CASTOR 2

Introduction and overview

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Sebastien Ponce, DM group meeting, January 15th 2008

Outline

- CASTOR2 context
  - Scope and constraints
  - History
    - SHIFT, CASTOR 1
  - Requirements
- CASTOR 2 architecture overview
  - blocks and components
  - practical example : lifecycle of get/put requests
  - extra components
- Technology choices
  - about languages
  - UML and the code generation
  - DB centric

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Outline

- CASTOR2 context
  - Scope and constraints
  - History
    - SHIFT, CASTOR 1
  - Requirements
- CASTOR 2 architecture overview
  - blocks and components
  - practical example : lifecycle of get/put requests
  - extra components
- Technology choices
  - about languages
  - UML and the code generation
  - DB centric

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Scope and constraints

- Provide a managed storage service for all physics data at CERN
  - Transparent tape media management
  - Automated disk cache management
  - Unique global name space
- Assure that CERN can fulfill the Tier-0 and Tier-1 storage requirements for the LHC experiments
  - Central Data Recording (CDR)
  - Data reconstruction
  - Data export to Tier-1 centers
- Strive to meet the CERN Tier-2 requirements
  - The exact requirements and scale are unknown but CASTOR should prove to integrate well with other services that may be part of an analysis facility
    - E.g. xrootd interface requested
- CASTOR2 assumes backend mass-storage
  - Not designed to work as a stand-alone disk cache solution for Tier-2 institutes

SHIFT

- **S**calable **H**eterogeneous **I**ntegrated **F**acili**T**y
  - Started as a project (HOPE) between the OPAL experiment and the CN division in early 90's
  - Main authors: Jean-Philippe Baud, Fabrizio Cane, Frederic Hemmer, Eric Jagel, Ashok Kumar, Gordon Lee, Les Robertson, Ben Segal, Antoine Trannoy
- All user file access on disk. No direct tape access
  - The idea of tape staging at CERN dates back to the early 70's
- Components
  - Stager daemon (stgdaemon) managing the disk pool
  - Remote Tape Copy (RTCOPY)
  - Remote File IO (RFIO)
  - Tape allocation and control (tpdaemon)
    - Label processing, load, unload, positioning
    - Operator interface for manual mounting
    - Interface to robotic mounting
  - Tape Management System (TMS)
    - Logical volume repository
- Users access files by Tape volume (VID) + tape file sequence number (FSEQ) → flat namespace: stagein –V EK1234 –q 35 …
  - The experiments normally had their own catalogue on top (e.g. FATMEN)
- CERN was awarded the 21st Century Achievement Award by Computerworld in 2001

SHIFT limitations

- Data rate: more than 10MB/s per stream is difficult to achieve
- Stager catalog does not really scale over 10,000 files
- SHIFT does not support many concurrent accesses
- No automatic allocation of tapes
- No easy access to files by name without an external package
- automatic migration/recall of files is not available

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Alternative solutions

- Since the mid-90's CERN had been looking and testing alternative solutions to SHIFT
  - OSM
  - ADSM (now TSM):
  - HPSS
  - Eurostore
- Only HPSS was run in production for 3 years (1998 – 2001)

CASTOR1

CERN IT Department

- Cern Advanced STORage manager
  - Project started in 1999 to address the immediate needs (NA48, Compass) and provide a base that would scale to meet the LHC requirements
  - Managed storage: tapes hidden from the users
  - Main authors: Jean-Philippe Baud, Fabien Collin, Jean-Damien Durand, Olof Bärring
- Components
  - Stager daemon enhancements
    - Automatic tape migration added
  - Remote File IO (RFIO) supports data streaming
  - Volume Manager (VMGR) replaces TMS
  - Name server (Cns) provides a CASTOR name space
  - Tape Volume and Drive Queue service (VDQM)
  - Tape repack automated media migration
- Users access file by their CASTOR file names

  stagein –M /castor/cern.ch/user/...

