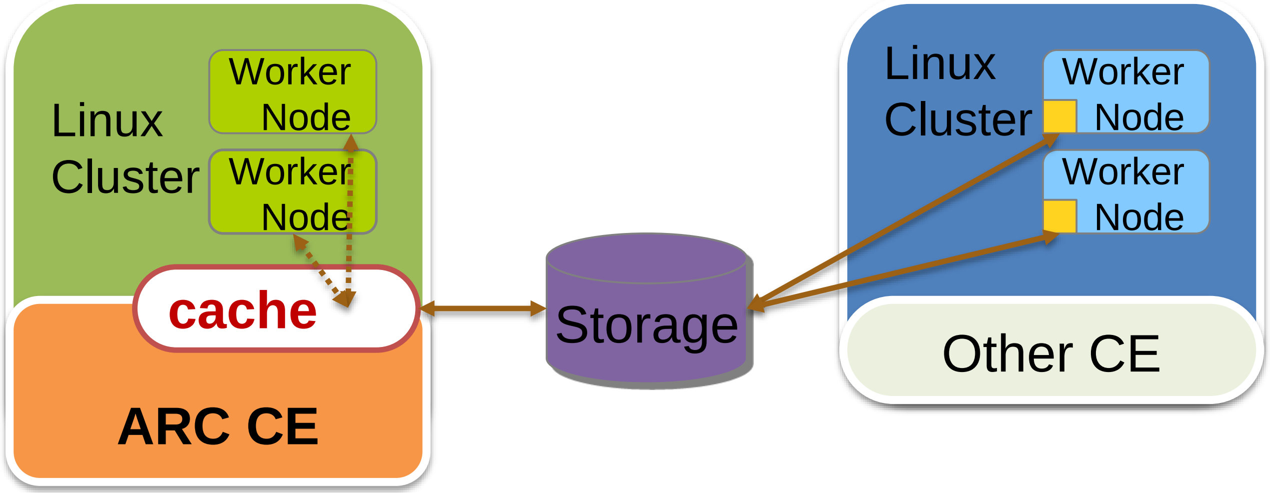
Simplest Grid job submission

Reducing Grid to one cluster, for illustration



- Your Grid client should:
  - Create a proxy:
    - » **`arcproxy`**
  - Submit the job description document to the cluster:
    - » **`arcsub -c cluster1.lu.se myjob.xrsl`**
    - » **`arcsub`** will refuse submission if the cluster does not meet job requirements
- The CE on the cluster should:
  - Check whether you are authorised
  - Fetch input file
  - Convert job description to a batch script and start a batch job
  - Upload output file

ARC CE is optimized for data-intensive jobs



ARC CE can do all the data transfer

- Allows to cache frequently used files
- Minimizes bandwidth
- Maximizes worker nodes usage efficiency

