# Distributed HTCondor at GRIF

A. Sartirana

# Plan of the Talk.

- HTCondor at GRIF
  - **why?** Motivations;
  - **when?** Timeline of the migration (so far);
  - **how?** Setup generalities;
  - **Feedback** after ~2 years. Goods and Bads;
- a distributed HTCondor pool for GRIF
  - …or at least for a subset of it;
  - **motivations**;
  - general **setup**;
  - tests, **current status**, planning.

# Motivations.

- **Maui** no longer maintained/developed
  - potential **security issues;**
  - potential **scaling issues** as sites grow bigger;
- **multicore** jobs **support** required by LHC Vos
  - not straightforward to implement it in torque/maui;
- no hierarchical fairshare;
- a **general tendency** at many grid sites
  - very **positive feedback.**

**Q4 2014 statements**

# Timeline & Status.

- **HTCondor + ARC-CE** (**Puppet** setup);
- new cluster/endpoint (node16);
- **prod Q4 2014.** Currently ~60% of resources;

- **HTCondor + CREAM** (**Quattor** setup);
- test 10/2014, **prod 4/2015** (**Big Bang**);
- also used for local HTC batch cluster;

- **same** setup **as LLR;**
- new cluster/endpoint;
- **prod 7/2015.** Migration now **completed;**

- currently **testing** (quattor setup);
- including **mpi** cluster at IPNO.

# Feedback.

- **Easy** to put in place
  - for std stuff, follow the doc and it works;
- **documentation** can be difficult
  - **monumental**: actually a + but easily lost in it;
  - some non-std topics poorly documented;
- very **stable** and very **reactive support**…
  - nearly no issues seen so far (mostly on CE);
  - doubts/issues quickly addressed and solved;
- …**but few longstanging issues are there**
  - cgroup pbm at IRFU (no hints from support);
  - cases of jobs mysteriously not running

# Feedback.

- very (very, very, …) **flexible and powerful;**
- quite **different logic** (no queues, classAds,…)
  - may take a while to get used to it;
- **upgrades** are **not transparent**…
  - **need to drain** nodes (auth drain is painful);
- …and may come **with surprises**
  - condor_q out changed, x509* classAds no def at submission, shared_port service by default;
  - …and this just moving from 8.4 to 8.6.

I realize there are lot of ⛔ and 🟡 but the **overall feedback is very positive!**

# Timeline & Status.

- **HTCondor + ARC-CE** (**Puppet** setup);
- new cluster/endpoint (node16);
- **prod Q4 2014.** Currently ~60% of resources;

- **HTCondor + CREAM** (**Quattor** setup);
- test 10/2014, **prod 4/2015** (**Big Bang**);
- also used for local HTC

Focus on this

- **same** setup **as LLR;**
- new cluster/endpoint;
- **prod 7/2015.** Migration now **completed;**

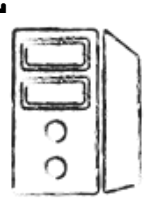
- currently **testing** (quattor setup);
- including **mpi** cluster at IPNO.

# How it works.

- Just a sketch of job-flow/resource-usage mgmt.
  - needed to have an idea how current setup works and how the distr. pool will work;
  - other tools for a complete control of jobs/resources are not mentioned here. E.g.
    - definition of machine resources (GPUs, accel.,…);
    - concurrency limits (slots per job type, licenses,…);
    - … and plenty of stuff I'm not even aware of;
- I'll omit also implementation details of
  - accounting;
  - BDII publication.

