Request Management System

Ph.Charpentier
CERN, LHCb

4th Dirac User Workshop
May 26-28 2014

Introduction

RMS

- The initial Request Management System (RMS)
  - Was designed mostly for dealing with failover for transfers and registration
    - replicateAndRegister, register, removeReplica
  - Evolved with time:
    - More types of subrequests
      - removeFile, forwardDISET, …
    - Used by DM transformations
      - Bulk transfers, removals
      - Many requests, problems of scaling and internal design
- The new RMS (as of v6r10)
  - Complete redesign of requests schema
  - More modular, full state machine
  - Hopefully more scalable, all requests handled by a single type of agent
    - Can have multiple instances

Requests structure

RMS

- Requests
  - They are a set of Operations
  - It has a Status, generated from the Status of Operations
    - Status set artificially `Assigned` while being owned by an Agent
- Operations
  - They have a Type, a Status, possibly additional parameters
  - They may act on Files
- Files
  - In case an Operation acts on a list of files
  - They have a Status

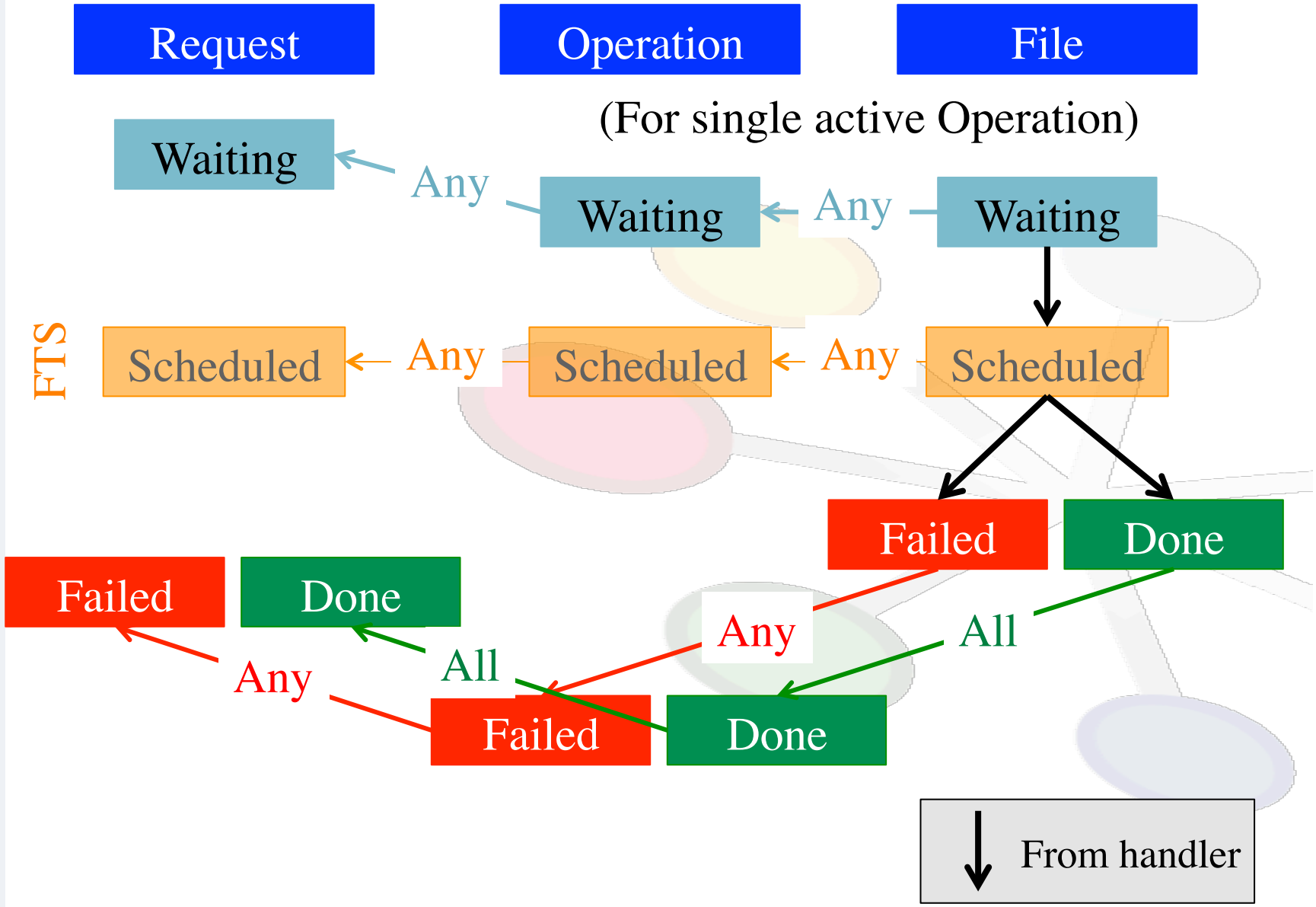
```
Request name='00036569_00014650_job_77936911' ID=728560 Status='Failed' Job=77936911
Created 2014-05-20 09:58:13, Updated 2014-05-20 12:06:54
Owner: '/DC=es/DC=irisgrid/O=ecm-ub/CN=Ricardo-Graciani-Diaz', Group: lhcb_data
  [0] Operation Type='ReplicateAndRegister' ID=1409859 Order=1 Status='Failed'
      SourceSE: CNAF-FAILOVER - TargetSE: GRIDKA-DST - Created 2014-05-20 09:57:56, Updated 2014-05-20 12:06:54
    [01] ID=1543098 LFN='/lhcb/LHCb/Collision12/SWIMSTRIPPINGD02KSKK.MDST/
00036569/0001/00036569_00014650_4.swimstrippingd02kskk.mdst' Status='Failed' Error='No such file or directory'
  [1] Operation Type='RemoveReplica' ID=1409860 Order=2 Status='Queued'
      TargetSE: CNAF-FAILOVER - Created 2014-05-20 09:57:56, Updated 2014-05-20 12:06:54
    [01] ID=1543099 LFN='/lhcb/LHCb/Collision12/SWIMSTRIPPINGD02KSKK.MDST/
00036569/0001/00036569_00014650_4.swimstrippingd02kskk.mdst' Status='Waiting'
```

