New stager commands
Details and anatomy

CASTOR external operation meeting
CERN - Geneva
14/06/2005

Sebastien Ponce, CERN-IT

Outline

- API
  - Common part with old stager
    - stage_get, stage_put, stage_qry
  - New commands
    - prepare_to_get, prepare_to_put, putDone
  - Future
    - getNext
- Command Line
    - rfcp, stager_qry, stager_get, stager_put, stager_putDone,
- Anatomy of a request
  - get + recall
  - put + migration

stager_get API

- stage_get Stages one file from CASTOR, and schedules the data access. The file is opened Read only

```
int stage_get(const char * userTag,
              const char * protocol,
              const char * filename,
              struct stage_io_fileresp ** response,
              char ** requestId,
              struct stage_options * opts)
```

  - userTag A string chosen by user to group requests
  - protocol The protocol requested to access the file
  - filename The CASTOR filename
  - response fileresponse structure
  - requestId Reference number to be used by the client to look up his request in the castor stager.
  - opts CASTOR stager specific options

stage_put API

- stage_put stages one file into CASTOR, and schedules the data access

```
int stage_put(const char * userTag,
              const char * protocol,
              const char * filename,
              mode_t mode,
              u_signed64 size,
              struct stage_io_fileresp ** response,
              char ** requestId,
              struct stage_options * opts)
```

  - userTag A string chosen by user to group requests
  - protocol The protocol requested to access the file
  - filename The CASTOR filename
  - mode The mode in which the file is to be opened
  - size The expected filesize of the file that is going to be writen (or 0, in which case the stager will take its default)
  - response fileresponse structure
  - requestId Reference number to be used by the client to look up his request in the castor stager.
  - opts CASTOR stager specific options

stage_qry API

- stage_filequery Returns summary information about the files in the CASTOR stager

```
int stage_filequery(struct stage_query_req * requests,
                    int     nbreqs,
                    struct  stage_filequery_resp ** responses,
                    int *   nbresps,
                    struct  stage_options * opts)
```

  - requests Pointer to the list of file requests
  - nbreqs Number of file requests in the list
  - responses List of file responses, created by the call itself
  - nbresps Number of file responses in the list
  - opts CASTOR stager specific options

Outline

- API
  - Common part with old stager
    - stage_get, stage_put, stage_qry
  - New commands
    - prepare_to_get, prepare_to_put, putDone
  - Future
    - getNext
- Command Line
    - rfcp, stager_qry, stager_get, stager_put, stager_putDone,
- Anatomy of a request
  - get + recall
  - put + migration

stage_prepareToGet API

- stage_prepareToGet stages the files from CASTOR , but does not schedule the file access

```
int stage_prepareToGet
        (const char * userTag,
         struct stage_prepareToGet_filereq * requests,
         int nbreqs,
         struct stage_prepareToGet_fileresp ** responses,
         int * nbresps,
         char ** requestId,
         struct stage_options * opts)
```

  - userTag A string chosen by user to group requests
  - requests Pointer to the flist of file requests
  - nbreqs Number of file requests in the list
  - responses List of file responses, created by the call itself
  - nbresps Number of file responses in the list
  - requestId Reference number to be used by the client to look up his request in the castor stager
  - opts CASTOR stager specific options

stage_prepareToPut API

- stage_prepareToPut Reserve space so as to put files in CASTOR, but do not schedule access to those files

```
int stage_prepareToPut
      (const char * userTag,
       struct stage_prepareToPut_filereq * requests,
       int nbreqs,
       struct stage_prepareToPut_fileresp ** responses,
       int * nbresps,
       char ** requestId,
       struct stage_options * opts)
```

  - userTag A string chosen by user to group requests
  - requests Pointer to the list of file requests
  - nbreqs Number of file requests in the list
  - responses List of file responses, created by the call itself
  - nbresps Number of file responses in the list
  - requestId Reference number to be used by the client to look up his request in the castor stager.
  - opts CASTOR stager specific options

stage_putDone API

- stage_putDone Changes the status of the files, indicating that the put request is successfully done

```
int stage_putDone(char * putRequestId,
     struct stage_filereq * requests,
     int nbreqs,
     struct stage_fileresp ** responses,
     int * nbresps,
     char ** requestId,
     struct stage_options * opts)
```

  - putRequestId ID of the related prepare to put request
  - requests Pointer to the list of file requests
  - nbreqs Number of file requests in the list
  - responses List of file responses, created by the call itself
  - nbresps Number of file responses in the list
  - requestId Reference number to be used by the client to look up his request in the castor stager.
  - opts CASTOR stager specific options

Outline

- API
  - Common part with old stager
    - stage_get, stage_put, stage_qry
  - New commands
    - prepare_to_get, prepare_to_put, putDone
  - Future
    - getNext
- Command Line
    - rfcp, stager_qry, stager_get, stager_put, stager_putDone,
- Anatomy of a request
  - get + recall
  - put + migration

stage_getNext API

- stage_getNext Schedules access to the next file in the prepateToGet request that is ready to be accessed

```
int stage_getNext(const char * reqId,
                  struct stage_io_fileresp ** response,
                  struct stage_options * opts)
```

  - reqId ID of the stage_prepareToGetRequest
  - response The location of the file
  - opts CASTOR stager specific options
- this allows optimization of the scanning of a large number of files distributed among many tapes
- the system takes care of always pre-staging few files in advance, but not all of them

Outline

- API
  - Common part with old stager
    - stage_get, stage_put, stage_qry
  - New commands
    - prepare_to_get, prepare_to_put, putDone
  - Future
    - getNext
- Command Line
    - rfcp, stager_qry, stager_get, stager_put, stager_putDone,
- Anatomy of a request
  - get + recall
  - put + migration

command line

- rfcp [-s size] [-v2] filename1 filename2
  - for simple get, simple put
- stager_get [-M hsmfile [-M ...]] [-S svcClass] [-U usertag] [-f protocol]
  - actually a prepareToGet. Then use rfcp to transfer data
- stager_prepareToPut [-P protocol] [-U usertag] [-S svcClass] [-M hsmfile [-M ...]]
  - then use rfcp to transfer data and putDone to unlock the file
  - Note that the file will not be migrated before putDone, that it will never be reset before putDone and that multiple simultaneous put are allowed
- stager_putDone [-R requestid][-M hsmfile [-M ...]]
  - ends a stager_put session