# How it works.

**CE**

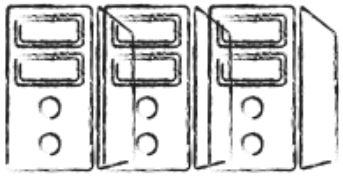


- cream
- condor sched (submission point + jobs tracking)
- accounting

**CM**

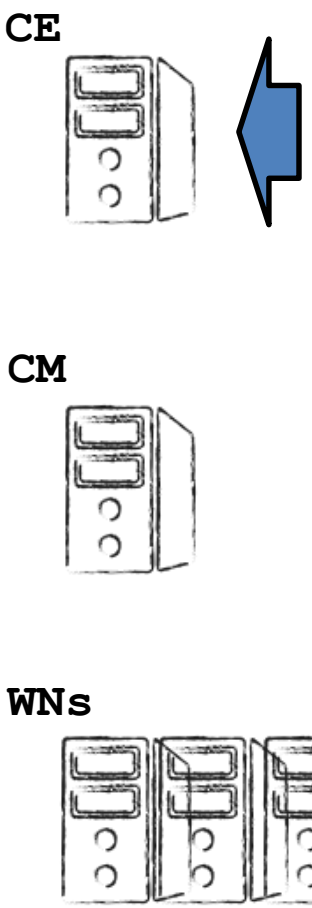
- condor negotiator
- resource BDII. To be moved to CE (see later)
- condor defrag (preempting for MC scheduling)

**WNs**



- execution nodes

# How it works.




**Blah custom** script condor_local_submit_attributes.sh used to **define suitable jobs attributes** (classAds)

VO, DN, FQAN, CREAM QUEUE

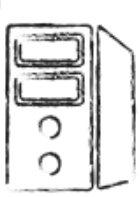
CMS TFC-like configurable stack of regexp

accounting_group, PolicyGroup, WNTag

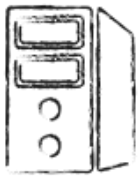
(see next slides to see what these are for)

# How it works.

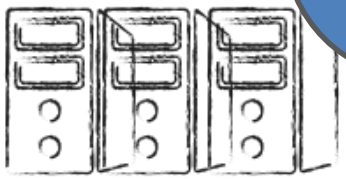
CE



CM



WNs



**Blah custom** script condor_local_submit_attributes.sh used to **define suitable jobs attributes** (classAds)

<unix account> ⇨ accounting_group_user

llrcream/cream-condor-<queue> ⇨ CreamQueue

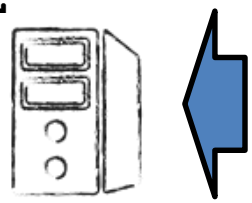
VO,FQAN,DN ⇨ MyVOName/ProxySubject/FQAN [*]

(see next slides to see what these are for)

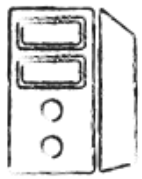
[*] there are standard classAds for these. But, since 8.6, they are not defined at submission time.

# How it works.

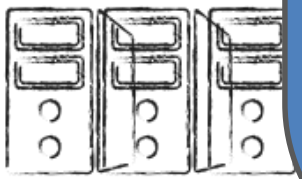
CE



CM



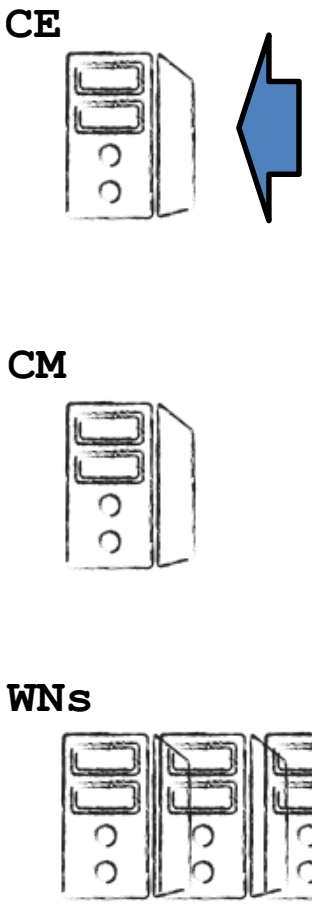
WNs



At submission time, the condor_schedd evaluates the **SUBMIT_REQUIREMENTS**. Boolean expressions, **if one is false** the submission is **rejected** with a message. E.g.

| rule | message if false |
|---|---|
| false | 'This CE is currently draining.' |
| (CreamQueue == “default") \|\| (CreamQueue == "multicore") | 'This queue does not exists.' |
| (RequestCpus == 1) \|\| (CreamQueue == "multicore") | 'Multicore jobs should be sent to multicore queue.' |
| MyVOName =!= "cms" | ‘The CMS queue is draining.' |

# How it works.




Once a job is submitted **SYSTEM_PERIODIC_REMOVE** is periodically evaluated. **If true it removes the job**. E.g.