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

CASTOR1 limitations

- Stager catalogue limitations:
  - Stager unresponsive beyond 200k disk resident files
- Tape migration/recall not optimal
  - Migration streams are statically formed and ordering among files cannot be changed depending on the load picture
  - Tape recalls request for same tape are not automatically bundled together
  - Large footprint from waiting requests
- No true request scheduling
  - Throttling, load-balancing
  - Fair-share
- Operational issues
  - No unique request identifiers
  - Problem tracing difficult
- Stager code had reached a state where it had become practically un-maintainable
  - Based on the >10 years old SHIFT stager with a long history of patches, hacks and add-ons for CASTOR1

CASTOR2

CERN IT Department

- The original CASTOR plans from 1999 contained a development of a new stager
  - The re-use of the SHIFT stager was temporary
- Project proposed at 2003 CASTOR users' meeting: develop a replacement for the CASTOR1 stager
  - The CASTOR1 stager had already proved to not scale to meet LHC requirements
  - Authors :
    - Originally : Olof Bärring, Ben Couturier, Jean-Damien Durand, Sebastien Ponce
    - Currently : Sebastien Ponce, Giuseppe Lo Presti, Giulia Taurelli, Rosa Garcia Rioja, Dennis Waldron, Steven Murray, Maria Isabel Serrano, Victor Kotlyar, ...

CASTOR2 Proposal

- Pluggable framework rather than total solution
  - True request scheduling
    - delegate the scheduling to a pluggable scheduler, possibly using third party sofware
  - Policy attributes
    - externalize policy engines governing the resource matchmaking
- Restricted access to storage resources to achieve predictable load
  - No random rfiod eating up the resources behind the back of the scheduling system
- Disk server autonomy as far as possible
  - In charge of local resources: file system selection and execution of garbage collection
  - Loosing a server should not affect the rest of the system

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Outline

- CASTOR2 context
  - Scope and constraints
  - History
    - SHIFT, CASTOR 1
  - Requirements
- CASTOR 2 architecture overview
  - blocks and components
  - practical example : lifecycle of get/put requests
  - extra components
- Technology choices
  - about languages
  - UML and the code generation
  - DB centric

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Context View

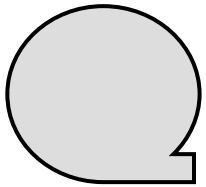
User

Experiment Framework

Grid enabled applications

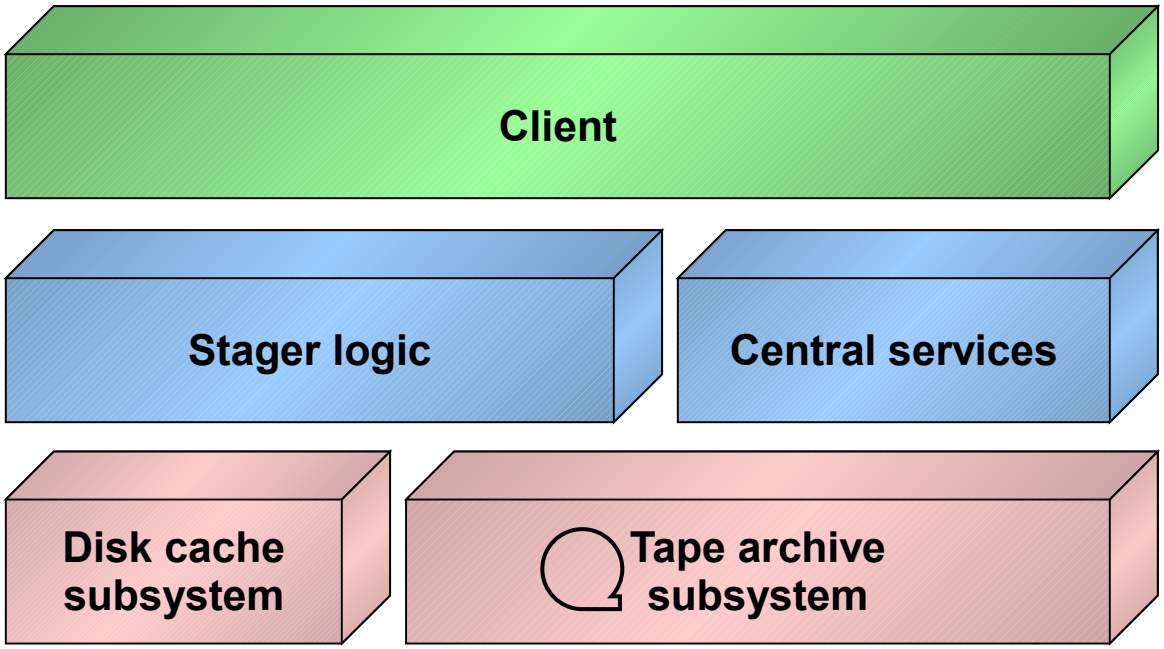
Storage Interface (SRM)

