# Table of Contents

# DIRAC Priority schema

This page describes how the DIRAC handles priorities

## Scenario

There are two user profiles:

- Users that submit jobs on behalf of themselves. For instance normal analysis users.
- Users that submit jobs on behalf of the group. For instance production users.

In the first case, users are competing for resources, and on the second case users share them. But this two profiles also compete against each other. DIRAC has to provide a way to share the resources available. On top of that users want to specify a "UserPriority" to their jobs. They want to tell DIRAC which of their own jobs should run first and which should ran last.

DIRAC implements a priority schema to decide which user gets to run in each moment so a fair share of CPU is kept between the users.

## Priority implementation

DIRAC handles jobs using *TaskQueues*. Each *TaskQueue* contains all the jobs that have the same requirements for a user/group combination. To pioritize user jobs, DIRAC only has to prioritize *TaskQueues*.

To handle the users competing for resources, DIRAC implements a group priority. Each DIRAC group has a priority defined. This priority can be shared or divided amongst the users in the group depending on the group properties. If the group has the **JOB_SHARING** property the priority will be shared, if it doesn't have it the group priority will be divided amongst them. Each *TaskQueue* will get a priority based on the group and user it belongs to:

- If it belongs to a **JOB_SHARING** group, it will get 1/N of the priority being N the number of *TaskQueues* that belong to the group.
- If it does **NOT**, it will get 1/(N*U) being U the number of users in the group with waiting jobs and N the number of *TaskQueues* of that user/group combination.

On top of that users can specify a "UserPriority" to their jobs. To reflect that, DIRAC modifies the *TaskQueues* priorities depending on the "UserPriority" of the jobs in each *TaskQueue*. Each *TaskQueue* piority will be P*J being P the *TaskQueue* priority. J is the sum of all the "UserPriorities" of the jobs inside the *TaskQueue* divided by the sum of sums of all the "UserPiorities" in the jobs of all the *TaskQueues* belonging to the group if it has **JOB_SHARING** or to that user/group combination.

# Dynamic share corrections

DIRAC includes a priority correction mechanism. The idea behind it is to look at the past history and alter the priorities assigned based on it. It can have multiple plugins but currently it only has one. All correctors have a CS section to configure themselves under /Operations/Scheduling//ShareCorrections. The option Operations/Scheduling//ShareCorrections/ShareCorrectorsToStart defines witch correctors will be used in each iteration.

## WMSHistory corrector

This corrector looks the running jobs for each entity and corrects the priorities to try to maintain the shares defined in the CS. For instance, if an entity has been running three times more jobs than it's current share, the priority assigned to that entity will be one third of the corresponding priority. The correction is the inverse of the proportional deviation from the expected share.

Multiple timespans can be taken into account by the corrector. Each timespan is weighted in the final correction by a factor defined in the CS. A max correction can also be defined for each timespan. The next example defines a valid WMSHistory corrector:

```
ShareCorrections
{
  ShareCorrectorsToStart = WMSHistory
  WMSHistory
  {
    GroupsInstance
    {
      MaxGlobalCorrectionFactor = 3
      WeekSlice
      {
        TimeSpan = 604800
        Weight = 80
        MaxCorrection = 2
      }
      HourSlice
      {
        TimeSpan = 3600
        Weight = 20
        MaxCorrection = 5
      }
    }
    lhcb_userInstance
    {
      Group = lhcb_user
      MaxGlobalCorrectionFactor = 3
      WeekSlice
      {
        TimeSpan = 604800
        Weight = 80
        MaxCorrection = 2
      }
      HourSlice
      {
        TimeSpan = 3600
        Weight = 20
        MaxCorrection = 5
      }
    }
  }
}
```

The previous example will start the WMSHistory corrector. There will be two instances of the WMSHistory corrector. The only difference between them is that the first one tries to maintain the shares between user groups and the second one tries to maintain the shares between users in the *lhcb_user* group. It makes no sense to create a third corrector for the users in the *lhcb_prod* group because that group has **JOB_SHARING**, so the priority is assigned to the whole group, not to the individuals.

Each WMSHistory corrector instance will correct at most x[ 3 - 1/3] the priorities. That's defined by the *MaxGlobalCorrectionFactor*. Each instance has two timespans to check. The first one being the last week and the second one being the last hour. The last week timespan will weight 80% of the total correction, the last hour will weight the remaining 20%. Each Timespan can have it's own max correction. By doing so we can boost the first hour of any new entity but then try to maintain the share for longer periods. The final formula would be:

```
hourCorrection = max ( min( hourCorrection, hourMax ), 1/hourMax )
weekCorrection = max ( min( weekCorrection, weekMax ), 1/weekMax )
finalCorrection = hourCorrection * hourWeight + weekCorrection * weekWeight
finalCorrection = max ( min( finalCorrection, globalMax ), 1/globalMax )
```

---

This topic: LHCb > DIRACWMSPrioritySchema
Topic revision: r3 - 2009-09-07 - AdriaCasajus