**GC-ing held jobs**

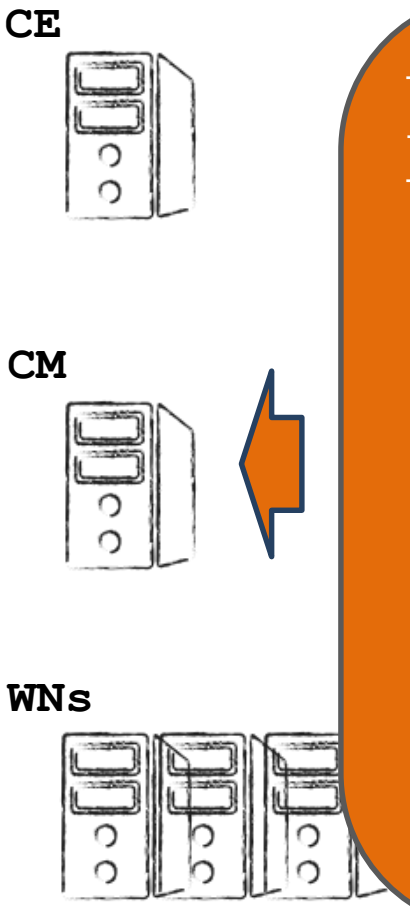
**WCT limit on running jobs**

```
(JobStatus == 5 && time() - EnteredCurrentStatus >3600*48) | |
((JobStatus == 2)&&(($(MAXWALLTIME)>0)&&((time() -
EnteredCurrentStatus) > (60*$(MAXWALLTIME)))))
```

**We select the limits** as MAXWALLTIME **using the PolicyGroup** classAd. e.g.

```
MAXWALLTIME = IfThenElse( PolicyGroup ==
"vo_grif_fr.gridq“,60,4320)
```

# How it works.

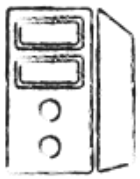



The **Negotiator** implements the **matchmaking** respecting the accounting groups quotas.
They have **hierarchic quota implementation**

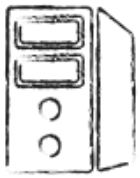
```
GROUP_QUOTA_DYNAMIC_group_mygroup = 1.0
GROUP_QUOTA_DYNAMIC_group_mygroup.sub1 = 0.1
GROUP_QUOTA_DYNAMIC_group_mygroup.sub2 = 0.2
GROUP_QUOTA_DYNAMIC_group_mygroup.sub3 = 0.1
GROUP_QUOTA_DYNAMIC_group_mygroup = 2.0
...
```

# How it works.

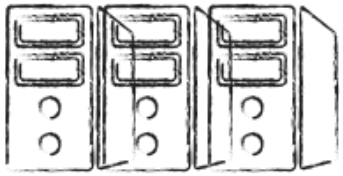
CE



CM



WNs



At **node level** a **START** boolen classAd can be defined to decide whether a **node can run a job**. For example we have it composed of few terms

START_OFFLINE ⇨ false if the node is offline

START_DRAIN ⇨ ( x509UserProxyVOName == "ops" ) if the node is draining

START_CUSTOM ⇨ whatever we want

**START_TAG** ⇨ ((WNTag == "ALL") | | (WNTag == “bigmem") | | false)

# How it works.

- Easily configurable via quattor

```
variable CONDOR_CONFIG = {
  SELF['pwd_hash']=…
  SELF['allow'] = '*.in2p3.fr';
  SELF['groups'] = dict( 'group_cms', nlist('quota', '1.0') );
  SELF['params']['MAXWALLTIME']['vo_grif_fr.gridq']=60;
  SELF['group_defaults']['autoregroup'] = true;
  SELF['multicore'] = true;
  SELF['submit_rules'] = dict(
    #SUBMIT_REQUIREMENTS: “name”,dict(“rule”,<RULE>,”reason”,<TEXT>)
  );
  SELF['tags_regexps'] = list(<RULES FOR Wntags>);
  SELF['gc_rules']= list(<PERIODIC REMOVES RULES>);
  SELF;
};

variable WN_ATTRS = dict(
  "DEFAULT",          dict("state", "free"),
  "polgrid121.in2p3.fr", dict("tags",list(“bigmem")),
  …
);
```