CASTOR



CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

General Picture



CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

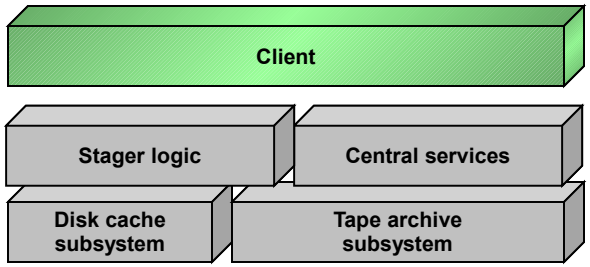
General Picture

- Key requirements
  - Fault tolerance
  - Scalability
  - Transaction-like behavior
- Design features
  - Distributed system
  - Stateless replicated daemons
  - Database-centric architecture
    - State information stored in a RDBMS
    - Rely on the DBMS performances in terms of fault tolerance

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Client deliverables

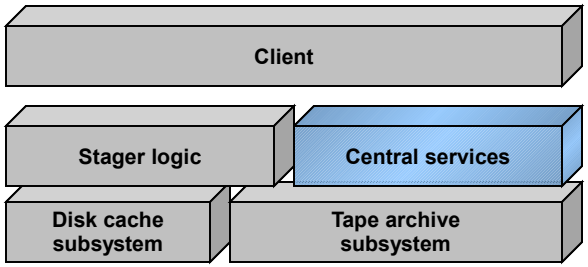
- Command line interface
  - stager commands (**`stager_xxx`**)
  - RFIO commands (**`rfxxx`**)
- Client API
  - only C
  - internal API in C++
- Supported Protocols
  - RFIO: rfio://server:port//castor/cern.ch/...
  - ROOT: root://server:port//castor/cern.ch/...
  - XROOT: rootd://server:port//castor/cern.ch/...
  - GridFTP:
    - internal (via SRM) : gsiftp://server:port//castor/cern.ch/...
    - external : gsiftp://server:port//local/mnt/point/...
- SRM v1 and v2 interfaces



CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Central Services

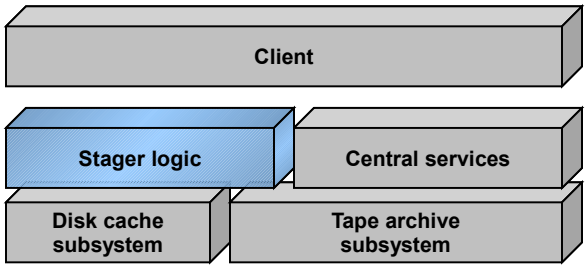
- NameServer
  - Database for the Castor "FileSystem"
  - Stores tape-related info as well



- Volume and Drive Queue Manager (VDQM)
  - Daemon for tape queue management
- Volume Manager (VMGR)
  - Archive of all tapes available in the libraries
- Castor User Privileges (CUPV)
  - Authorization daemon: provides rights to users and admins for tape related operations

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Stager Logic



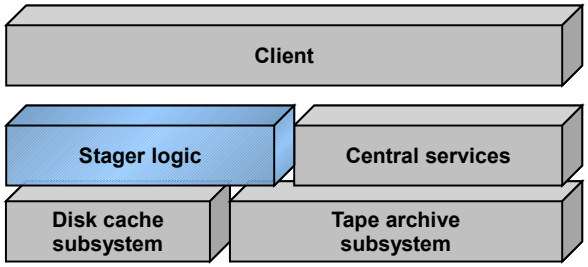
- Design principles
  - Decisions taken:
    - at DB level (stored procedures), or
    - by external plugins (scheduler, expert system)
  - Typical decisions
    - Preparation of ***migration*** or ***recall*** streams
    - Weighting of file systems used for migration/recall
    - Garbage collection decisions
  - Actions performed by dedicated daemons

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Stager Architecture

CERN IT Department

- Database centric architecture
  - daemons are stateless
  - Well defined database interfaces separated from the rest of the code
  - only **Oracle** is fully supported



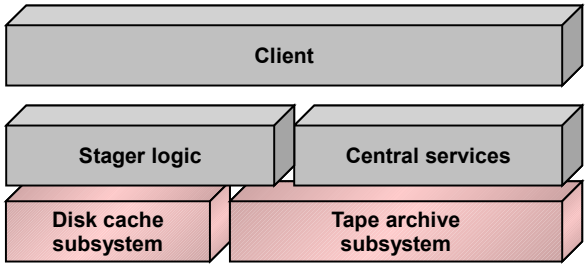
- Stateless components
  - can be restarted/parallelized easily
    ⇒ no single point of failure
  - Stager split in many independent services
    - distinction between queries, user requests and admin requests
    - fully scalable
- Minimal footprint of inactive requests
  - Requests are not instantiated in terms of processes until they run
    - Stored in DB and/or scheduler while waiting for resources
    - Number of *migrator / recaller* instances ~ nb drives (no process instances while waiting for drive)