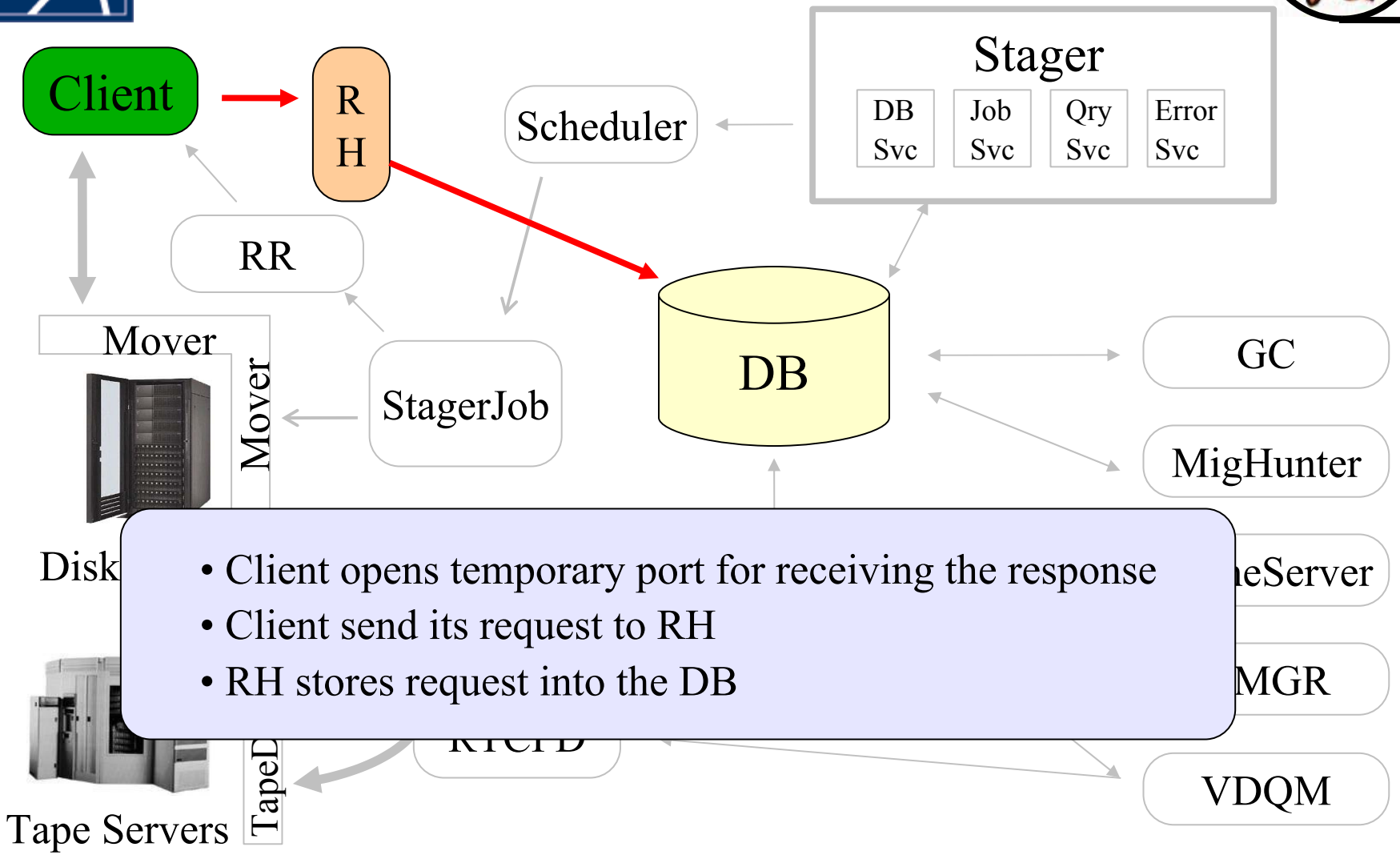
command line

- stager_qry [-M hsmfile|-F fileid|-U usertag|-R requestid]
  - queries for the status of (a) files(s). Three selections criteria are possible
    - by file (name or fileid)
    - by userTag
    - by requestId

