



University of  
Zurich <sup>UZH</sup>

Grid Computing Competence Center

---

# Pilot jobs model.

**Sergio Maffioletti**

GC3: Grid Computing Competence Center,  
University of Zurich

## Scaling issues with current submission model

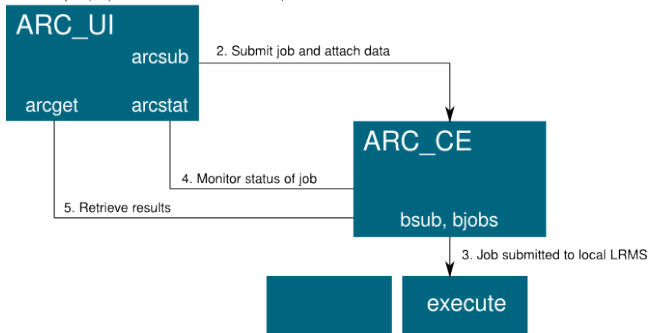
- **Exact** scheduling comes with a price: scheduler needs to have a **comprehensive** view of the system.
- Need **Real-time** information system
- Need to make **prediction** on how the local LRMS will schedule jobs.
- **Meta-scheduler** is just another scheduling layer (is NOT replacing local scheduling)

## Scaling issues with current submission model

- Each job is a **unique** unit of work (e.g. data association)
- Need to keep track of **all** running jobs
- is all **centralized**: there's a single entity in control

# Matchmaking model

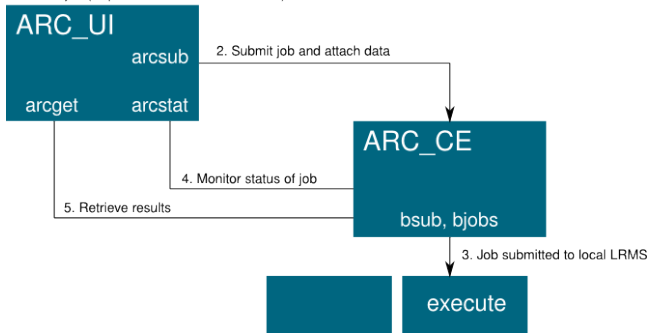
1. Define job (requirements & data information)



- Description of the job done **prior** submission
- Repartition of data done at job **description**
- Each job represents a **precise unit** of work
- **Each job** needs to be monitored

# Matchmaking model

1. Define job (requirements & data information)



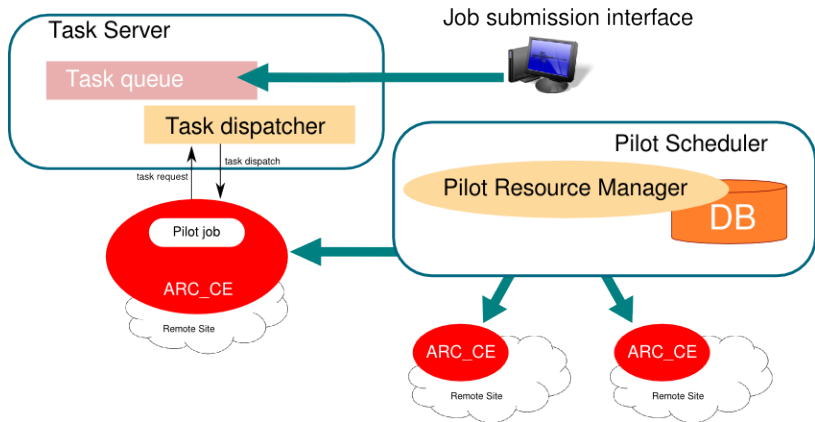
- Matchmaking done at submission time
- Data transferred together with the job
- Each failed job needs to be resubmitted
- Scheduling requires lot of information

## Pilot job model

A **pilot** job is a special type of job that **lands** on a distributed resource and then **pulls in** the users task to start executing it.

This approach, often called the **PULL** model (in contrast with the Matchmaking called **PUSH** model)

# Generic Pilot model



## Advantages

This **late binding** of pilot jobs to processing tasks prevents **latencies** and **failure modes** in slot acquisition.

