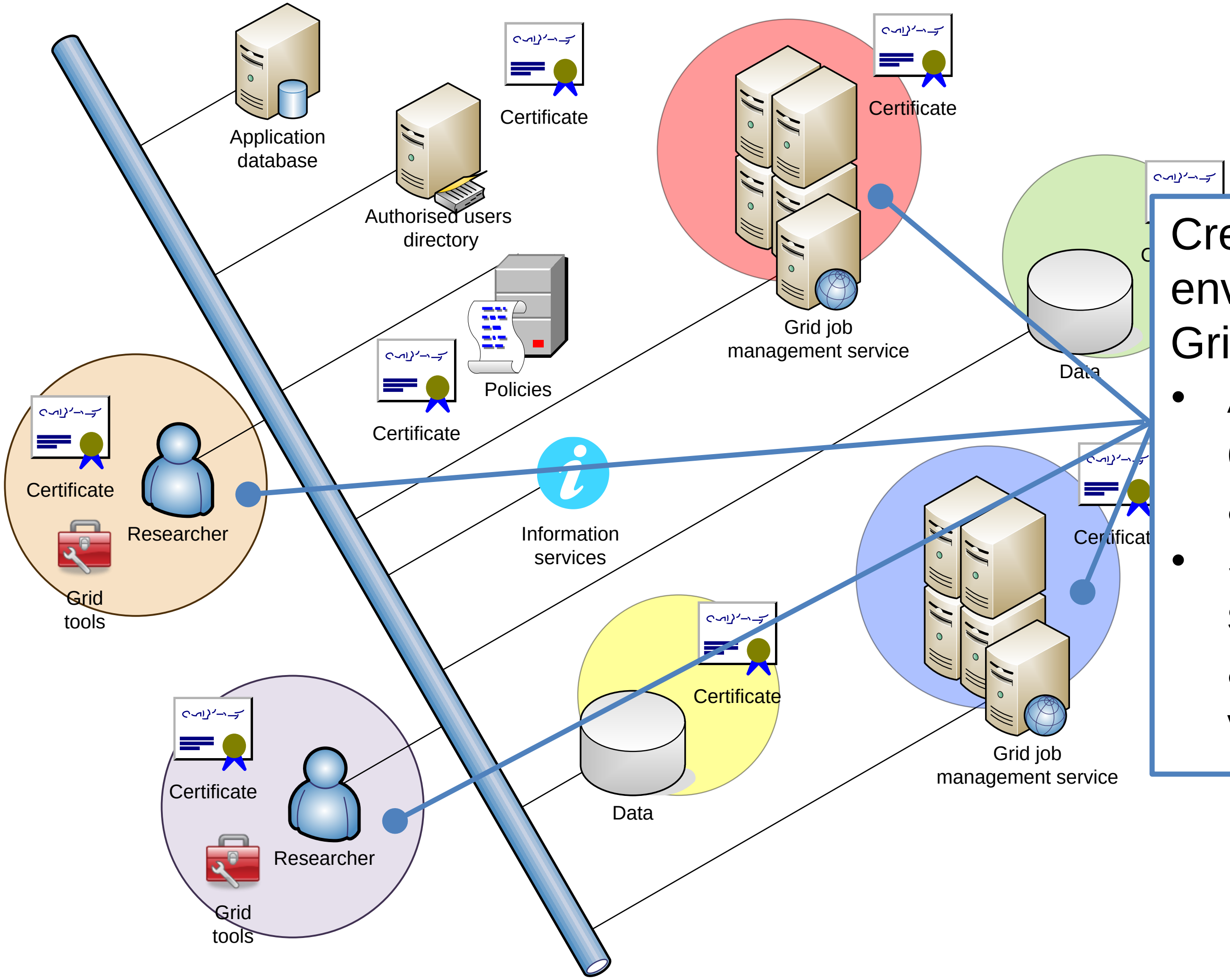
Runtime environments

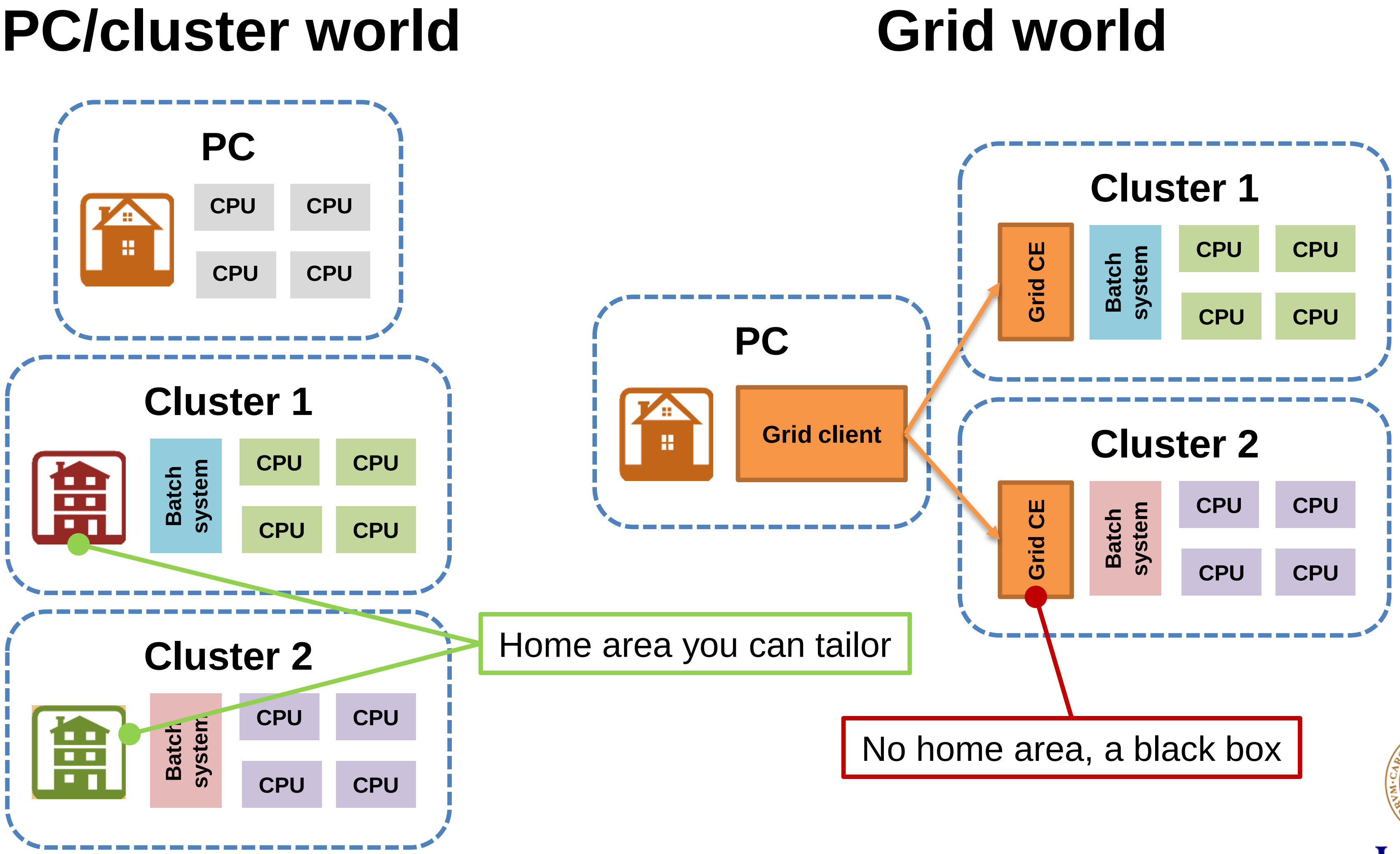