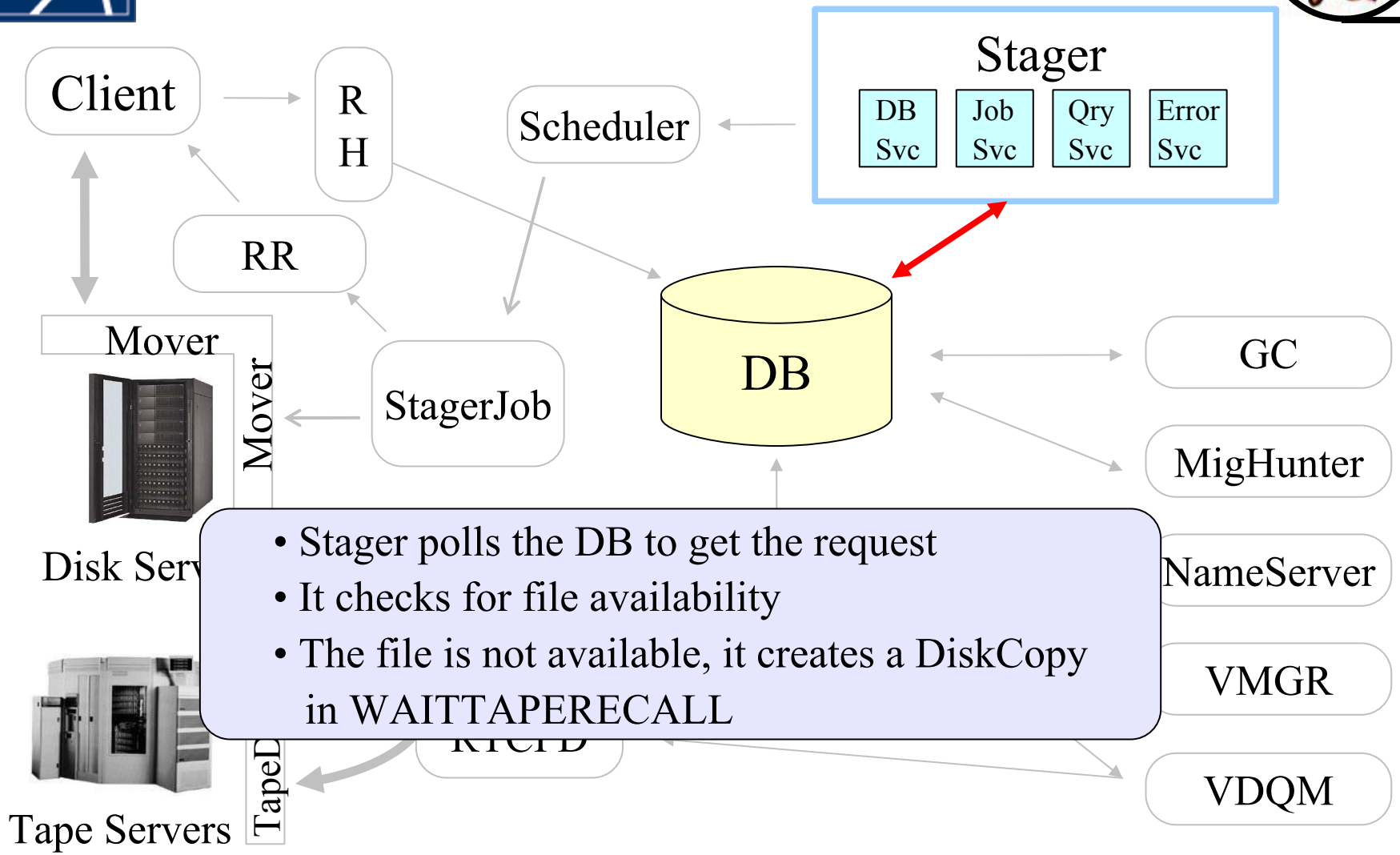
Outline

- API
  - Common part with old stager
    - stage_get, stage_put, stage_qry
  - New commands
    - prepare_to_get, prepare_to_put, putDone
  - Future
    - getNext
- Command Line
    - rfcp, stager_qry, stager_get, stager_put, stager_putDone,
- Anatomy of a request
  - ~~get + recall~~
  - put + migration

stager_get (1)

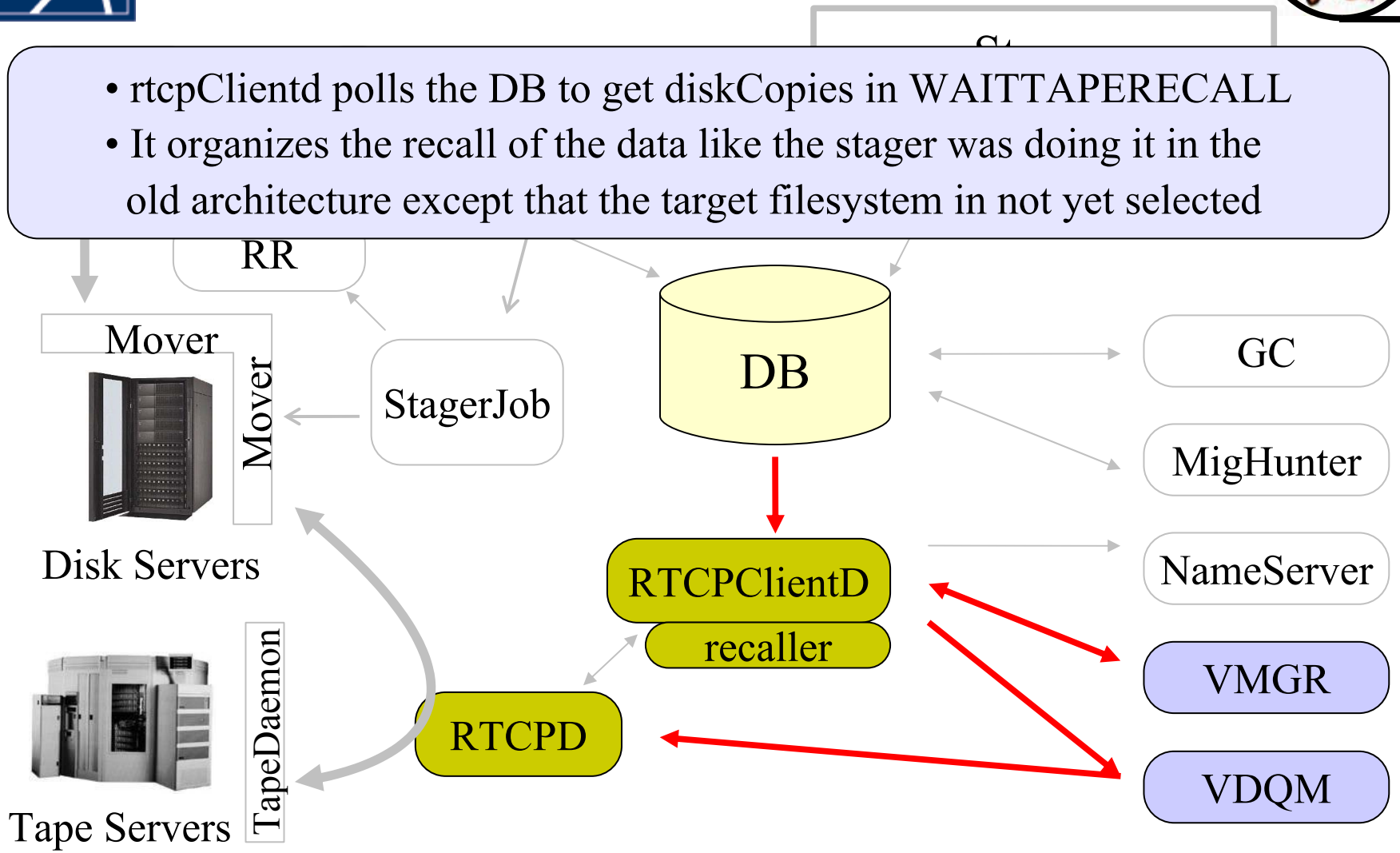

Client
R
H
Scheduler
Stager
DB Svc
Job Svc
Qry Svc
Error Svc
RR
Mover
Mover
StagerJob
DB
GC
MigHunter
Disk
eServer
MGR
VDQM
TapeD
Tape Servers
• Client opens temporary port for receiving the response
• Client send its request to RH
• RH stores request into the DB

stager_get (2)