Request execution

- A single Agent type is in charge of executing requests:
  - `RequestExecutingAgent`
- Operations are executed serially
  - When an Operation is Done, the next one (if any) is executed
    - If no next, the Request is Done
- Execution is delegated to Operation Handlers
  - Mapping between Operation Type and Handler can be defined in the CS
  - Called by an execution engine (`RequestTask`)
  - Easily extendable (new type, new handler)

RMS

State machine



RMS

Current operation handlers

RMS

- ReplicateAndRegister
  - Uses FTS unless otherwise setup (using group owner)
  - In LHCb: lhcb_user doesn't use FTS but directly Replica/ DataManager
- RegisterFile / RegisterReplica
  - Only register in file catalog(s)
- RemoveFile / RemoveReplica
  - Self understanding
- PhysicalRemoval / PutAndRegister / ReTransfer
  - Implemented, not used by LHCb
- ForwardDISET
  - Make any DISET call (arguments passed as a blob)
  - No "Files"

- Extensions for LHCb
  - LogUpload

Miscellaneous

- Only recoverable errors are retried
  - By default 256 retries before Failed
- FTS transfers (Chris' talk)
  - Handled by a separate Agent FTSAgent
  - Uses the "Scheduled" status for passing requests
- Job callback
  - If a job is associated to a Request, its status can be modified when the Request is in a final status
    - For the time being only Completed to Done/Failed
- More interaction scripts
  - Show, create, fix requests
  - No web portal interface yet (to be defined what is needed)
- Still to do:
  - Last update timestamp update for all items
    - Should only be the one that changed
  - Error string overwritten by "Max attempts reached"
  - Request optimisation (grouping files into Operations)

RMS

LHCb

Future usage

RMS

- Moving the staging to RMS
  - Staging uses a similar logic, but was developed independently
  - Plan to move staging to RMS
  - Opens option to use FTS3 for staging (see Chris' talk)
  - Can use the job feedback for moving from Staging to Waiting (or Failed)
- Other extensions
  - Very simple to add any new type of operations
    - Just create a new Operation handler!

Conclusion

RMS

- The old legacy RequestManagementSystem was replaced by a better designed system
  - As of v6r10
  - In full use by LHCb for several months
  - Tested in production environment with a lot of use cases… and of failure cases!
- The new RMS is more flexible, extendible, reliable than the former one
- Scaling can be achieved easily by using multiple RequestExecutingAgent instances
- FTS handled by a separate agent