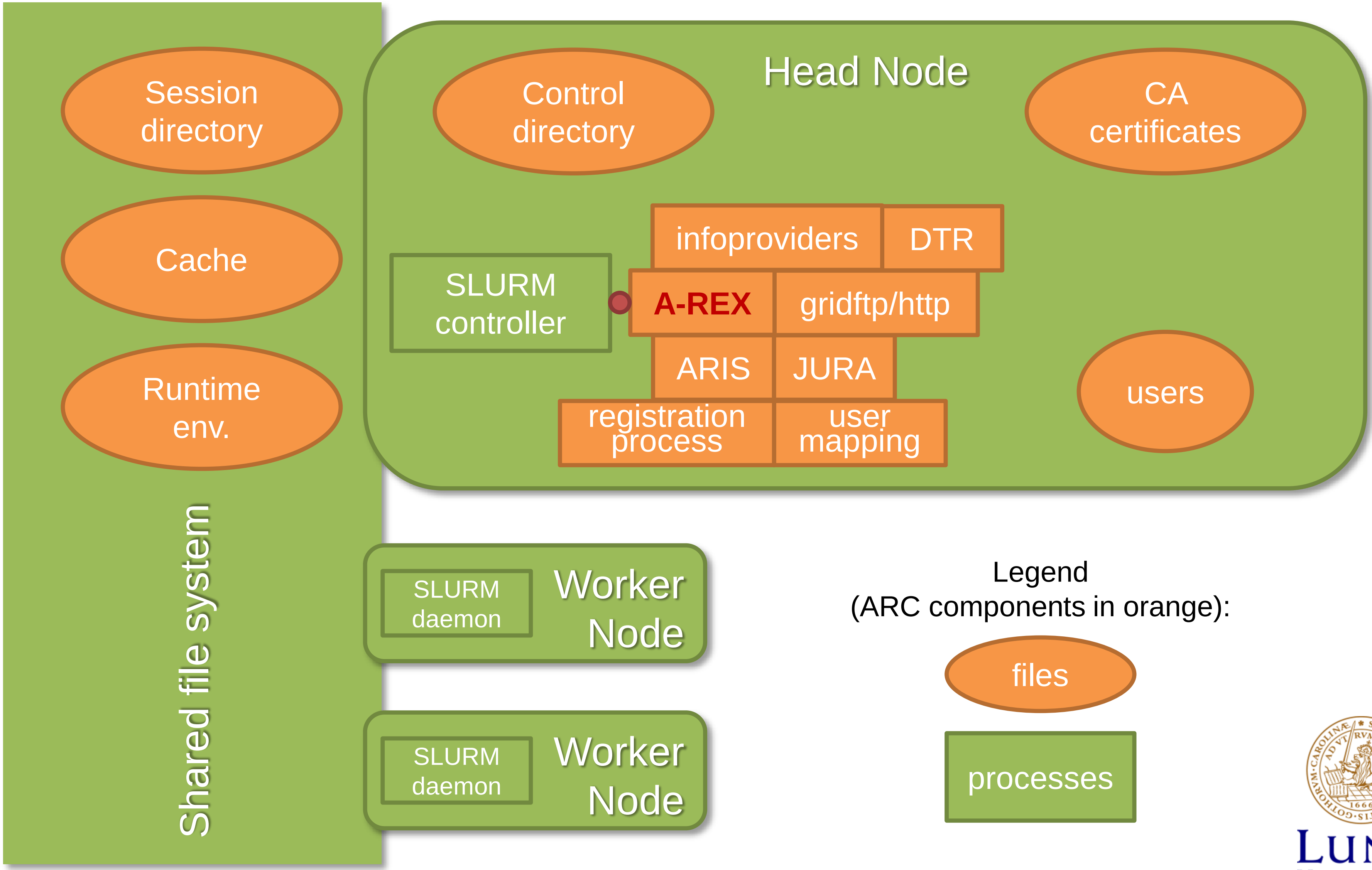
ARC CE is a very complex service

- Is built of many individual services and tools
- Requires high-end storage for cache

- There are few other CEs on the market
  - ARC CE's share is ~10% and growing
  - In Nordic countries, only ARC CE is used

