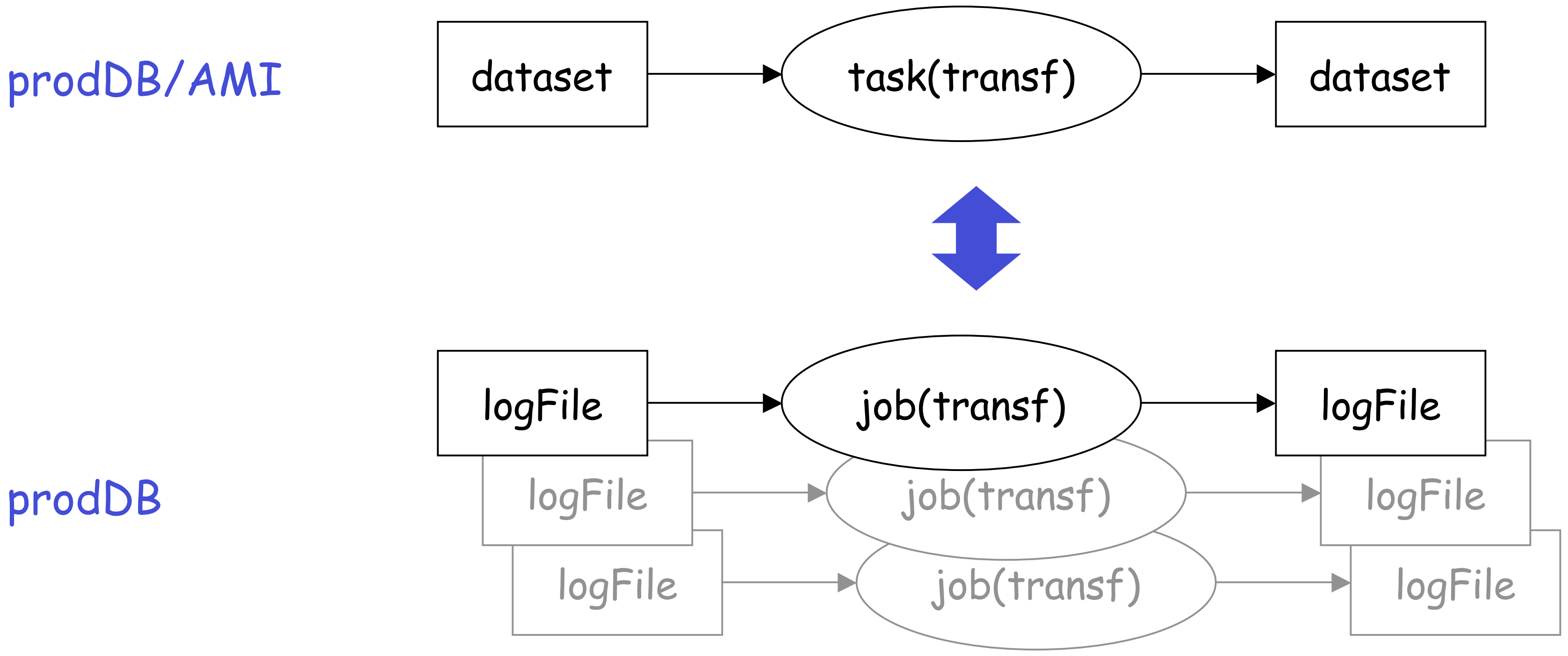
ATLAS Production System in ATLAS Data Challenge 2

Luc Goossens (CERN/EP/ATC)
Kaushik De (UTA)