There are no two identical computers



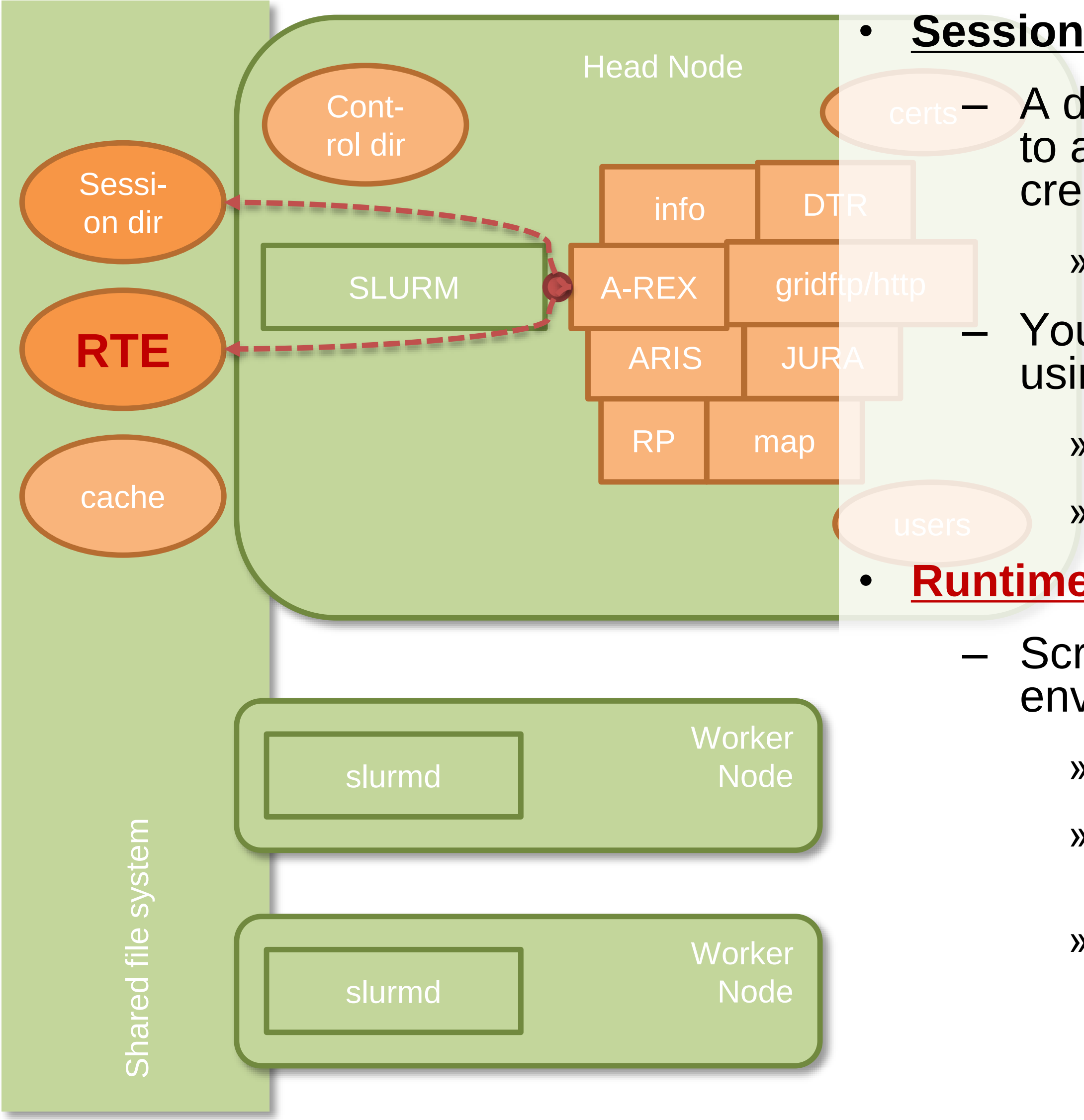
Creating uniform environment across the Grid is a major challenge