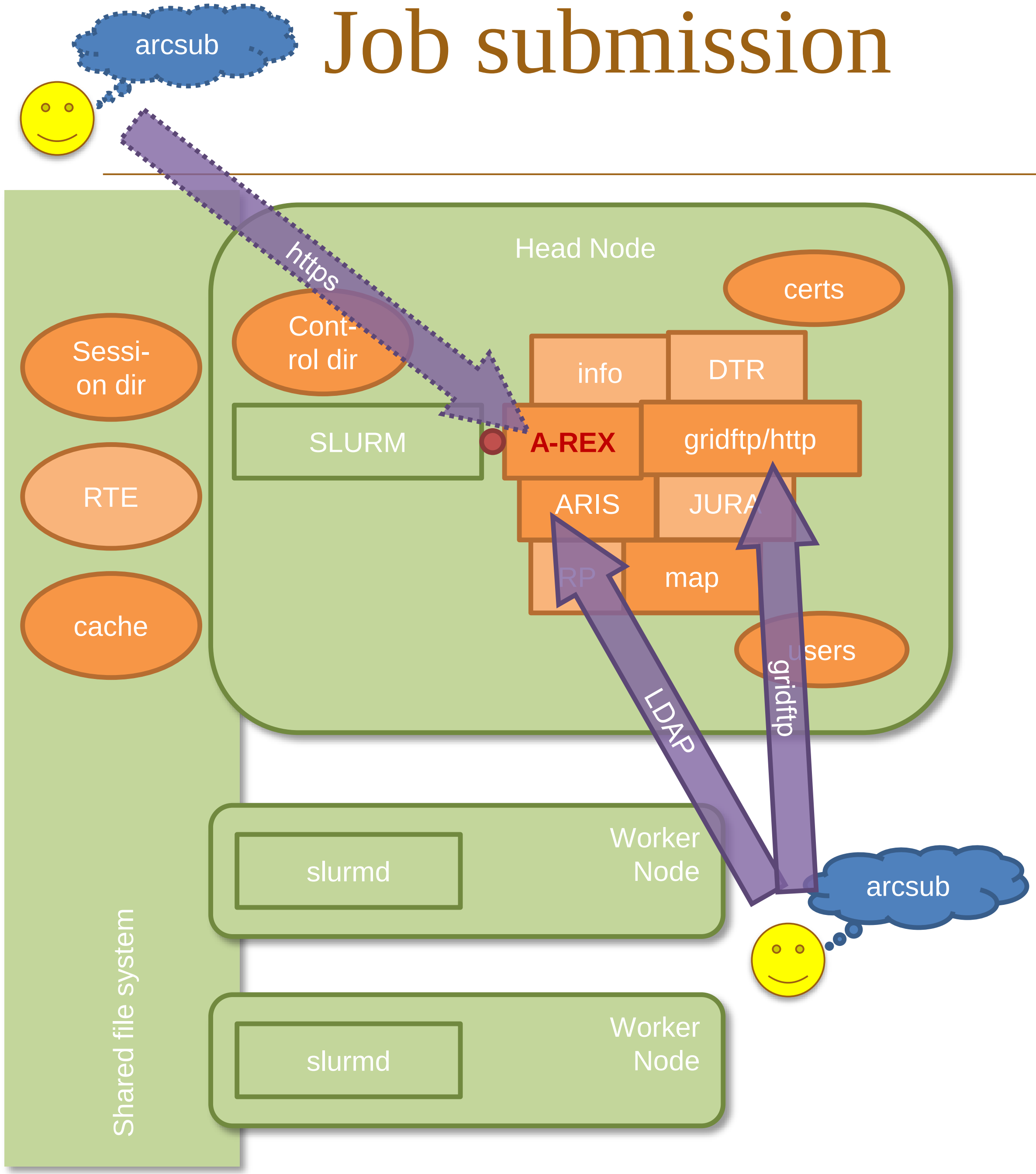
LUND UNIVERSITY

ARC CE components on a cluster



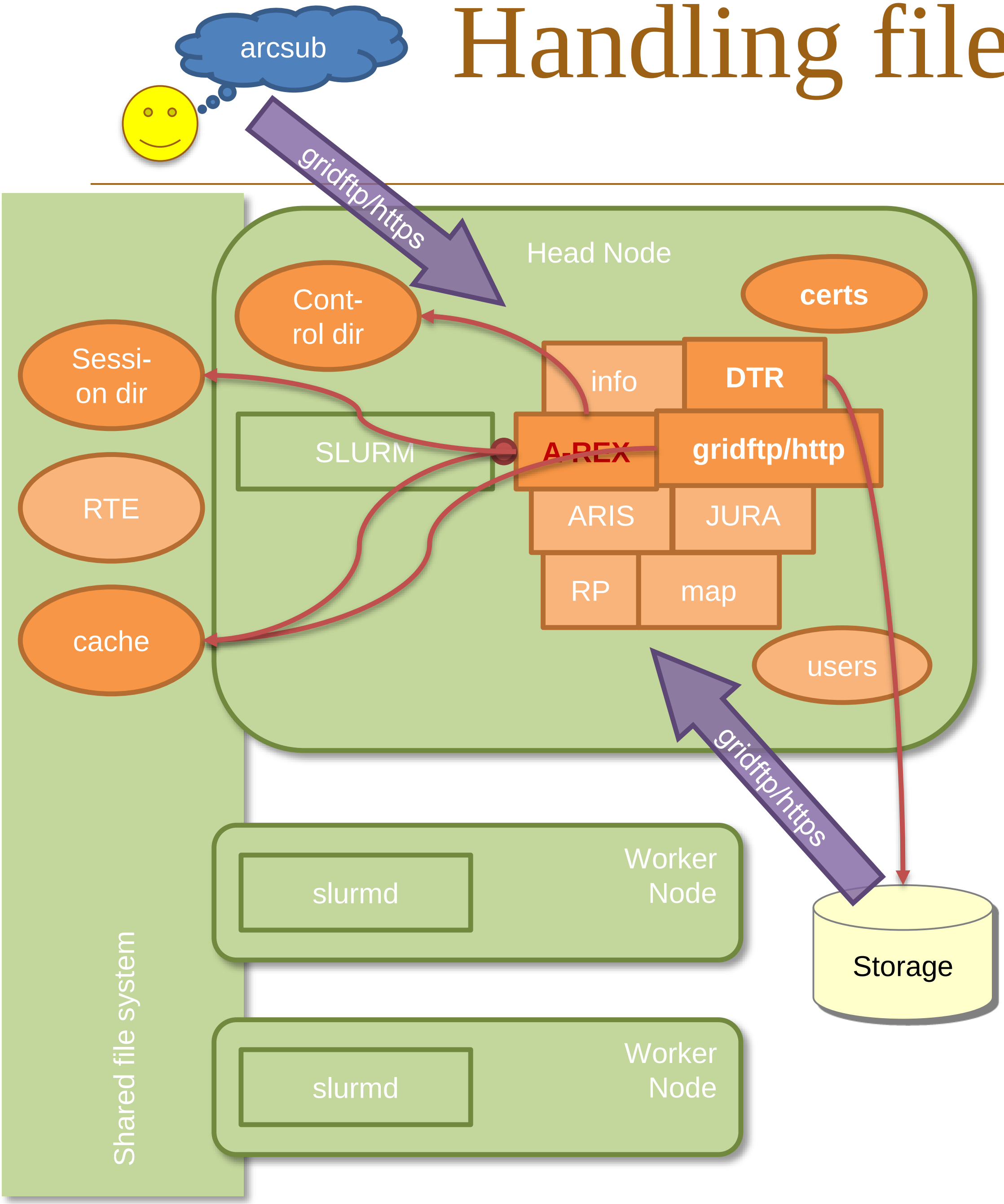
Legend
(ARC components in orange):