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

DM Disk Cache and Tape Archive

CERN IT Department

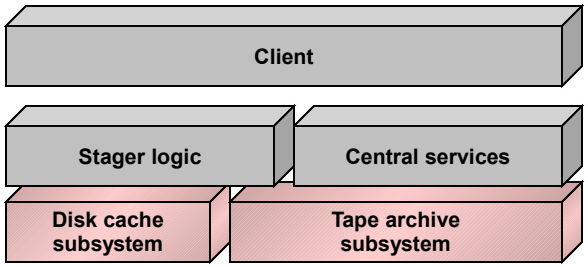
- Scheduled disk access
  - All user requests are scheduled
  - Advanced scheduling features for ‘free’ (e.g. fair-share)
  - Only **LSF** is supported
    - **Maui** used to be but development froze last year
- Dynamic *migration / recall* streams to / from tape
  - Multiple concurrent requests for same volume will be processed together
  - New requests arriving after the stream has started are automatically added to the stream



CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Disk Cache and Tape Archive

- Pluggable policies
  - For recall, migration, I/O scheduling, GC
  - Defined per disk pool but centrally written
  - Allows support for
    - volatile storage (GC, no migration)
    - durable storage (no GC, no migration)
    - permanent storage (GC, migration)
- “Pluggable” protocols
  - RFIO and ROOT internal, XROOT, GridFTP both internal and external
  - “Easy” addition of new protocols
    - but no clean interface



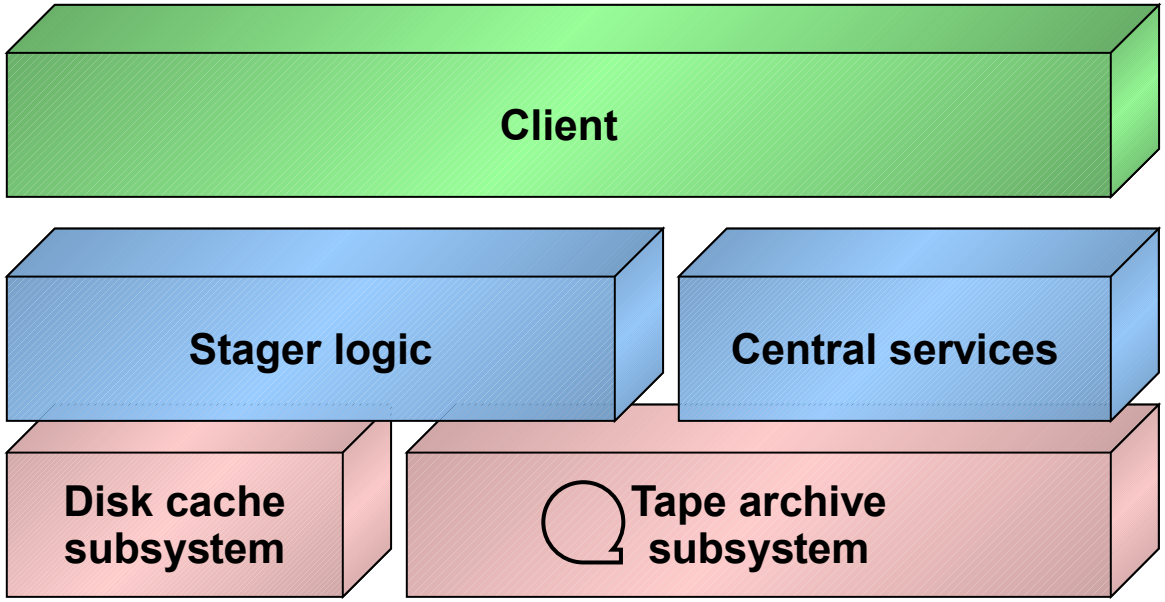
CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Security and safety aspects

- Authorization
  - per file ACLs at the namespace level
  - restricted access to diskservers (rootd, rfio, xrootd)
- Authentication
  - strong authentication under work
    - first version available in 2.1.6 to be released now
- Resiliency against hardware failures
  - any node can die with no major impact
    - relying on the DB for data, all deamons can be replicated
- Disaster recovery
  - regular backups of the DBs