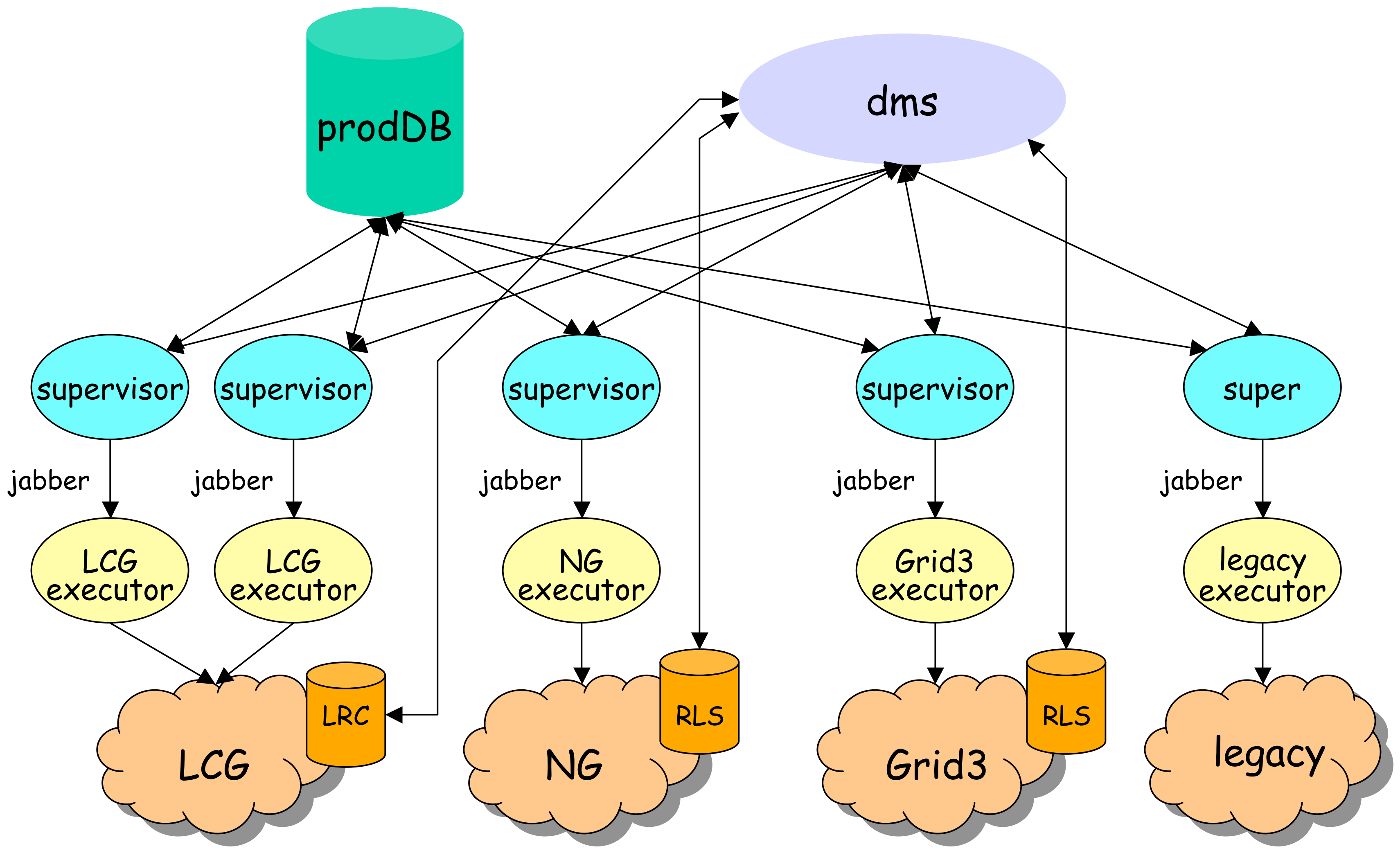
- in this talk
  - introduction
  - terminology and conceptual model
  - architecture and components
  - experience so far
  - conclusions and outlook