- All computers are different, software- and hardware-wise
- Some reasonable similarity can be achieved without virtualization

Recall: out on the Grid, you have no home

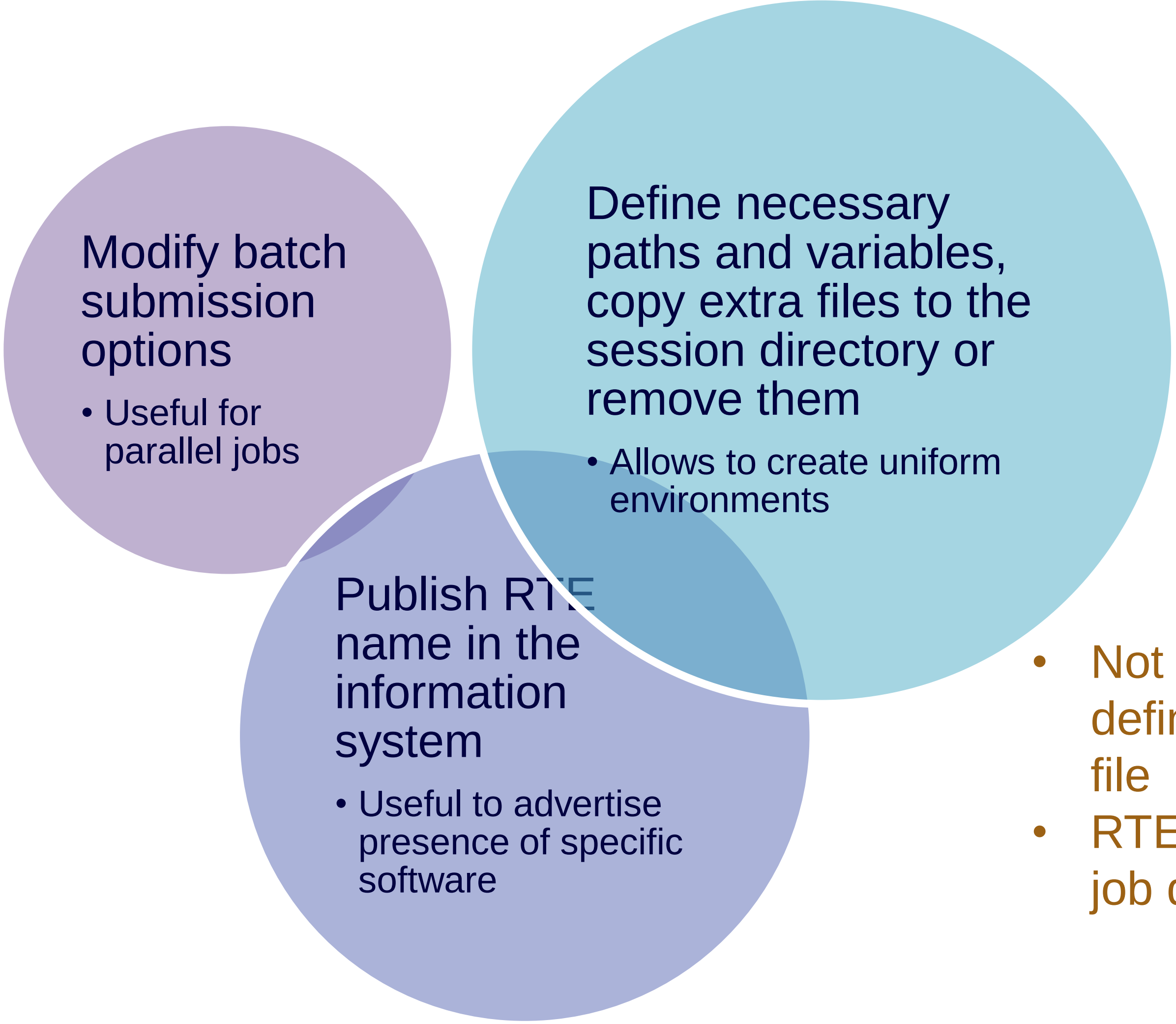


Ways to create a proper environment in ARC



- **Session directory:**
  - A directory where all the files belonging to a single job are collected and created
    - » Scripts, linked input data, output etc
  - Your “sandbox” where you can play using scripts
    - » Single-use, for one job only
    - » A-REX creates and deletes it
- **Runtime Environment (RTE)** scripts:
  - Scripts that prepare uniform environment
    - » Installed by sysadmins, persistent
    - » Advertised in information systems for matchmaking
    - » Executed by A-REX if the job description asks for them

Three main use cases for RTE



Modify batch submission options

- Useful for parallel jobs

Define necessary paths and variables, copy extra files to the session directory or remove them

- Allows to create uniform environments

Publish RTE name in the information system

- Useful to advertise presence of specific software

- Not everything can be defined in the job description file
- RTEs are complementary to job description