- Simplified **Matchmaking** process
- **Node validation** prior execution of application
- **Distribution** of load done according to the **responsiveness** of the entire system
- No need sophisticated **Information System**

# Matchmaking

Instead of trying to predict where the job will start first

- pilots are **generic** containers sent to **all available** resources
- Pilot can **provide information** to Master node to do matchmaking
- The **first** pilot that starts gets the task
- The remaining will either start **another available** task
- or **terminate**

## Node Validation

- Pilot inspects **local environment** prior execution (and data staging)
- Correctly **configured** users tasks are less likely to fail

Pilot jobs may **fail** as well, but this will **not be visible** at the user level.

User only monitor the **task** success ratio.

## Distribution of load

- Task is requested **only** when a pilot is ready to execute it
- No need to gather a priori **full knowledge** of the system

# Simplified Information System

- At Pilot submission time, only **limited knowledge** of the entire system is required by Pilot Scheduler
- **'late binding'** of workload jobs to processing tasks
- Pilot provide **real-time information** of the node where it is running (thus providing a more realistic representation of the execution system)

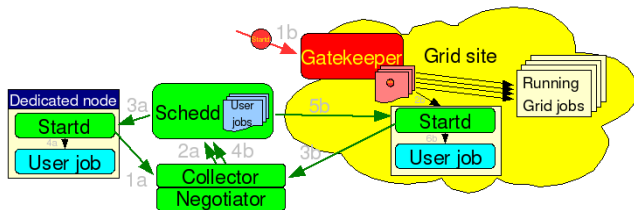
## Disadvantages

- Outbound connectivity required
- bypass local site allocation policies
- pilot job needs to be able to switch **user identity** when running
- The infrastructure is **flooded** by lot of pilot jobs
- **Complexity** now reside at the Pilot Scheduler and Tasks Manager level

# Pilot Scheduler

- Pilot Scheduler needs to **monitor** running pilots
- What strategy when deciding which and how many pilots
  1. pre-defined set of pilots (send blocks of similar pilots)
  2. adaptive (detect how the system is behaving and determine which pilot container works best)

## Condor Glide-in



- Deploy Condor **daemons** and configuration on the remote resource
- Resource **reports** to and **joins** a local Condor pool
- Receive **standard** Condor jobs from joined pool

## Execution task

- Starts the Condor **daemons** (glide-in does this by submitting a Condor job to run under the grid universe)
- Job runs condor **agent** on the remote grid resource
- Condor agent contacts the **local pool**'s condor collector to join the pool
- Condor daemons exit gracefully when no jobs run on the daemons for a preset period of time.

## After Condor glide-ins

Once the resource has been added to the local Condor pool with condor glide-in, job(s) may be submitted.

# Final Project Assignment

Pilot-based Analysis pipeline of positive Darwinian selection on gene families

Selectome is a database of **positive selection** based on a rigorous branch-site specific likelihood test.

Positive selection is detected using **CODEML** on all branches of animal gene trees.

[http://wiki.isb-sib.ch/grid-selectome/  
Codeml\\_Requirements\\_Form](http://wiki.isb-sib.ch/grid-selectome/Codeml_Requirements_Form)

## Application specs

Provided input folder of the following structure:

- 1 folder per Primate type
- each folder contains
  - *Primate\_identifier.XXX.nwk*
  - *Primate\_identifier.XXX.phy*
  - *Primate\_identifier.XXX.H0.ctl*
  - *Primate\_identifier.XXX.H1.ctl*
- 1 application run requires
  - H[0,1].ctl, .phy, .nwk
  - 1 core, 1GB mem, 1':60 minutes walltime
  - Requirement: H0 and H1 MUST be executed on same node

```
$CODEML_LOCATION/codeml.pl <filename>.H[0,1].ctl  
<filename>.phy <filename>.nwk
```

## Application specs

- single-user execution (no need to switch from pilot user to task user)
- each execution should bring back:
  - `codeml.log` (stdout and stderr)
  - `Primate_identifier.XXX.mlc` (codeml output)
- output is sent by pilot to an output aggregator service (free to use any available service)

## Job execution specs

if CODEML execution:

- terminated successfully: **nothing**
- failed: **retry**
- goal is 100% data analyzed

## Implementation guidelines

- CODEML is available in a **Virtual Machine** (details on this will come in next lectures)
- VM to be submitted to SMSCG as **regular** grid job
- Execution using a pilot-like model:
  - VM **starts, connects** to a Master that **decides** which task to assign
  - VM **downloads** the task (the input files file) and **executes** CODEML on it
  - Master should recognize whether a job/task is **stale** and assign the same task to the next pilot

## available datasets

3 test datasets (10, 100, 1000) 1 production dataset (12000)