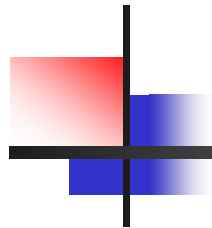


# On Scheduling on the Grid



Gabriel Mateescu  
Grid Computing Group  
Leibniz-Rechenzentrum  
mateescu@lrz.de



# Overview

---

- What is metascheduling
- Where does metascheduling fit in the Grid service hierarchy
- Stand-alone metascheduling service
- Metascheduler as an extension of the Job Execution services
- Conclusions



# What is a Metascheduler

---

- A local scheduler, aka a Local Resource Management System (LRMS), is the workload management system that decides **what jobs** are run, at **what time**, and on **which resource**
- A metascheduler interacts with several local schedulers to schedule job that originate from clients external to the LRMS
- Unlike a local scheduler, which controls the resources assigned to jobs, a metascheduler does not control the remote resources
  - At most, a metascheduler can negotiate with LRMS a certain allocation of resources
  - Negotiation: WS-Agreement, Advance Reservation

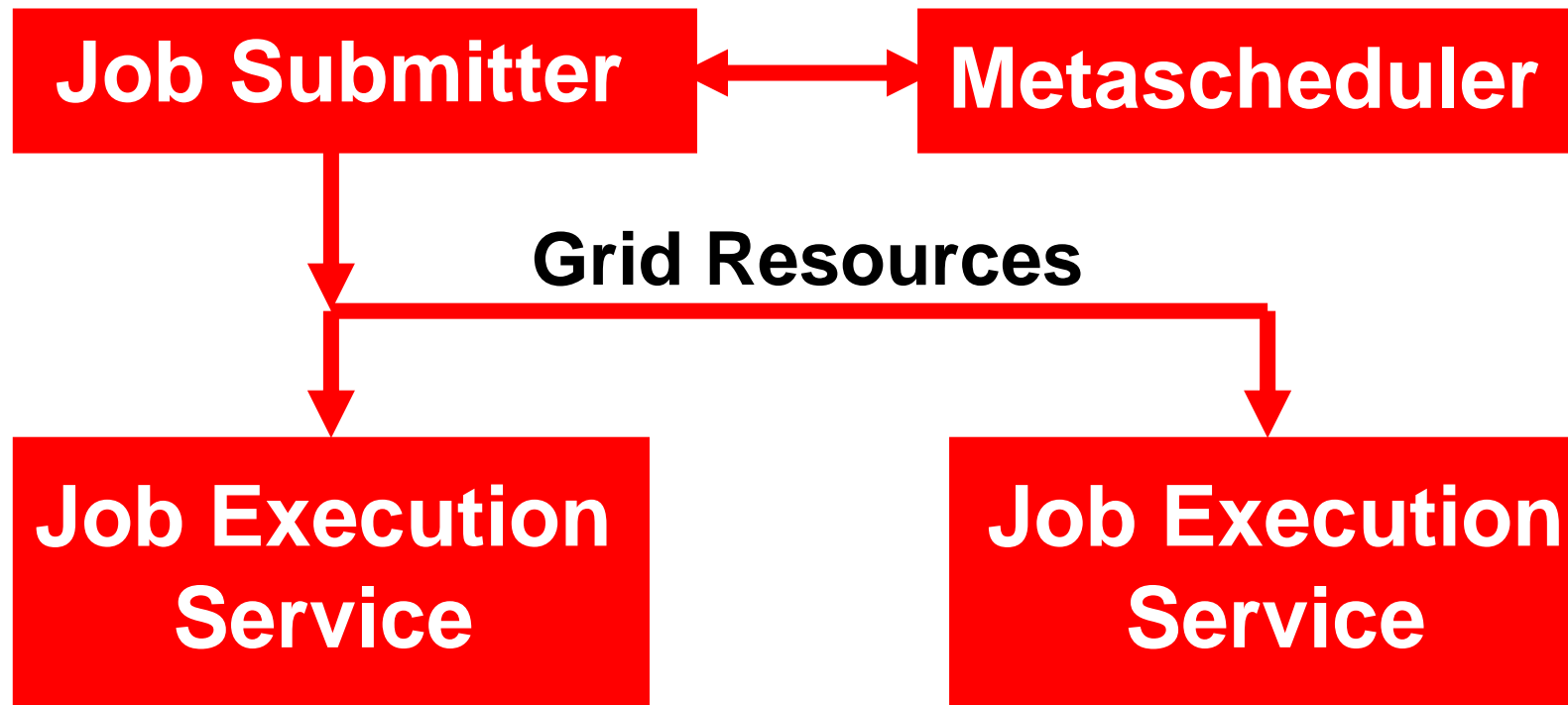


# When to meta-schedule?

---

- A metascheduler service can be exposed in several ways
  - A service that receives a job description from a job submitter and returns the resource selected for that job
  - As a “smart” job execution service that performs, in addition to the regular functions of a job execution service, the functions of
    - Scheduling to a grid resource
    - File staging (either data transfer or only setup)