Examples of RTEs in infosystem

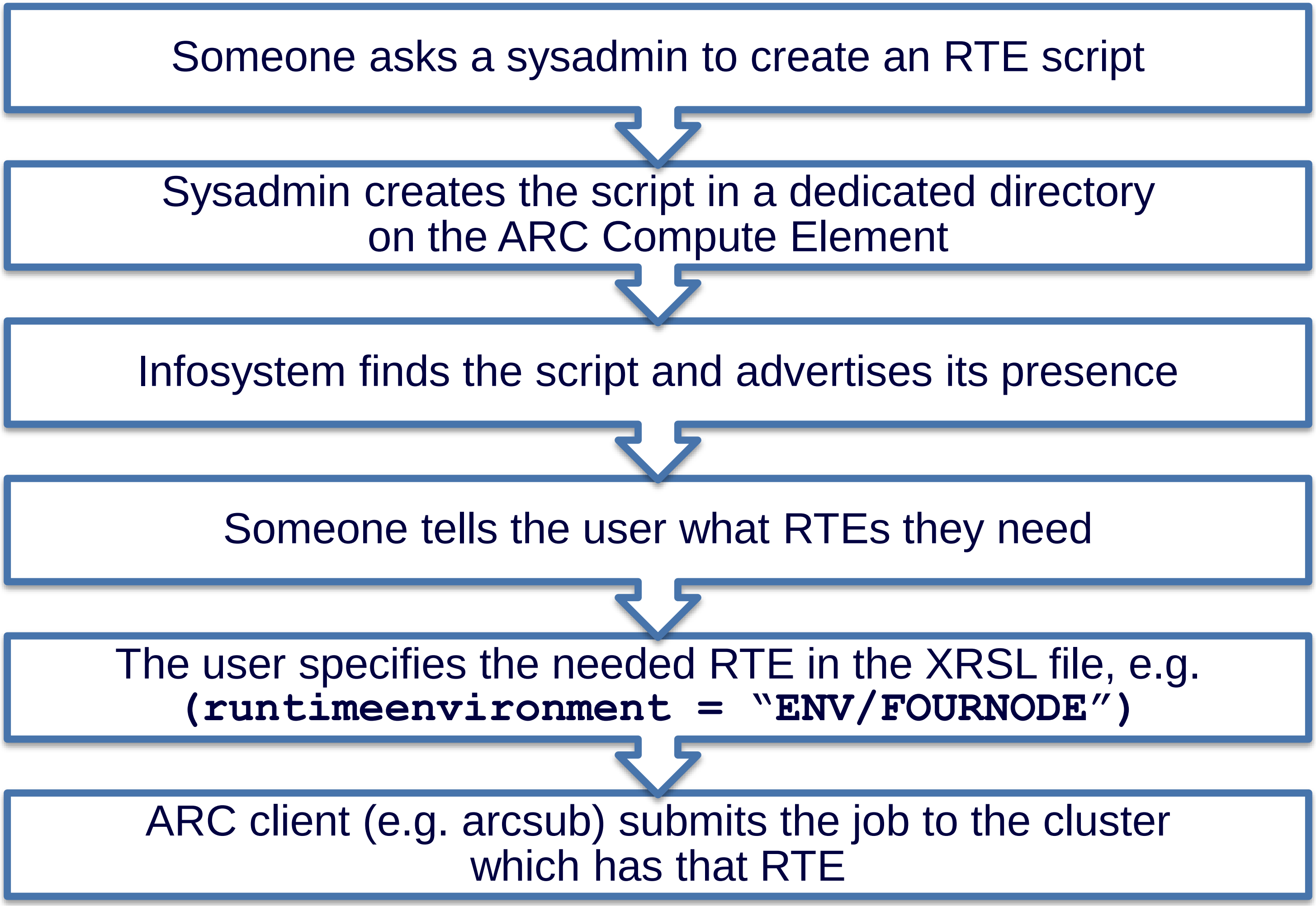
- Note namespaces, e.g.
  - **`APPS, ENV, SAM`**
  - **`APPS/BIO, APPS/CHEM`** etc
- Note different version numbers
  - Indicate available software versions
- Note **`ENV/ONENODE`** and such
  - Used to modify batch submission options
- **`ENV/PROXY`** copies proxy to the session directory
  - Allows your program to work with files on the Grid
- **`SAM/GETCA`** provides monitoring information not found in infosystem

```
APPS/BIO/BLAST+-2.2.28
APPS/BIO/BLAST+-2.2.29
APPS/BIO/BLAST12CORE-2.2.28
APPS/BIO/BOWTIE-2.0.5
APPS/BIO/CUFFLINKS-2.1.1
APPS/BIO/SHRIMP-2.2.3
APPS/CHEM/AMBER-12
APPS/CHEM/AUTODOCK-4.2.3
APPS/CHEM/GROMACS-4.5.5
APPS/CHEM/MEAD-2.2.9
APPS/CHEM/ORCA-3.0.0
APPS/CHEM/ORCA-3.0.1
APPS/IMAG/FREESURFER-5.1.0
APPS/MATH/MCR-R2012A
APPS/MATH/MCR-R2013A
APPS/PHYS/CP2K-2.3rev12343
APPS/PHYS/GEANT4-9.5.1
APPS/PHYS/GPAW-0.9
APPS/PHYS/PARCAS
APPS/STATISTICS/R-2.15.1
APPS/STATISTICS/R-3.0.2
ENV/ASCIIDATA-1.1
ENV/ONENODE
ENV/TWONODE
ENV/FOURNODE
ENV/G95
ENV/NUMPY-1.6
ENV/PROXY
ENV/PYFITS-3.1
ENV/PYTHON-2.7
ENV/SCIPY-0.11
SAM/GETCA
VO-t2k.org-ND280-v10r11p13mod
```

What’s in a name?

- RTE scripts are actually shell-files in a special area on the cluster, managed by sysadmins
  - Not accessible by users
- The names you see are file paths
  - **`APPS/PHYS/PARCAS`** is actually a file called **`PARCAS`** in the RTE folder **`APPS/PHYS/`**
- The hierarchy is convenient for sysadmins to manage the scripts
  - There is no standard convention for such namespaces
  - There is no even technical need for the namespaces
    - » You can find “orphan” RTEs on the Grid
- Information system simply advertises all scripts it finds in the RTE folder and below