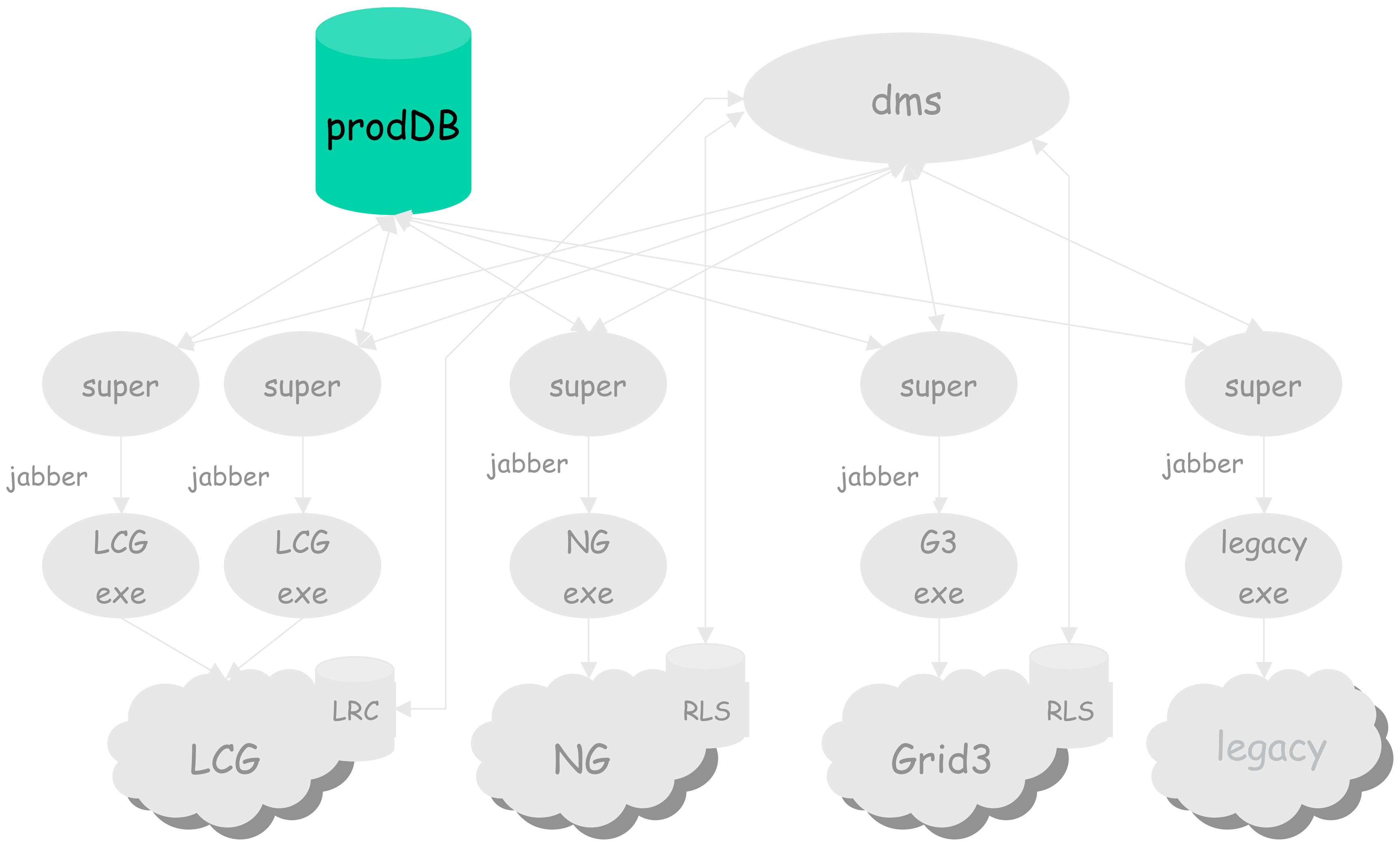
- introduction
  - ATLAS decided to undertake a series of Data Challenges (DC) in order to validate its Computing Model, its software, its data model
  - DC2 started July 2004:
  - introduced the new ATLAS Production System (prodsys) :
    - unsupervised production across many sites spread over three different Grids (US Grid3, NorduGrid, LCG-2)
    - 4 major components:
      - production supervisor
      - executor –
        - one executor per "grid-flavor" developed by corresponding grid experts
      - common data management system
      - common central production database for all ATLAS