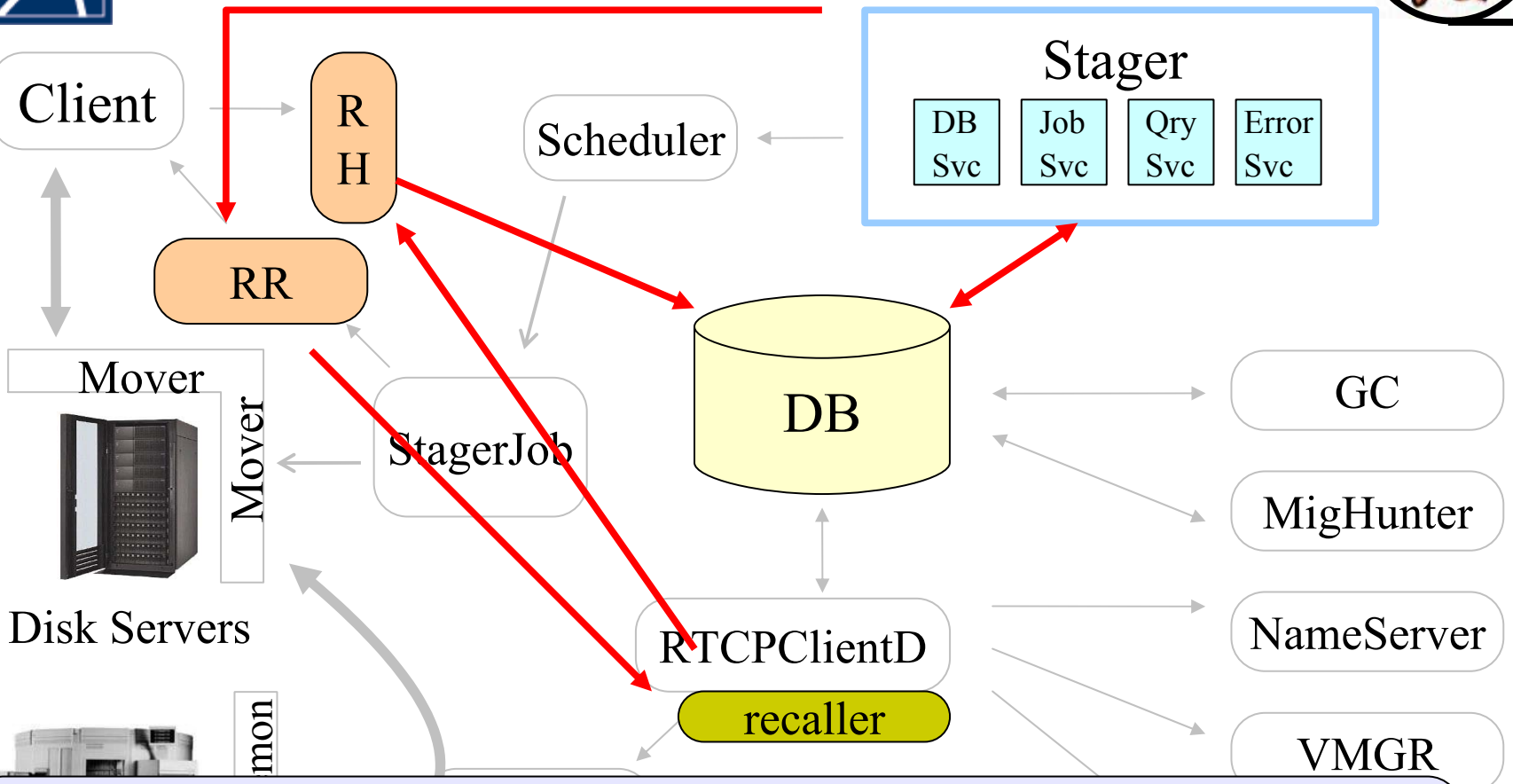

Client
R
H
Scheduler
Stager
DB Svc
Job Svc
Qry Svc
Error Svc
RR
Mover
Mover
StagerJob
DB
GC
MigHunter
Disk Serv
NameServer
VMGR
Tape Servers
TapeD
VDQM
• Stager polls the DB to get the request
• It checks for file availability
• The file is not available, it creates a DiskCopy in WAITTAPERECALL

stager_get (3)

- rtcpClientd polls the DB to get diskCopies in WAITTAPERECALL
- It organizes the recall of the data like the stager was doing it in the old architecture except that the target filesystem in not yet selected



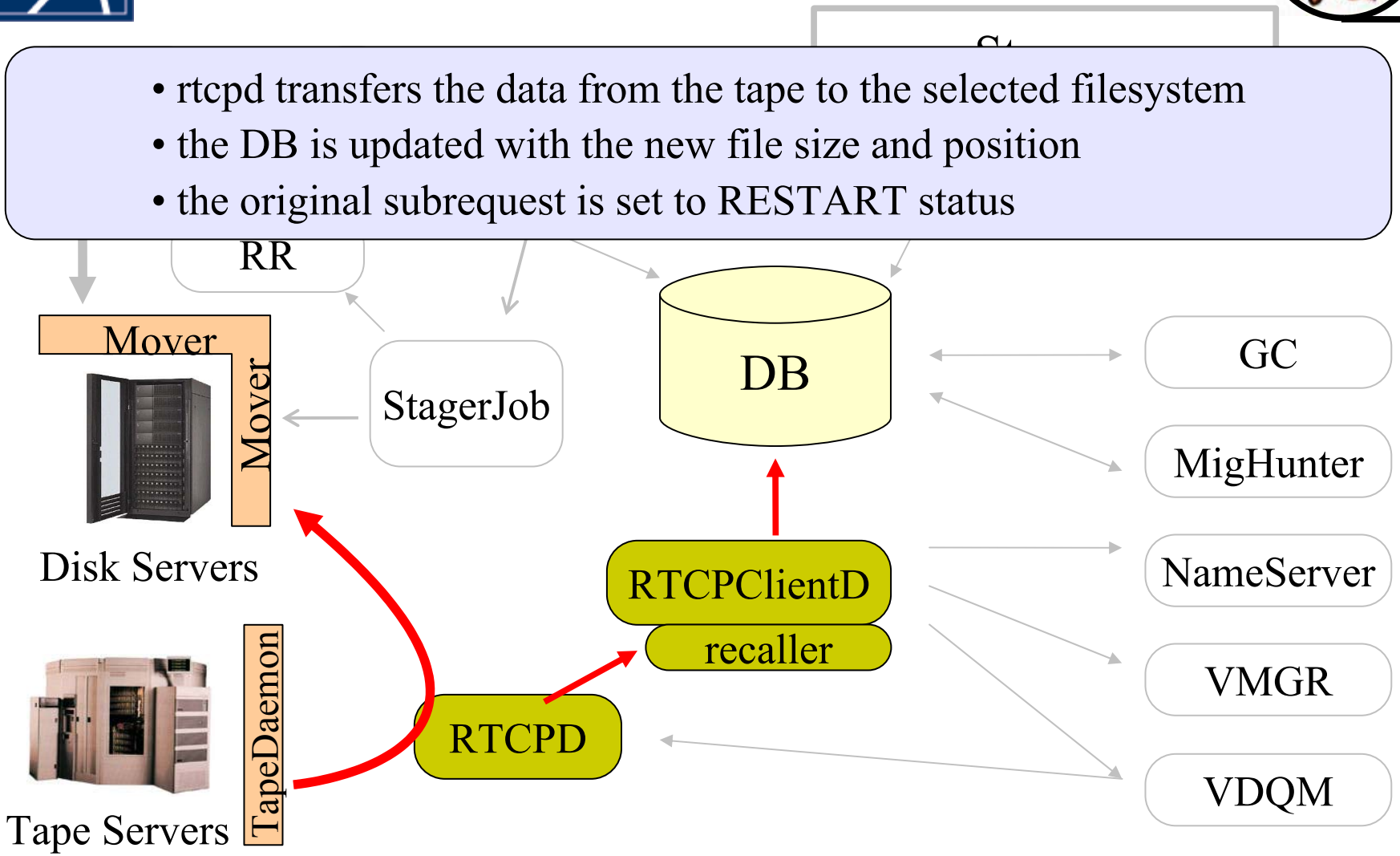
stager_get (4)