How does it work



Someone asks a sysadmin to create an RTE script

Sysadmin creates the script in a dedicated directory on the ARC Compute Element

Infosystem finds the script and advertises its presence

Someone tells the user what RTEs they need

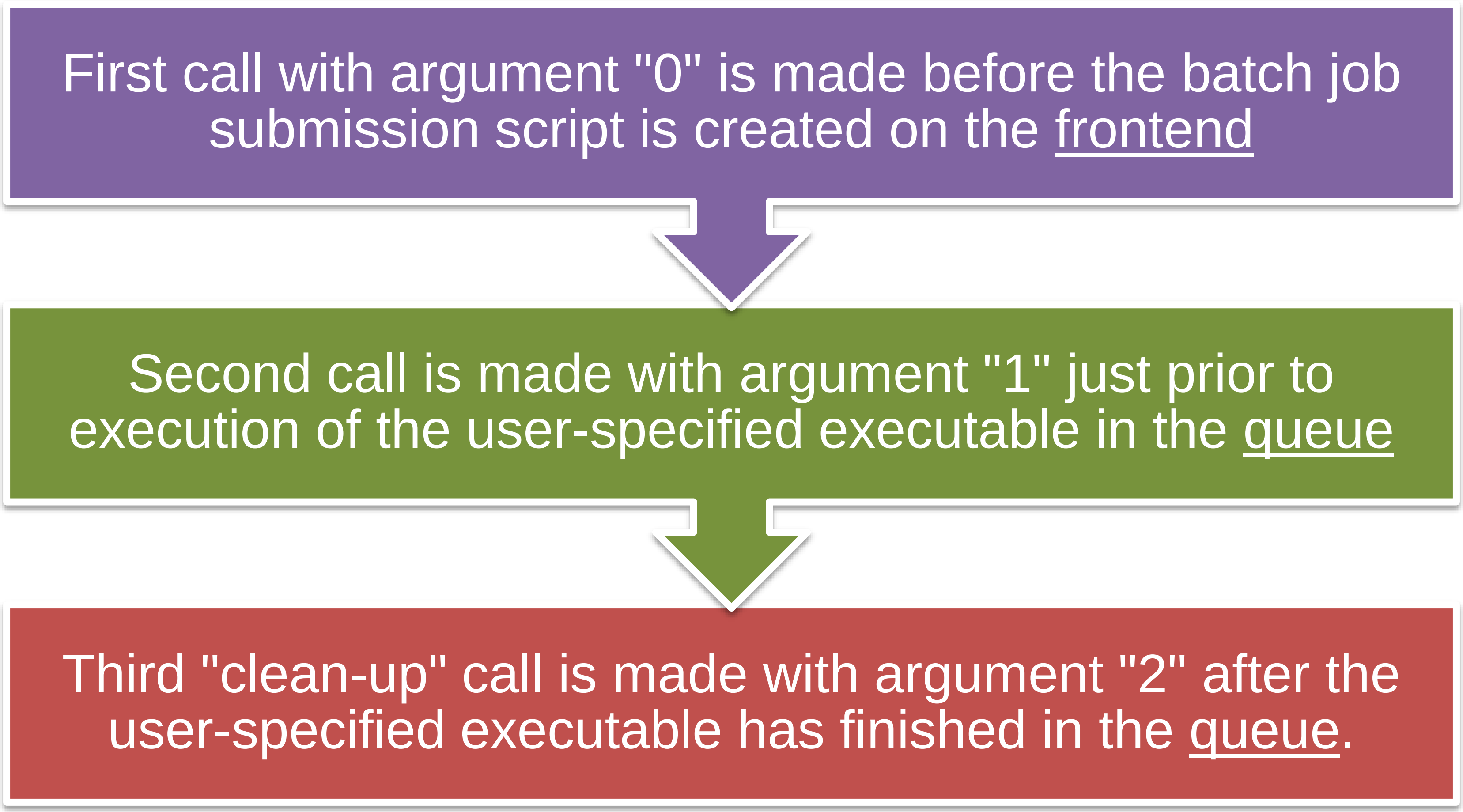
The user specifies the needed RTE in the XRSL file, e.g.

```
(runtimeenvironment = “ENV/FOURNODE”)
```

ARC client (e.g. arcsub) submits the job to the cluster which has that RTE

What happens next

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



First call with argument "0" is made before the batch job submission script is created on the frontend

Second call is made with argument "1" just prior to execution of the user-specified executable in the queue

Third "clean-up" call is made with argument "2" after the user-specified executable has finished in the queue.