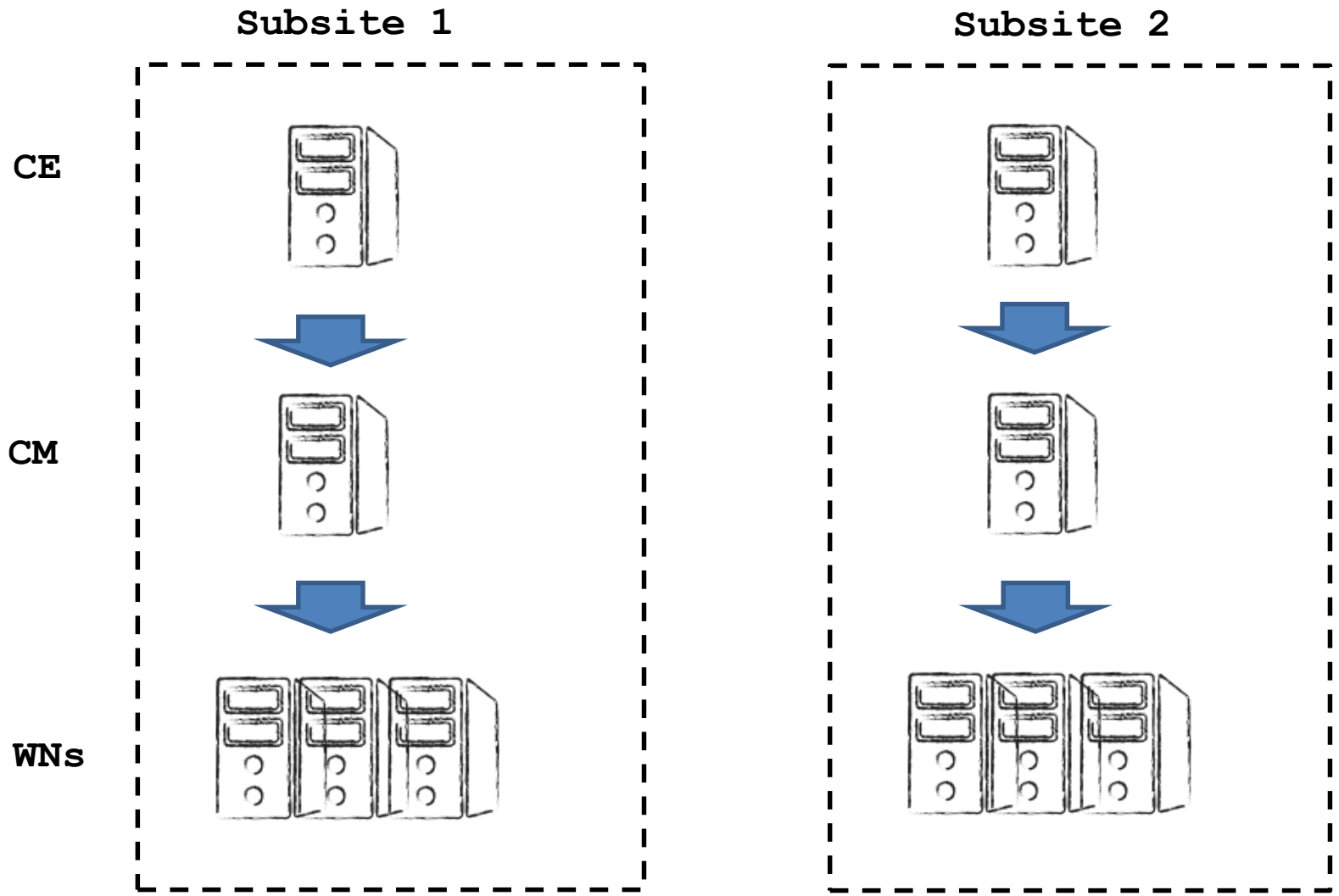
# Distributed setup.