- terminology and conceptual model

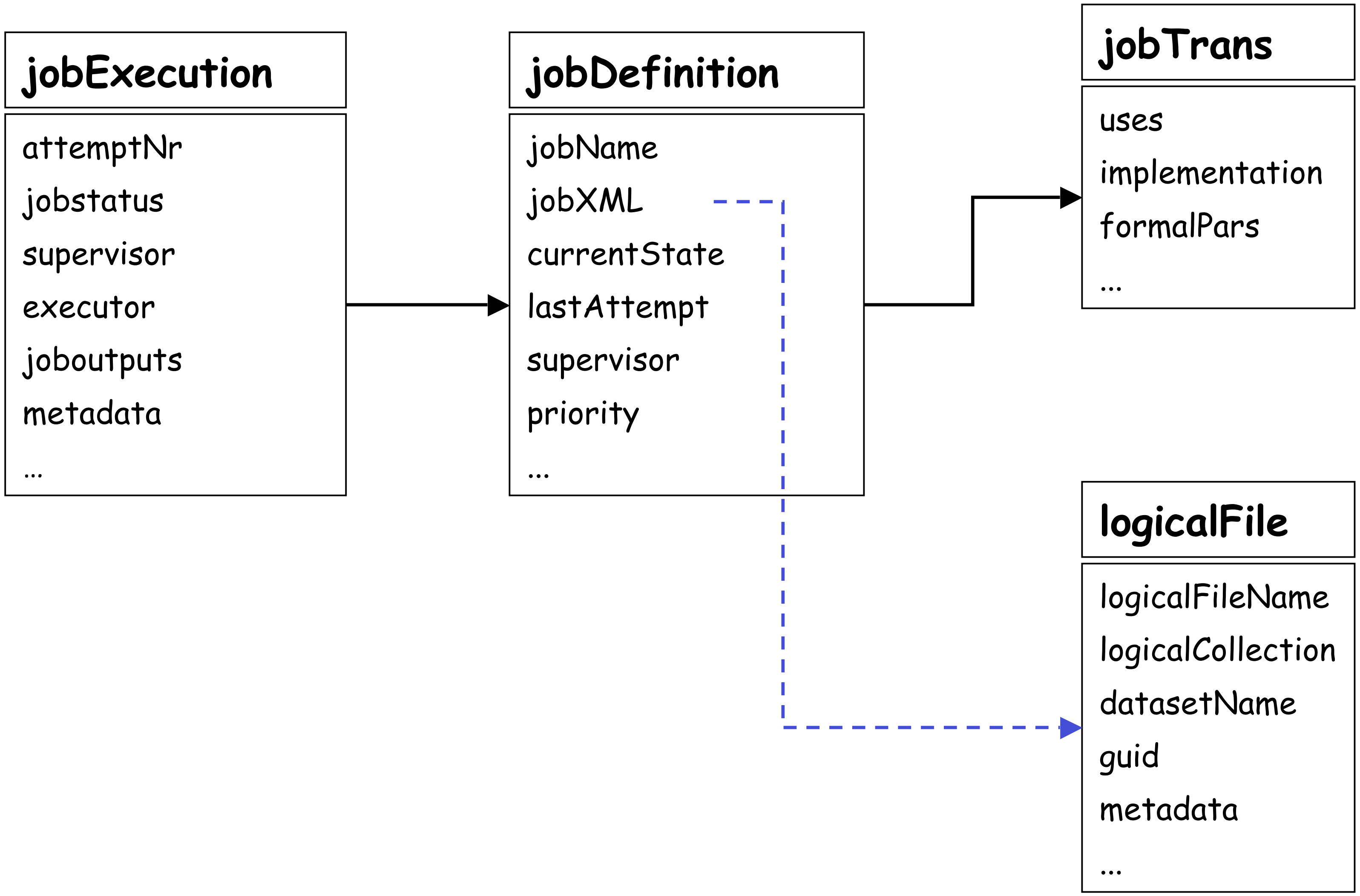


- architecture
  - as simple as possible (well almost)
  - flexible
  - target automatic production
  - based on DC1 experience with AtCom (DC1 interactive production system) and GRAT
    - core engine with plug-ins
  - some buzz technologies
    - XML, Jabber, Webservices, ...
  - federation of grids
    - LCG, Nordugrid, Grid3
    - legacy systems only as backup
  - use middleware components as much as possible
    - avoid inventing ATLAS' own version of grid
      - broker, catalogs, information system, ...
    - risky dependency !


prodDB
dms
supervisor
supervisor
supervisor
supervisor
super
jabber
jabber
jabber
jabber
jabber
LCG executor
LCG executor
NG executor
Grid3 executor
legacy executor
LCG
LRC
NG
RLS
Grid3
RLS
legacy


prodDB
dms
super
super
super
super
super
jabber
jabber
jabber
jabber
jabber
LCG exe
LCG exe
NG exe
G3 exe
legacy exe
LRC
RLS
RLS
LCG
NG
Grid3
legacy

- prodDB = production database
  - holds records for
    - job transformations
    - job definitions
      - status of jobs
    - job executions
    - logical files
  - Oracle database hosted at CERN


jobExecution
attemptNr
jobstatus
supervisor
executor
joboutputs
metadata
...
jobDefinition
jobName
jobXML
currentState
lastAttempt
supervisor
priority
...
jobTrans
uses
implementation
formalPars
...
logicalFile
logicalFileName
logicalCollection
datasetName
guid
metadata
...