Example of a simple RTE script

- The **ENV/CAVER** RTE:
  - Always defines **GETCAVER** variable (needed by the monitoring job)
  - On the first run (option “0”), creates a new folder and copies over CA certificates
    - » On options “1” and “2” it does nothing – there’s nothing to do
  - Makes use of the internal **$joboption_directory** variable to refer to the session directory
  - When a job requesting this RTE starts, it has all the CA keys available for inspection

```
#!/bin/bash
export GETCAVER=1
if [ $# -ne 1 ]; then
          echo "Script needs one and only one parameter (0, 1 or 2)"
          exit 1
fi

if [ $1 = "0" ]; then
        mkdir $joboption_directory/cacerts
        cp -a /etc/grid-security/certificates/* $joboption_directory/cacerts
fi
```

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 the job description file: `(queue=mpi_jobs)`
- And so on, introduce (and remember!) a new queue for every imaginable configuration
- Both ways need a sysadmin

RTE benefits

- In practice it is the only way to make parallel/multi-core applications work
- 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
- Application software version control through RTE names

- Logistical problem:
  - Who/how keeps track of all RTEs?
    - » Currently, just a Web page: http://pulse.fgi.csc.fi/gridrer/htdocs/index.phtml

Exercises

- Get impression of **local** environment on your PC (list installed software packages and pre-set environment variables)
  - If you use a class desktop (CentOS):

```
rpm -qa
printenv
```

  - If you use Ubuntu:

```
dpkg --get-selections
printenv
```

- Check the **remote** environment your Grid job is experiencing: submit simple jobs that execute the above commands. Compare the output. What if the chosen Grid site has no **rpm** software?

```
arcproxy –S nordugrid.org
arccp srm://srm.swegrid.se/nordugrid/tutorial/xrsl/rpm.xrsl ./
arccp srm://srm.swegrid.se/nordugrid/tutorial/xrsl/printenv.xrsl ./
arcsub rpm.xrsl
arcsub printenv.xrsl
arcget <jobID-for-each-job>
```

Exercises

- Find out which RTEs are available e.g. on **alarik-grid.lunarc.lu.se**. Use any of the **ldapsearch, arcinfo, jxplorer** or the Grid Monitor on the Web
  - For example: `arcinfo -l -c alarik-grid.lunarc.lu.se`
- Modify the **printenv.xrsl** so that the Grid job requests some RTEs:
  - Add a **runtimeenvironment** XRSL attribute
  - Submit the Grid job, observe how a matching cluster was selected, then check how the environment was changed on the remote cluster:

```
arcsub printenv_rte.xrsl -d VERBOSE
arcget <jobID>
```

- When in doubt, get

```
srm://srm.swegrid.se/nordugrid/tutorial/xrsl/printenv_rte.xrsl
```

Exercises

- Run an image conversion Grid task that makes use of the **`APPS/TEST/MYCONVERT`** runtime environment:
  - Download and have a look at the original picture at

```
srm://srm.swegrid.se/nordugrid/tutorial/ARC-logo.png
```

  - Download the prepared XRSL file and have a look at it:

```
srm://srm.swegrid.se/nordugrid/tutorial/xrsl/myconvert_rte.xrsl
```

  - Submit the job and view the converted picture:

```
arcsub myconvert_rte.xrsl
arcget <jobID>
```