- **Each** GRIF **subsite** has **its own** T2 **cluster**
  - doing otherwise **with torque/maui** was **not technically straightforward**…
    - shared homes, non std ports, HN needs WNs list;
  - …would have led to a **SPOF**…
    - no HA setup (at least for CM);
  - …and potential **scaling issues**
    - (2 subsites == ~7k slots);
- **not optimal** for many reason
  - e.g. **non optimal resources usage** by VOs
    - **burden of** sub-resources **balancing on VOs.** Even when no pbm with N storage inst. (e.g. CMS).

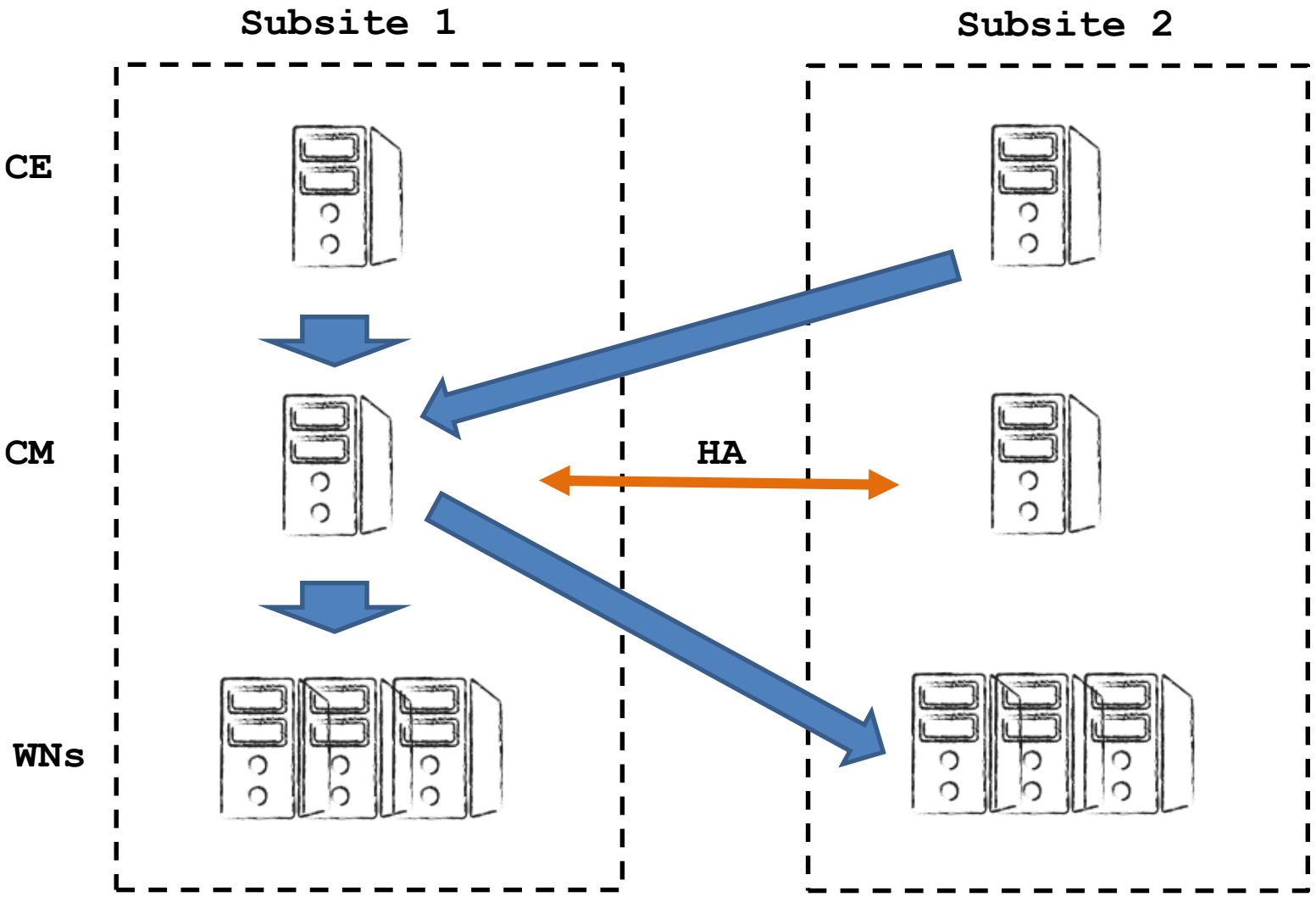
# Distributed setup.