# Stand-alone Metascheduler



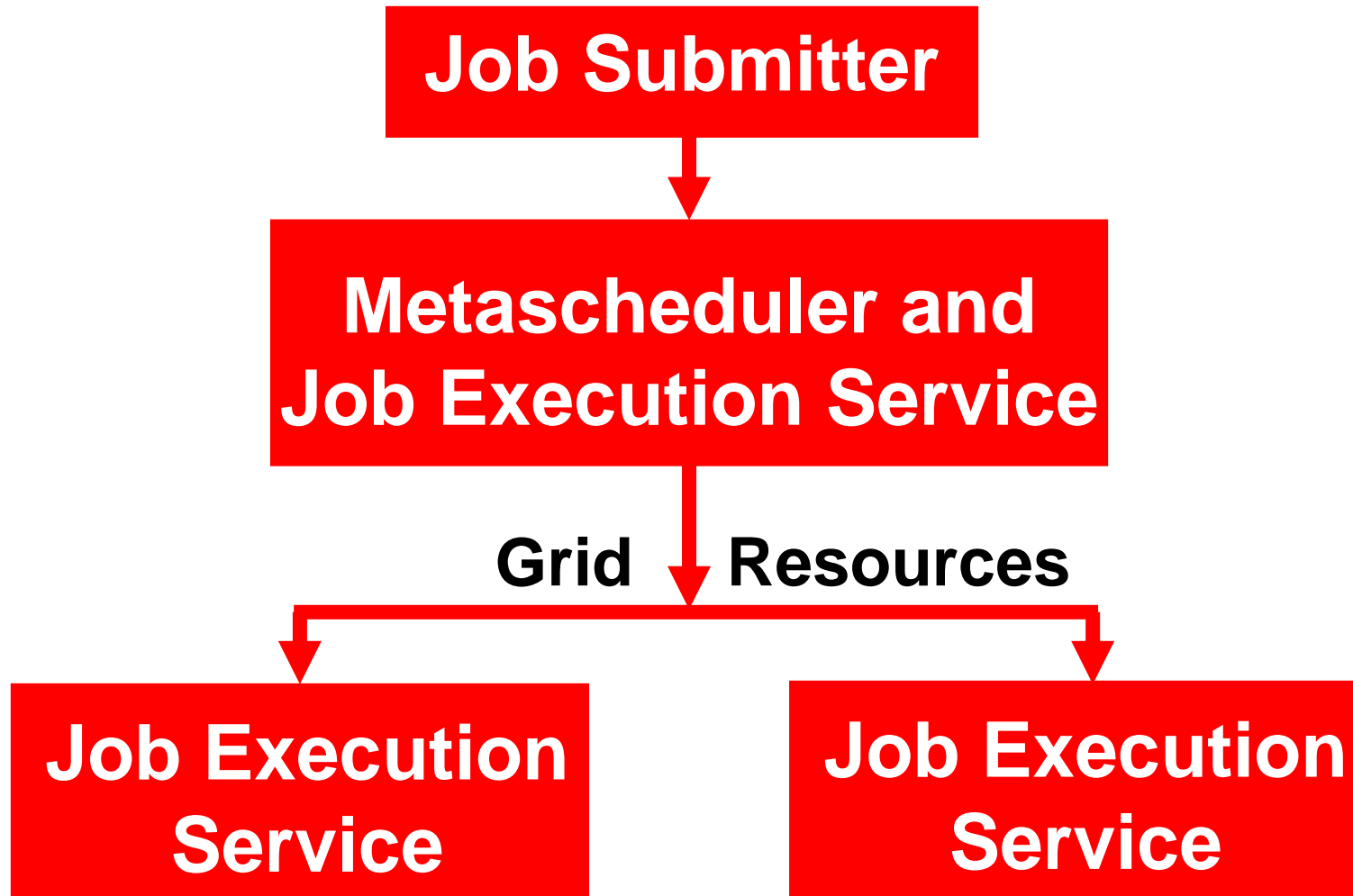


## Cons of stand-alone metascheduler

---

- The job submission client must be aware of the MS, and before submitting the job, invoke the MS service, then insert the resource selected by MS in the job description
- Resource management becomes more complex
  - Resource allocation is done by MS
  - Job execution on that resource is done by the Job Execution service
  - MS does not know if job execution has succeeded, unless an additional component is built for this purpose
  - Thus, either allocations made by MS must expire or a mechanism to inform MS about unused allocations must be developed