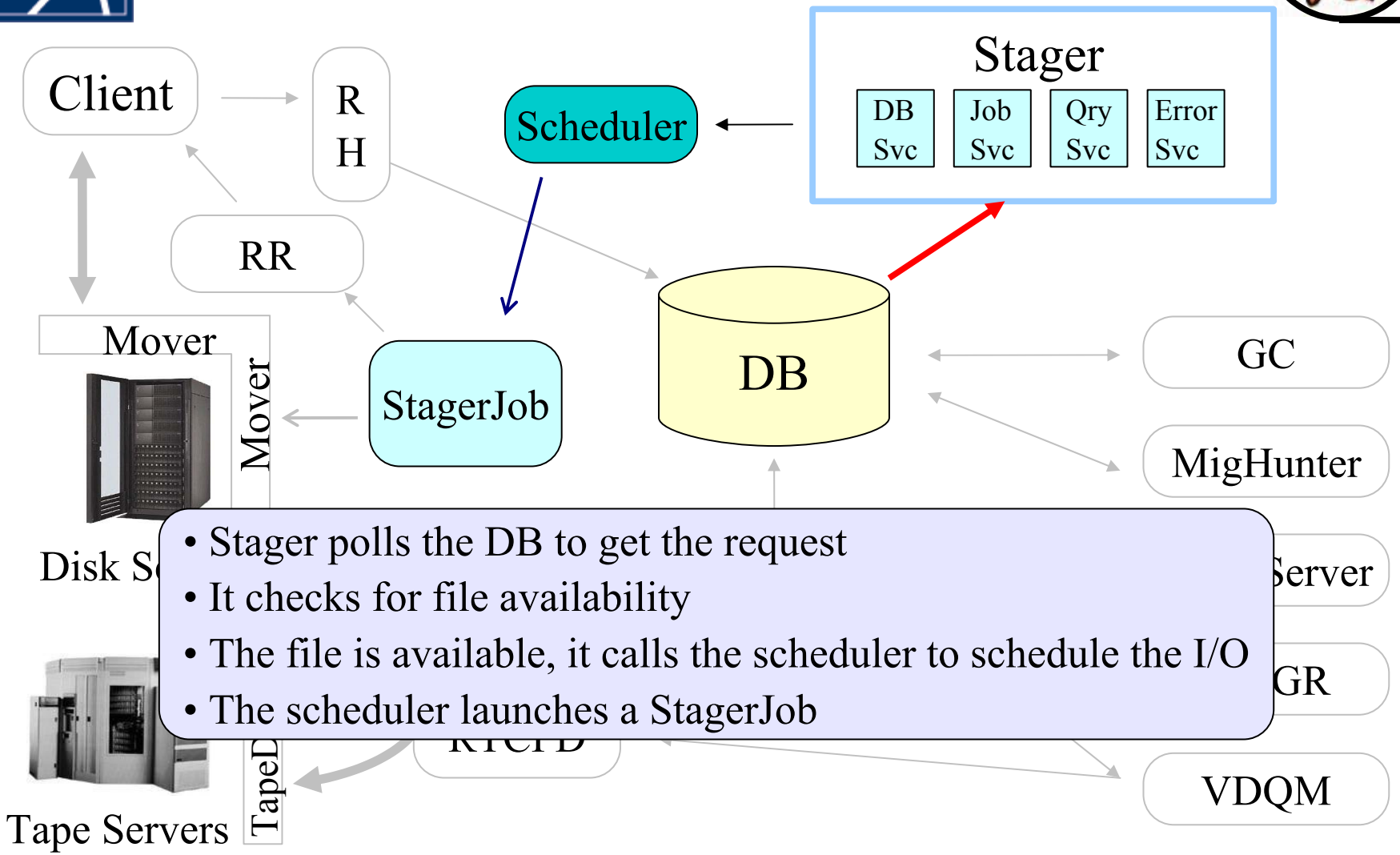
- recaller sends a request to the stager in order to know where to put the file
- the request goes through the usual way : Request Handler, DB, stager (job service), Request Replier

stager_get (5)