Job submission



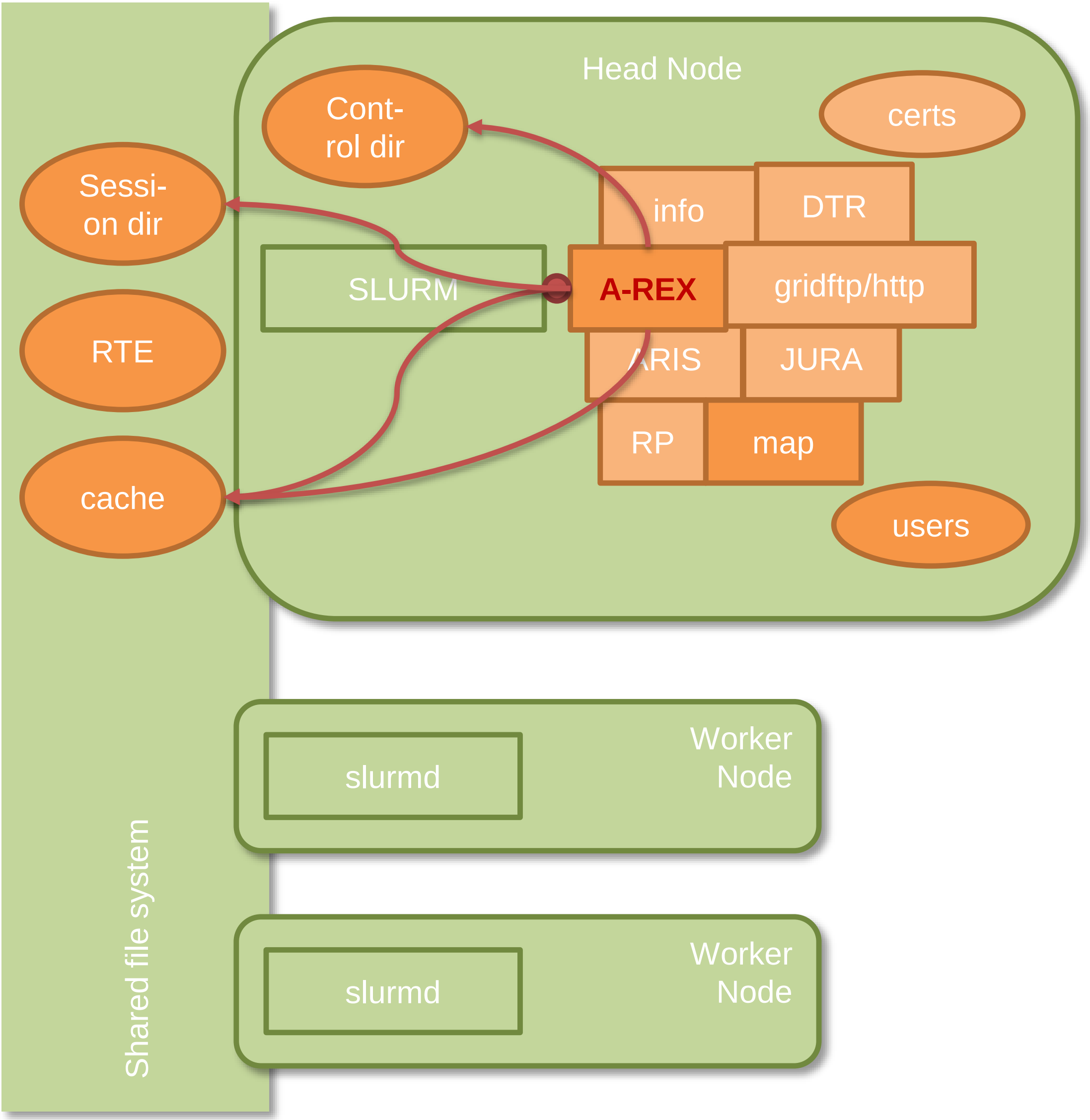
- **A-REX** is the central component
  - Orchestrates other components and communicates to the batch system
- A-REX discovers uploaded job files and launches job processing
- Information and upload can use different protocols
- All steps require **authorisation**