# Metascheduler as Extension of Job Execution



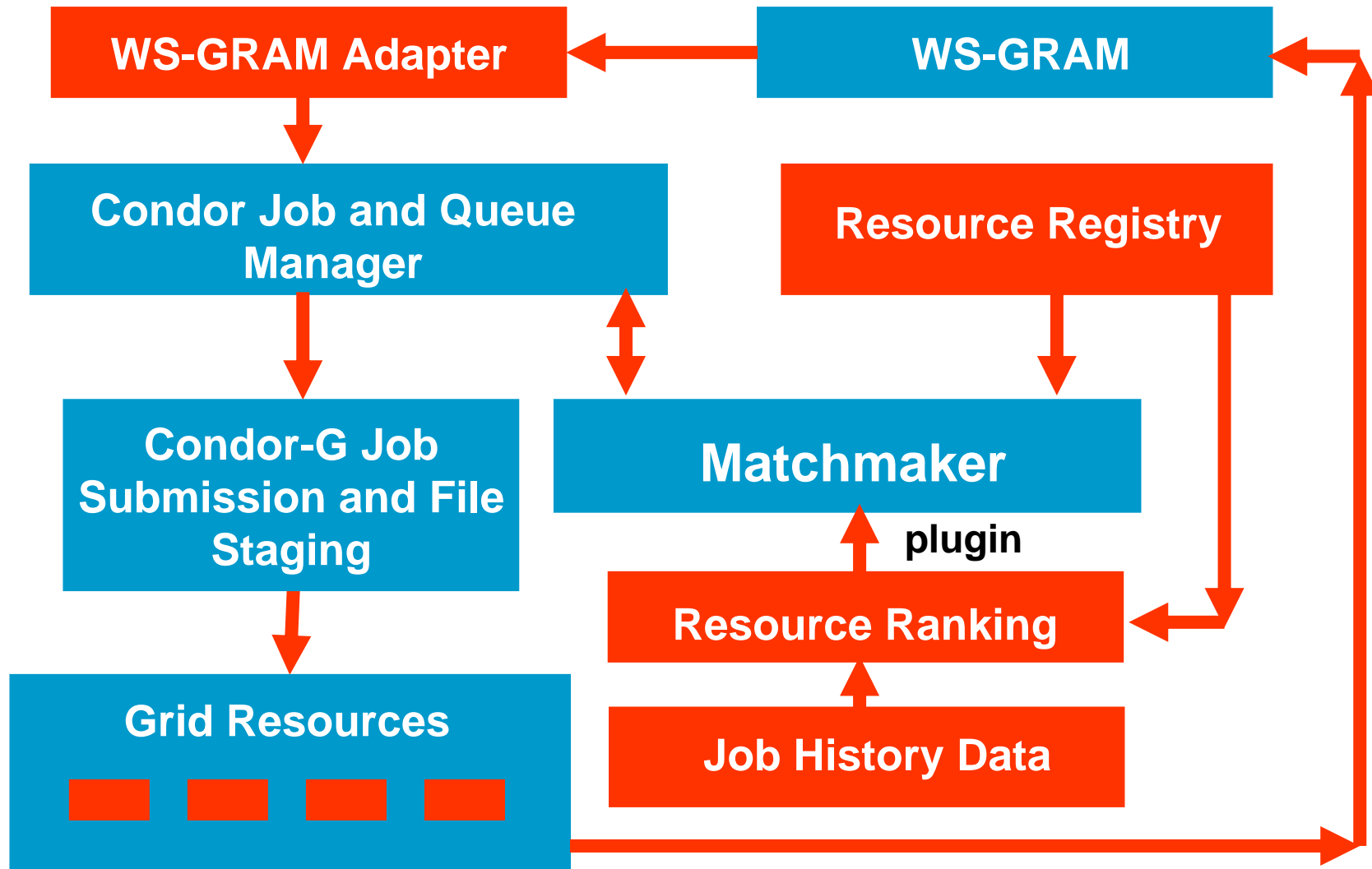


# Metascheduler as Extension of Job Execution

---

- The job submission client sends the job to the Job Execution service which performs scheduling behind-the-scene
- The Job Execution services handles
  - Resource allocation
  - Job dispatching to the resource
  - Exception handling: resource de-allocation in case job dispatching fails
  - Since WS-GRAM strips off file staging commands, special attention must be given to file staging

# Globus with Metascheduling





# Changed or New modules

---

- GRAM adapter: converts GRAM job description into Condor-G job description
  - This includes generating the Requirements and Rank expressions used in matchmaking and handling file staging
- Resource ranking plug-in: call-out invoked by the matchmaker to evaluate the Rank
  - Allows to compute a rank that depends on multiple resources
  - Substitute for computing Rank in the matchmaker, which evaluates a pair (job, Resource) so Rank can be a function of only the current resource
- Resource registry and job history
  - Resource registry contains detailed resource information
  - Job history database allows to predict the turnaround time of a new job



# Regular Globus Job Specification

---

**<job>**

**<executable>/bin/hostname</executable>**

**<directory>\${GLOBUS\_USER\_HOME}</directory>**

**<argument>-f</argument>**

**<stdout>\${GLOBUS\_USER\_HOME}/hostname.out</stdout>**

**<stderr>\${GLOBUS\_USER\_HOME}/hostname.err</stderr>**

**<jobType>single</jobType>**

**<fileStageOut>**

**<transfer>**

**<sourceUrl>file:///\${GLOBUS\_USER\_HOME}/hostname.OUT</sourceUrl>**

**<destinationUrl>**

**gsiftp://g.lrz.de/home/gabriel/hostname.OUT**

**</destinationUrl>**

**</transfer>**

**</fileStageOut>**

**</job>**



# Job Specification for MS Stage-out (1)

---