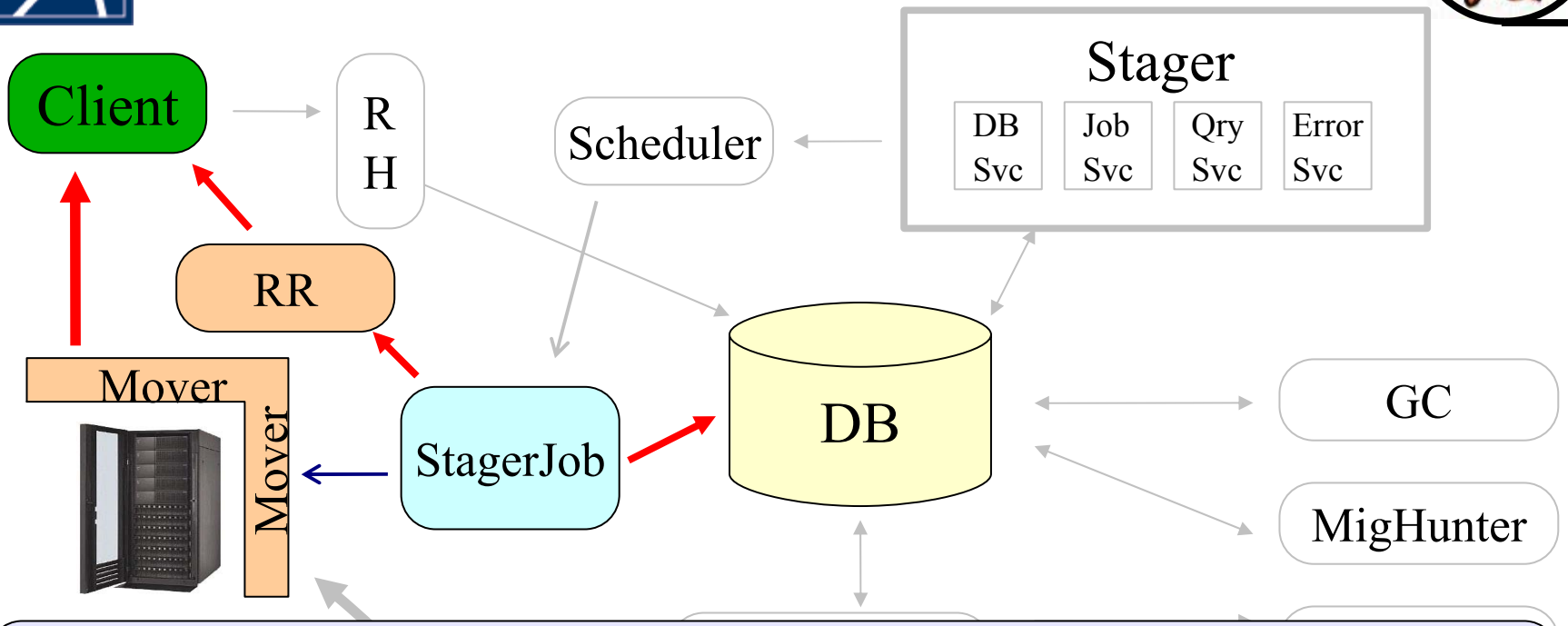
- rtcpd transfers the data from the tape to the selected filesystem
- the DB is updated with the new file size and position
- the original subrequest is set to RESTART status



stager_get (6)


Client
R
H
Scheduler
Stager
DB Svc
Job Svc
Qry Svc
Error Svc
RR
Mover
Mover
StagerJob
DB
GC
MigHunter
Disk S
Server
GR
TapeD
VDQM
Tape Servers
• Stager polls the DB to get the request
• It checks for file availability
• The file is available, it calls the scheduler to schedule the I/O
• The scheduler launches a StagerJob

stager_get (7)

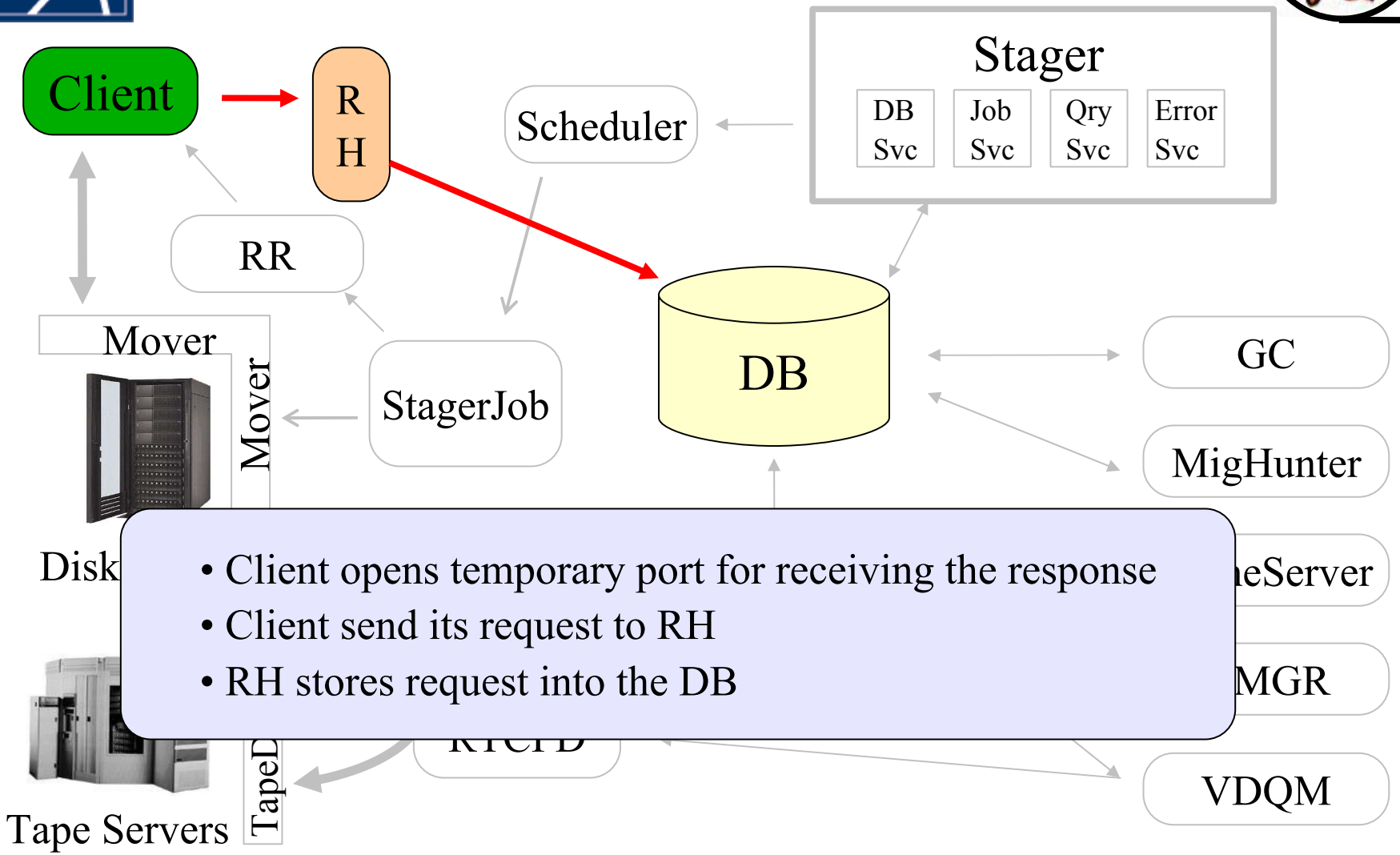


- the StagerJob launches the right mover corresponding to the client request (note that the scheduler takes available movers into account)
- it answers to the client, giving to it the machine and port where to contact the mover
- data is transfered
- DB is updated and cleaned up