jobTrans:formalPars

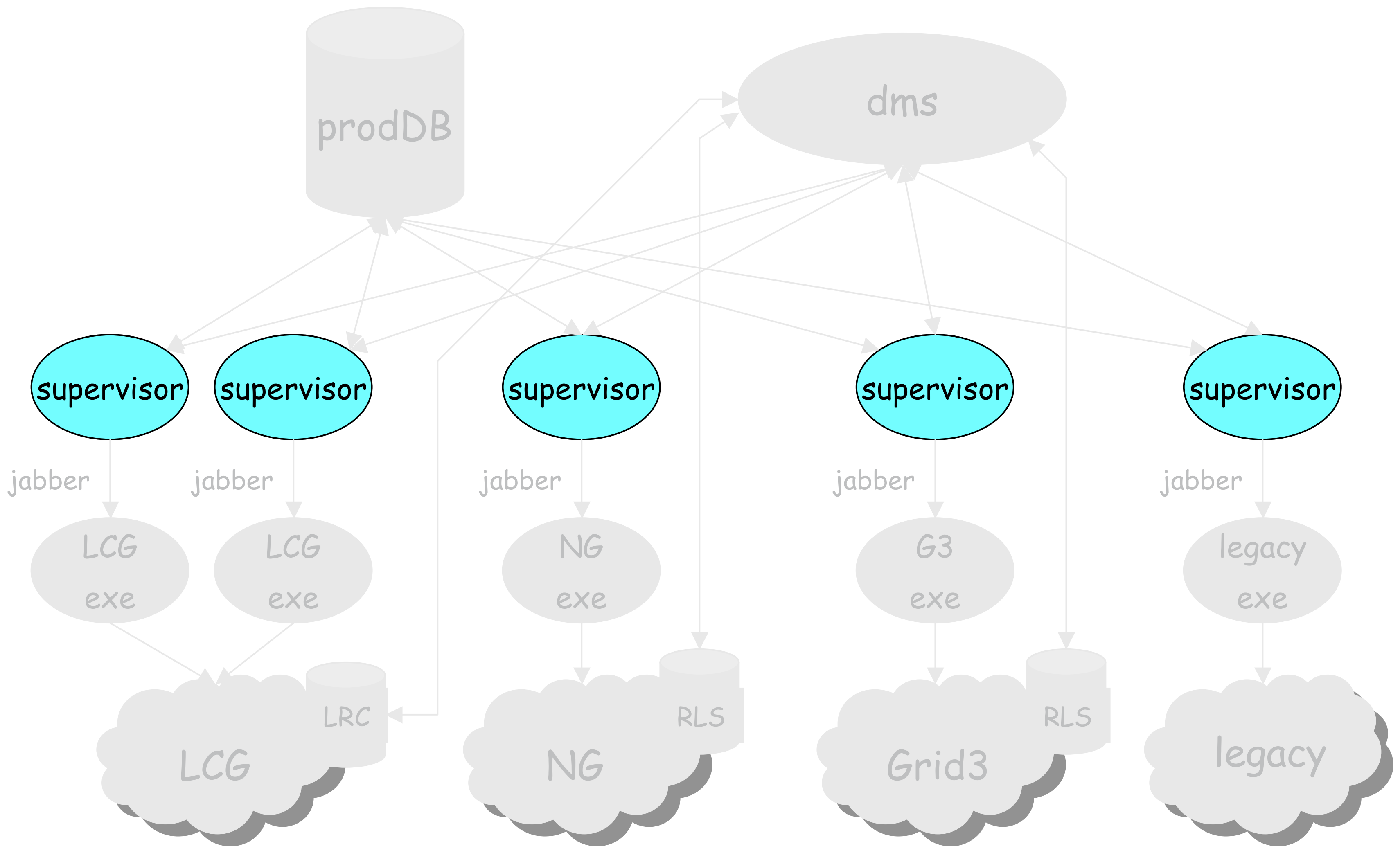
```
<signature>
 <formalPar>
  <name>inputfile</name>
  <position>1</position>
  <type>LFN</type>
  <metaType>inputLFN</metaType>
 </formalPar>
 <formalPar>    <name>outputfile</name>
   <position>2</position>
   <type>LFN</type>
   <metaType>outputLFN</metaType>
 </formalPar>
 ...
 <formalPar>
  <name>ranseed</name>
  <position>7</position>
  <type>natural</type>
  <metaType>plain</metaType>
 </formalPar>
</signature>
```