Handling file transfers



- Jobs won't start before all input files are present
- Input files provided by the user are uploaded by the client tool
  - normally, cached
- External files are downloaded by **DTR** when triggered by A-REX
  - also cached by default
- All inputs are copied or linked to the session directory
- Output files are uploaded by DTR to external storage if requested

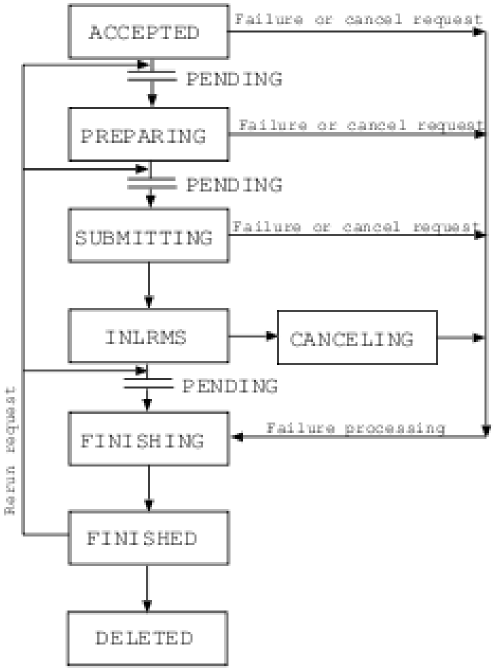
Job submission to the batch queue



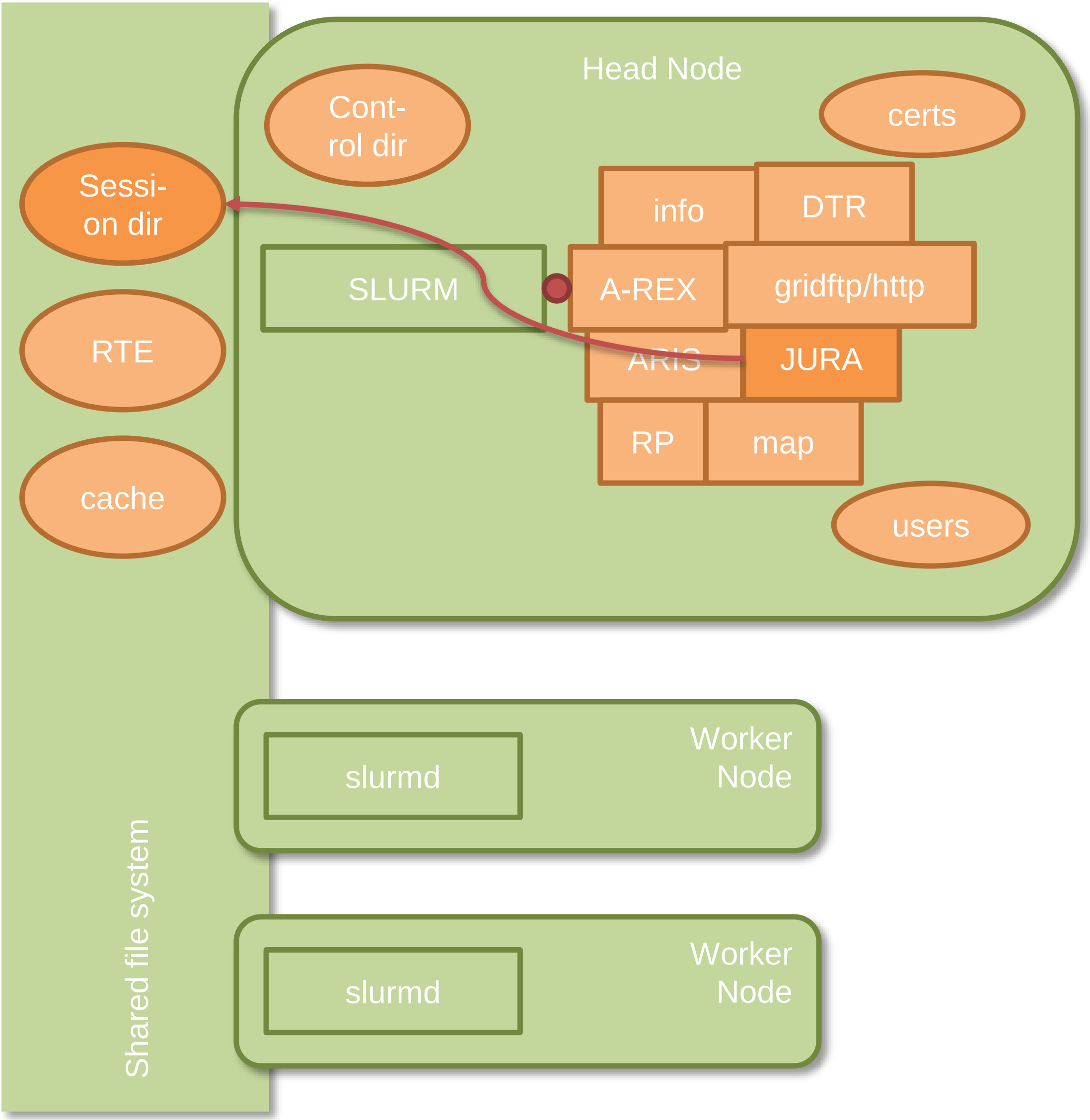
- Key component: batch “back-ends”
  - Encapsulate specific properties of different batch systems and map them to generic functionalities
- A-REX handles the job life cycle
  - Sends them to the batch queue via back-ends
  - Monitors status
  - Triggers data movement
  - Authorisation

Job handling by A-REX

Job state	Description
Accepted	the job has been submitted to the CE but hasn’t been processed yet
Preparing	input data are being gathered
Submitting	job is being submitted to the LRMS
Executing (InLRMS)	job is queued or being executed in the LRMS
Killing (Canceling)	job is being canceled
Finishing	output data are being processed (even if there was a failure)
Finished	job is in this state when either it finished successfully or there was an error during one of the earlier steps
Deleted	after specified amount of days the job gets deleted and only minimal information is kept about it



Accounting



- JURA harvests job information and submits it to an external accounting service
  - For completed jobs only
- Compatible accounting services:
  - SGAS (Swedish Grid Accounting Service), developed in Umeå