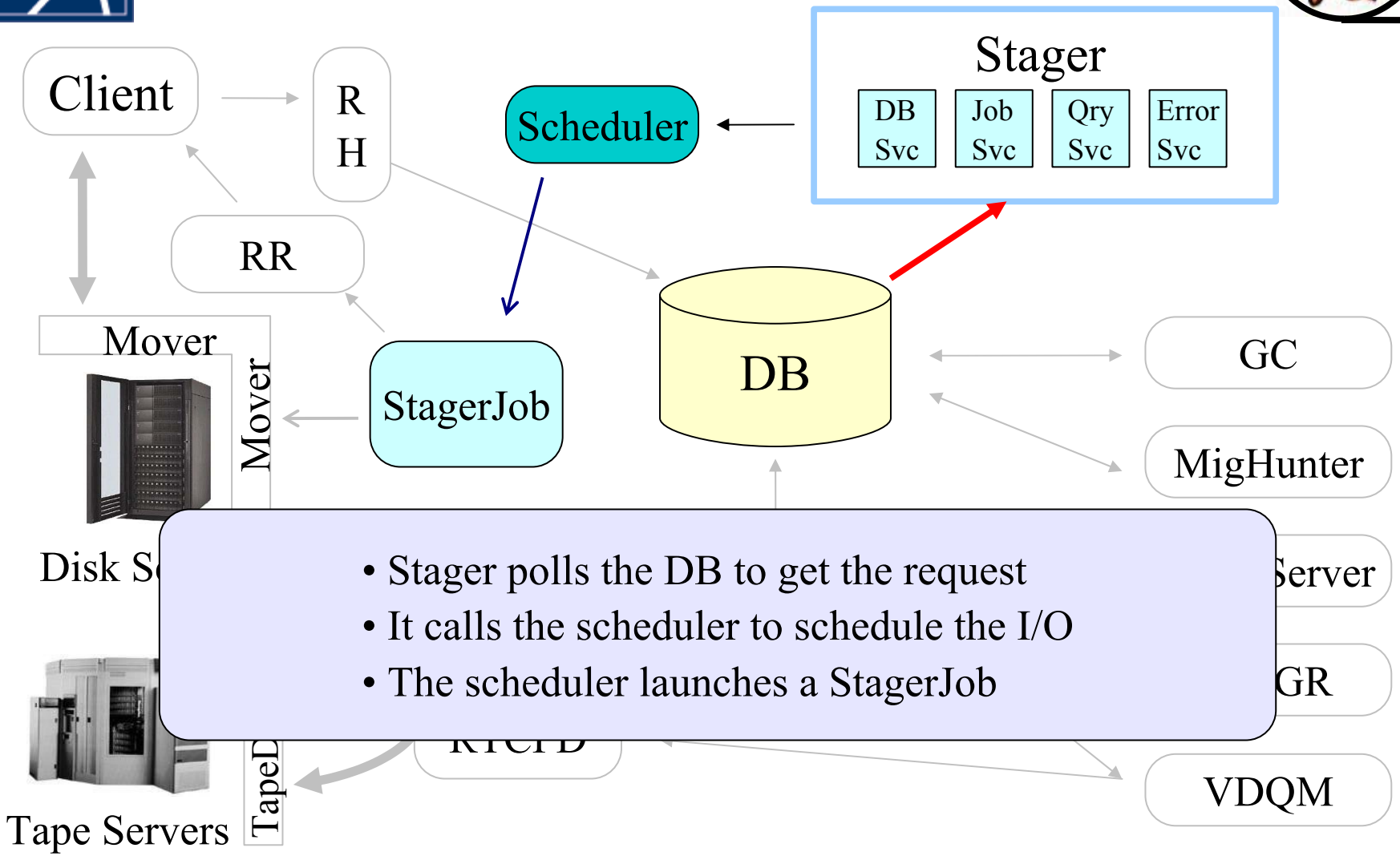
Outline

- API
  - Common part with old stager
    - stage_get, stage_put, stage_qry
  - New commands
    - prepare_to_get, prepare_to_put, putDone
  - Future
    - getNext
- Command Line
    - rfcp, stager_qry, stager_get, stager_put, stager_putDone,
- Anatomy of a request
  - get + recall
  - put + migration

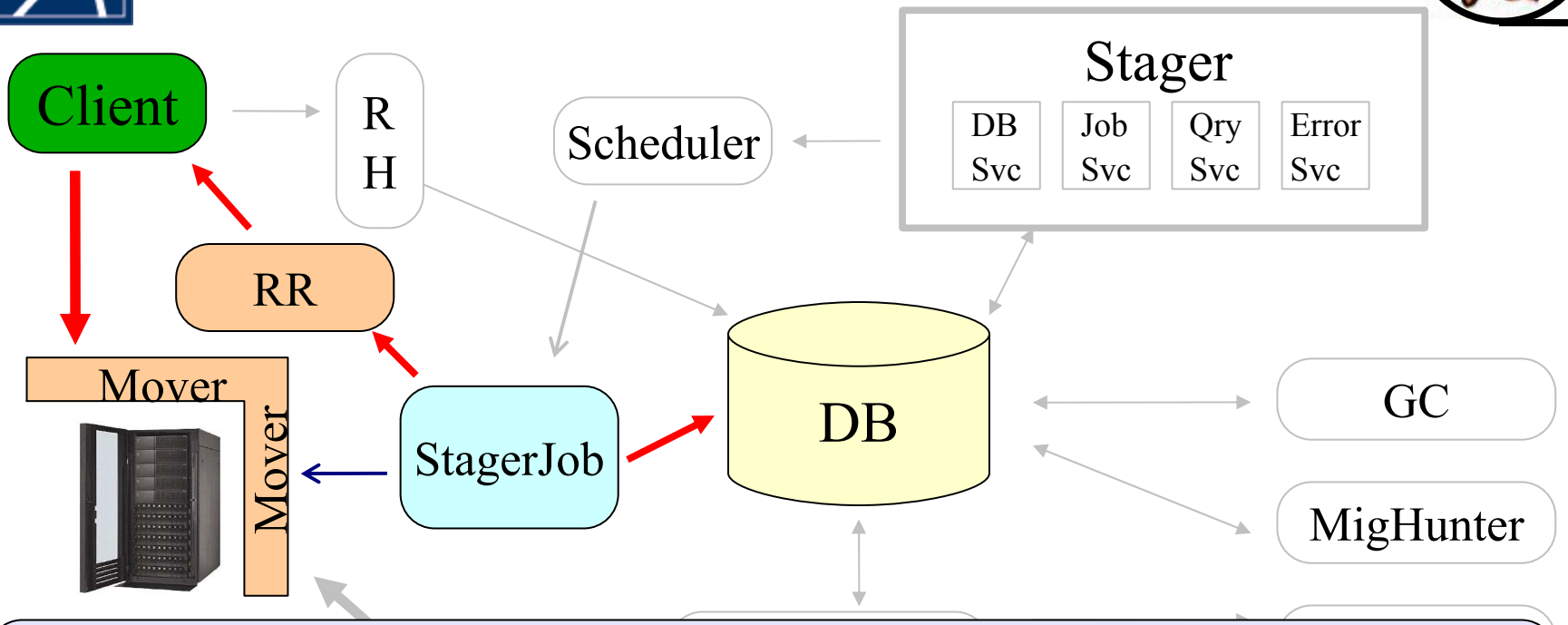
stager_put (1)