jobDefinition:jobXML

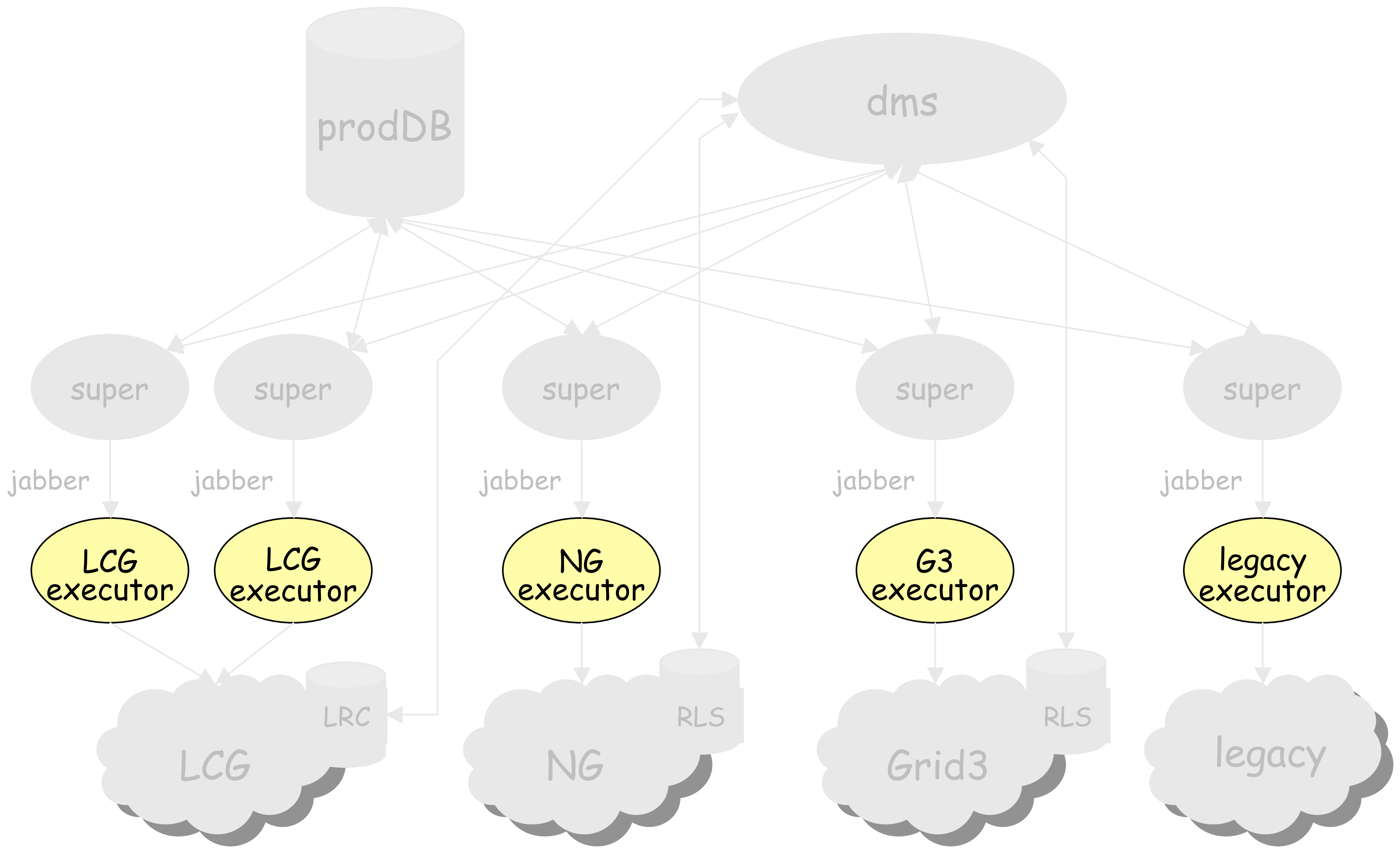
```
<jobDef>
 <jobPars>
  <actualPar>
   <name>inputfile</name>
   <position>1</position>
   <type>LFN</type>
   <metaType>inputLFN</metaType>
   <value>dc2.003014.evgen.M1_minbias._00020.pool.root</value>
  </actualPar>
  ...
 </jobPars>
 <jobInputs>
  <fileInfo>
   <LFN>dc2.003014.evgen.M1_minbias._00020.pool.root</LFN>
   <logCol>/datafiles/dc2/evgen/dc2.003014.evgen.M1_minbias/</logCol>
  </fileInfo>
 </jobInputs>
 <jobOutputs>...</jobOutputs>
 <jobLogs>...</jobLogs>
</jobDef>
```