  - APEL (Accounting Processor for Event Logs), developed in the UK

Exercises

- **Prepare job description** for the “hello grid” task:
  - Use e.g. **`gedit`** to create a text file **`hello_grid.xrsl`**
  - Use at least the following XRSL attributes**: `executable, arguments, jobname`**
- **Submit your first grid job** by explicitly requesting a cluster:
  - prerequisite: make sure you have a valid proxy:

    ```
    arcproxy -S nordugrid.org:/nordugrid.org/tutorial/Role=student
    ```

  - use one of the two clusters:

    `arc-iridium.lunarc.lu.se` or **`alarik-grid.lunarc.lu.se`**

  - Use the **`arcsub`** command with direct cluster selection:

    ```
    arcsub -c cluster_name hello_grid.xrsl
    ```

  - Inspect the returned jobid, try to check the “session directory” content:

    ```
    arcls <jobid>
    arccat <jobid>
    ```

Exercises

- **Submit and manipulate more complex job** that calculates prime numbers
  - Investigate the `prime_calc.xrsl`, try to understand the stage-in, stage-out phase,
  - Launch several job instances with different job names (hint: change the `jobname` attribute)

    ```
    arcsub –c cluster_name prime_calc.xrsl
    ```

  - check the status of your jobs:

    ```
    arcstat -a
    ```

  - Terminate some of them and check the status afterwards:

    ```
    arckill –k <jobid> ; arcstat <jobid>
    ```

- **Retrieve task result** (download job output), check the `arcget` manual for the options used below:

    ```
    arcget -J –k <jobid>
    ```

  - Inspect the content of the downloaded gridlog directory (`gmlog` subdirectory)