Client
R
H
Scheduler
Stager
DB Svc
Job Svc
Qry Svc
Error Svc
RR
Mover
Mover
StagerJob
DB
GC
MigHunter
Disk
eServer
MGR
VDQM
TapeD
Tape Servers
• Client opens temporary port for receiving the response
• Client send its request to RH
• RH stores request into the DB

stager_put (2)

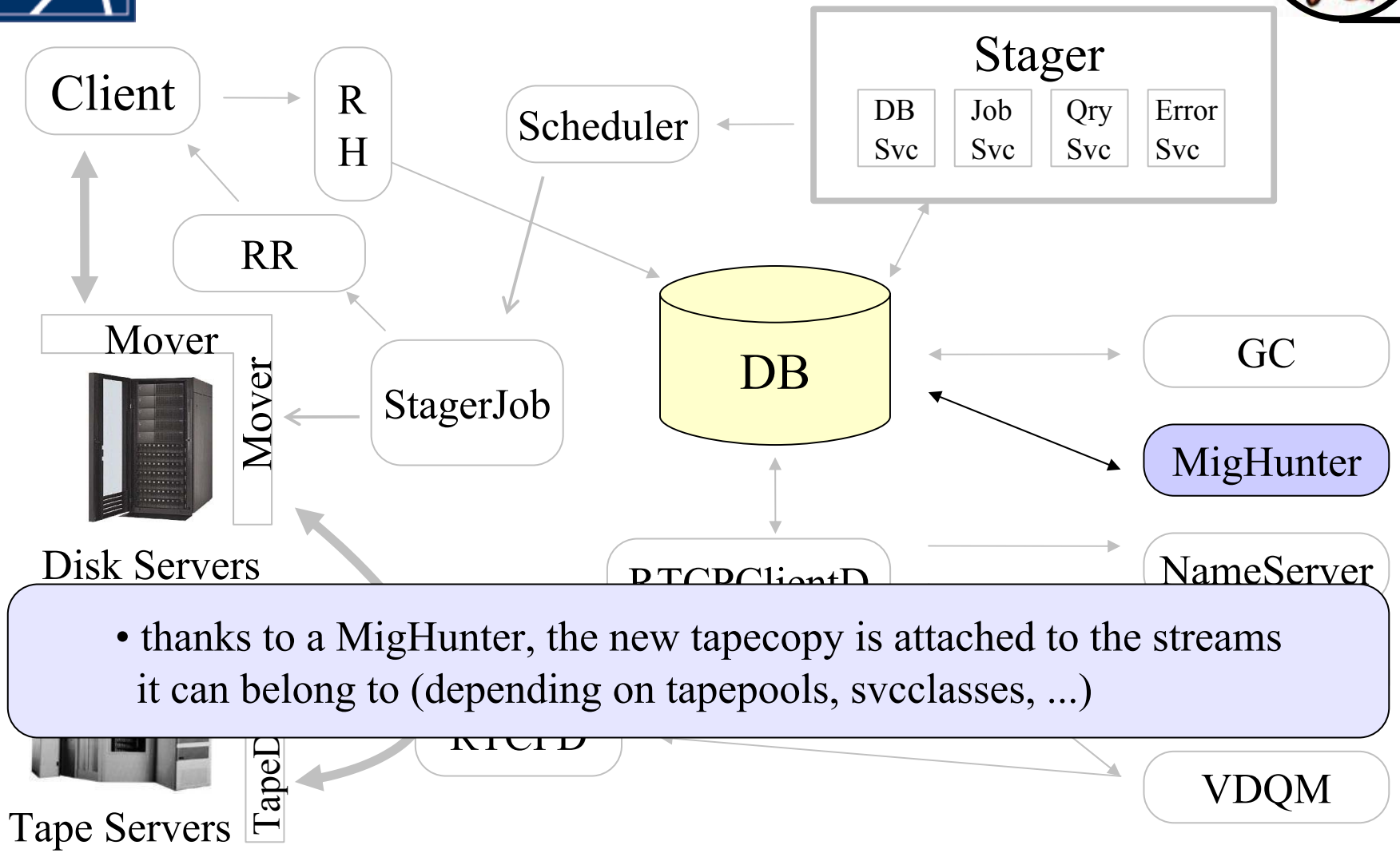

Client
R
H
Scheduler
Stager
DB Svc
Job Svc
Qry Svc
Error Svc
RR
Mover
Mover
StagerJob
DB
GC
MigHunter
Disk S
Server
GR
TapeD
VDQM
Tape Servers
• Stager polls the DB to get the request
• It calls the scheduler to schedule the I/O
• The scheduler launches a StagerJob

stager_put (3)



- the StagerJob launches the right mover corresponding to the client request (note that the scheduler takes available movers into account)
- it answers to the client, giving to it the machine and port where to contact the mover
- data is transfered
- DB is updated with the file size and the diskcopy is set in CANBEMIGR and one or many TapeCopies are created

stager_put (4)



- thanks to a MigHunter, the new tapecopy is attached to the streams it can belong to (depending on tapepools, svcclasses, ...)

stager_put (5)

- rtcpclientd will launch a migrator
- this one asks the DB for the next migration candidate
- the DB takes the best candidate in the stream (based on filesystems availability)
- the file is written to tape and the DB updated