jobDefinition:jobXML

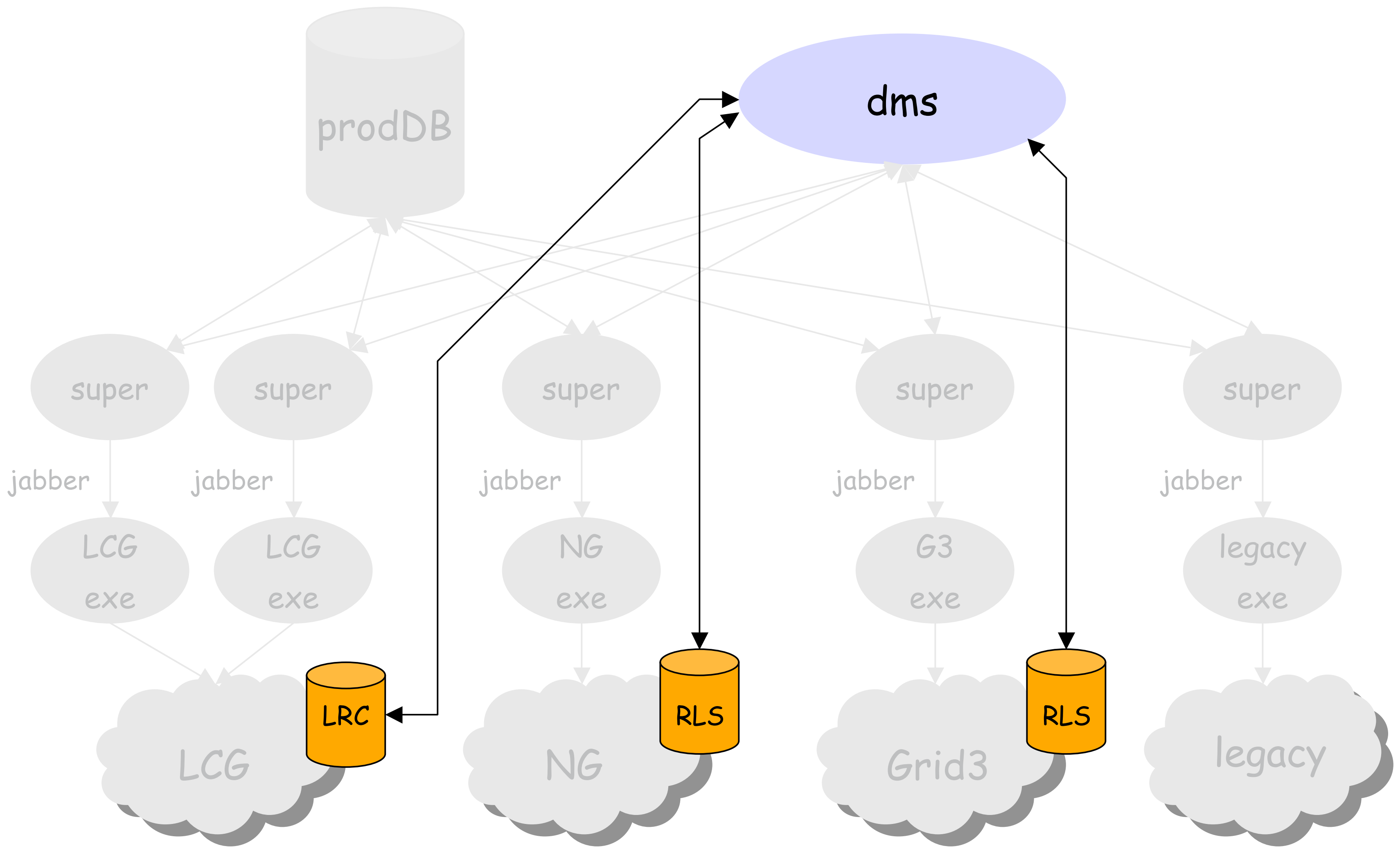
```
<jobDef>
 <jobPars>...</jobPars>
 <jobInputs>   ...  </jobInputs>
 <jobLogs>
  <fileInfo>
   <stream>stdboth</stream>
   <LFN>dc2.003014.simul.M1_minbias._00980.job.log</LFN>
   <logCol>/logfiles/dc2/simul/dc2.003014.simul.M1_minbias/</logCol>
   <dataset><name>dc2.003014.simul.M1_minbias.log</name></dataset>
   <SEList><SE>castorgrid.cern.ch</SE></SEList>
  </fileInfo>
 </jobLogs>
 <jobOutputs>
  <fileInfo>
   <LFN>dc2.003014.simul.M1_minbias._00980.pool.root</LFN>
   <logCol>/datafiles/dc2/simul/dc2.003014.simul.M1_minbias/</logCol>
   <dataset><name>dc2.003014.simul.M1_minbias</name></dataset>
   <SEList><SE>castorgrid.cern.ch</SE></SEList>
  </fileInfo>
 </jobOutputs>
</jobDef>
```