- now **with HTCondor** we have the possibility to "merge" (some of the) **resources in one pool**
  - compliant with a **distributed setup**
    - flexible management of WAN ports;
    - no need for shared homes;
    - CM does not need to have the list of WN's;
  - **HA** setup **available**
    - one active Negotiator + N backups;
  - **no scaling issues**
    - LHC VO's pilot pools manage O(100k) jobs in a distributed env.

# Distributed setup.



# Distributed setup.




# What it takes….

- …to make this work? Not much indeed
  - the time to look at it…
    - low prio: 1y WCT ~ 2 weeks CPUT;
  - …and some **easy technical setup**;
- take care of cross-firewalls **WAN interaction**
  - **"shared port"** condor service
    - pipes all the inter-services communications on one port (default collector port 9618);
    - good to have it (to avoid ports explosion) even in non distr. setup. **Default since 8.6.**
  - few other **CREAM ports** (job notification: 9091);

# What it takes….

- **high availability** setup
  - HAD and REPLICATION services
    - works just fine with the documented config (well… after some iteration with devs);
    - takes care of HA between primary and the secondary negotiator(s);
- a suitable and hierarchic **Quattor config**
  - each subsite configure its own part: CE and WNs
    - condor makes this easy (e.g. no central WN list);
    - take care of specific stuff: e.g. supported vos;
  - centrally configure the Negotiator and ensure the consistency of the whole.

# Devil is in the details.

- **Accounting?**
  - **current setup** (each CE accounts its jobs) **ok**
    - not "subsite contribution". Only CE "popularity";
- **BDII publication**? Currently on CM
  - glue2 shares depend on queues and VOs
    - now purely in the CE (and subsite) scope;
  - so publication **moved to CE**
    - BTW it makes sense removing grid stuff from CM;
  - **redefine glue2 shares** ids
    - was: GLUE2ShareID=<QUEUE>_<VO>_<CM>_ComputingElement
    - now: GLUE2ShareID=<CE>_<QUEUE>_<VO>_<CM>_ComputingElement

# *Devil is in the details.*

- **nodes/jobs matching**
  - varies from subsite to subsite
    - different (and local) VO supported;
    - local sw areas;
    - specific HW (RAM, disk, …);
    - local downtimes;
  - technically not pbm (condor kung-fu is strong)
    - WNTags are there for this;
  - but logistic is complex. **Step-by-step procedure**
    - **step 0: subsites** logically **separated** (jobs of one CE go in the same subsite's nodes);
    - and **allow cross-subsite** submission **VO-per-VO** when sure it is ok.

# First tests.

- Multisite **testbed**
  - **primary** negotiator at **LPNHE;**
  - **secondary** negotiator + **CREAM-CE** at **LLR;**
  - **CREAM-CE** at **LPNHE;**
  - 4 **WNs** at **LLR;**
  - 1 **WN** at **LAL;**
- **functional tests**
  - job submission, scheduling, execution;
  - WNTags matching;
  - HA switching between primary and secondary;
- **completed beginning 2017.** All ok.

# Roadmap to production.

- Decided to proceed with **merging LLR and LAL**
  - natural choice seen the current status;
- **"merging" test** using preprod instances
  - tested that we can **merge instances with jobs running on it;**
  - **done last week** and all went fine (that is jobs and services survived);
  - now use this instance to **check/fix last things** about publication/accounting;
- **plan the step to production.**


# Summing Up.

- **GRIF** running **condor in prod** since **~2 years**
  - ARC-CE/puppet at **IRFU;**
  - CREAM-CE/quattor at **LLR** and **LAL;**
- very **positive feedback**
  - stable, well supported, flexible and powerful, no scaling issues;
- now possible to aggregate subsites in **distributed HTCondor pools**
  - general setup quite easy. Few tricky points;
  - **made functionality and merging tests;**
  - **ready for** moving it in **production.**

# Questions?