**<job>**

**<executable>/bin/hostname</executable>**

**<directory>\${GLOBUS\_USER\_HOME}</directory>**

**<argument>-f</argument>**

**<stdout>\${GLOBUS\_USER\_HOME}/hostname.out</stdout>**

**<stderr>\${GLOBUS\_USER\_HOME}/hostname.err</stderr>**

**<jobType>single</jobType>**

**<fileStageOut>**

**<transfer>**

**<sourceUrl>file:///\${GLOBUS\_USER\_HOME}/hostname.OUT</sourceUrl>**

**<destinationUrl>gsiftp://g.lrz.de/home/gabriel/hostname.OUT</destinationUrl>**

**</transfer>**

**</fileStageOut>**

**<extensions>**

**<universe>grid</universe>**

**<fileStageOut>**

**<sourceUrl>hostname.OUT</sourceUrl>**

**<fileStageOut>**

**</extensions>**

**</job>**



# Job Specification for MS Stage-out (2)

---

**<job>**

**<executable>/bin/hostname</executable>**

**<directory>\${GLOBUS\_USER\_HOME}</directory>**

**<argument>-f</argument>**

**<stdout>\${GLOBUS\_USER\_HOME}/hostname.out</stdout>**

**<stderr>\${GLOBUS\_USER\_HOME}/hostname.err</stderr>**

**<jobType>single</jobType>**

**<extensions>**

**<universe>grid</universe>**

**<fileStageOut>**

**<transfer>**

**<sourceUrl>file:///\${GLOBUS\_USER\_HOME}/hostname.OUT</sourceUrl>**

**<destinationUrl>**

**gsiftp://g.lrz.de/home/gabriel/hostname.OUT**

**</destinationUrl>**

**</transfer>**

**</fileStageOut>**

**</extensions>**

**</job>**



# Conclusions

---

- A Metascheduler exposed as a smart Job Execution service in an effective and inexpensive approach to providing the needed metascheduling functionality in Globus
- It allows incremental deployment of sophisticated resource selection algorithms
- WS-GRAM was not designed for hierarchical job submission, which requires special techniques for performing file staging