prodDB
dms
supervisor
supervisor
supervisor
supervisor
supervisor
jabber
jabber
jabber
jabber
jabber
LCG exe
LCG exe
NG exe
G3 exe
legacy exe
LRC
RLS
RLS
LCG
NG
Grid3
legacy

- supervisor
  - consumes jobs from the production database
  - submits them to one of the executors it is connected with
  - follows up on the job
  - validates presence of expected outputs
  - takes care of final registration of output products in case of success
  - possibly takes care of clean-up in case of failure
  - will retry n times if necessary
  - implementation -> Windmill
    - http://heppc12.uta.edu/windmill/
  - no brokering
    - "how-many-jobs-do-you-want" protocol
  - possibly stateless
  - uses Jabber to communicate with executors


prodDB
dms
super
super
super
super
super
jabber
jabber
jabber
jabber
jabber
LCG executor
LCG executor
NG executor
G3 executor
legacy executor
LRC
RLS
RLS
LCG
NG
Grid3
legacy

- executor
  - one for each facility flavor
    - LCG (lexor), NG (dulcinea), GRID3 (capone), PBS, LSF, BQS, Condor?, …
  - translates facility neutral job definition into facility specific language
    - XRSL, JDL, wrapper scripts, …
  - implements facility neutral interface
    - usual methods: submit, getStatus, kill, …
  - possibly stateless
  - two implementation strategies
    - executor subclass
    - SOAP adapter + executor webservice (Capone)
  - see other talks in this conference


prodDB
dms
super
super
super
super
super
jabber
jabber
jabber
jabber
jabber
LCG exe
LCG exe
NG exe
G3 exe
legacy exe
LRC
RLS
RLS
LCG
NG
Grid3
legacy

- data management system
  - allows global cataloguing of files
    - we have opted to interface to existing replica catalog flavors
  - allows global file movement
    - an ATLAS job can get/put a file anywhere
  - presents a uniform interface on top of all the facility native data management tools
  - we only counted on ability to do inter-grid file transfers
    - ideally jobs should be able to use input files located in other grids and write output files into other grids
    - this was not exercised
  - stateless
  - implementation -> Don Quijote
    - see separate talk by Miguel Branco

- experience
  - since start of DC2 (July) the system has
    - 235000 jobexecution, 158000 jobdefinition, 251000 logicalfile
      - approx. evenly distributed over the three Grid flavors
    - 157 task, 22 jobtrans
    - consumed ~ 1.5 million SI2k months of CPU (~ 5000 CPU months)
  - we had high dependency on middleware
    - broker in LCG, RLS in Grid3/NG, ...
    - we suffered a lot !
    - many bugs were found and corrected
  - DC2 started before development was finished
    - we suffered a lot !
    - many bugs were found and corrected
  - detailed experience reports per Grid in other talks

- conclusion
  - for DC2 ATLAS relies completely on a federation of grid systems (LCG, Nordugrid, Grid3)
  - the ATLAS production system allows for an automatic production on this federation of grids
  - the ATLAS production system is based directly on the services offered by these grids
  - stress-testing these services in the context of a major production was a new experience and many lessons were learned
  - it was possible, but not easy
    - a lot of manpower was needed to compensate for missing and/or buggy software