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

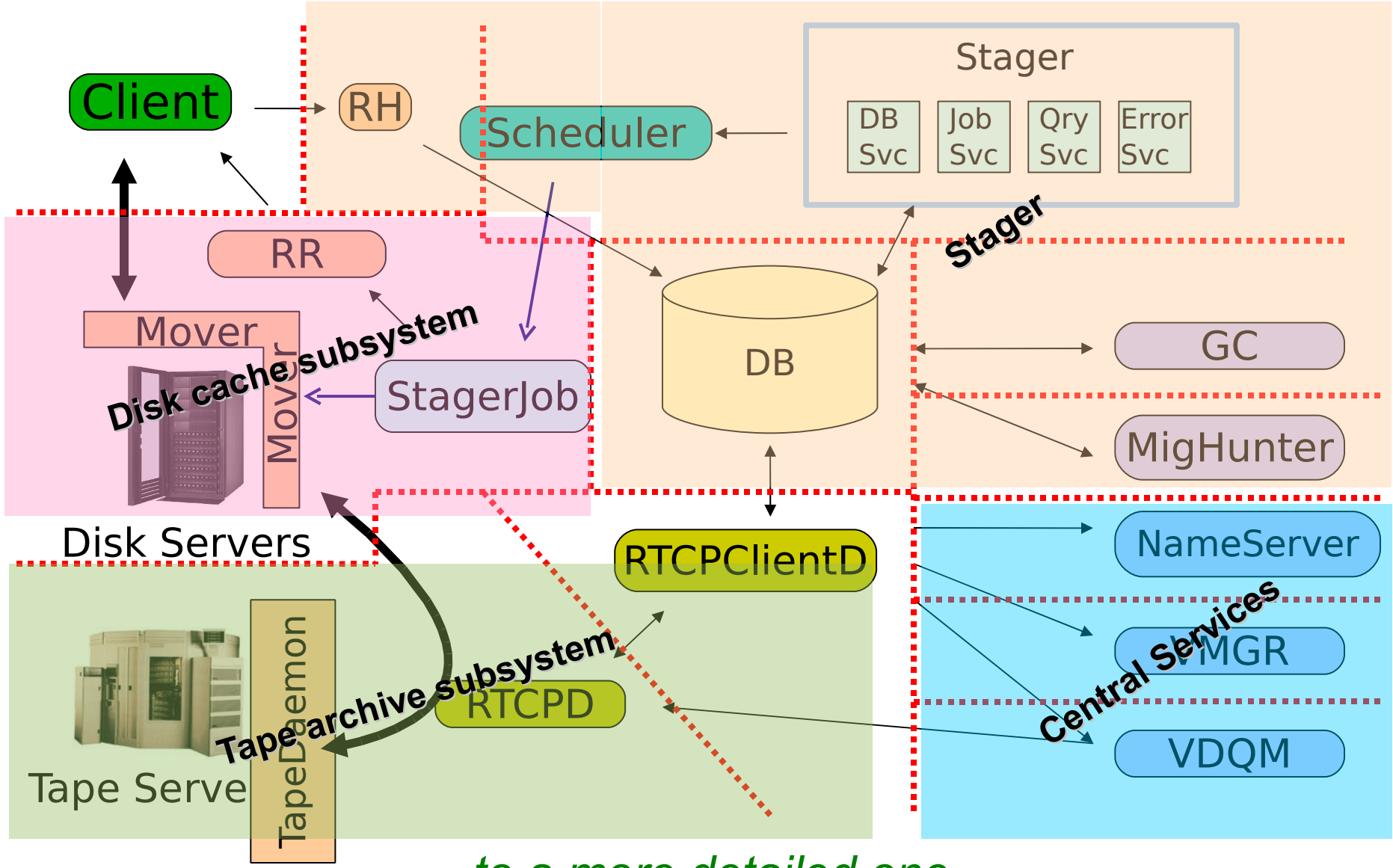
Castor 2 Architecture



From the "simple" view …

Castor 2 Architecture

CERN IT Department



… to a more detailed one

Old and new components

- Castor 2 has its roots in mid 1990s
  - Several components have been left almost unchanged
    - E.g. the Central Services
  - New components have been written from scratch
  - Happy mix of old and new code
    - Different styles due to different developers and
      (mostly) different languages
    - Most C++ code is interfaced in C

…not a surprise taking into account time extension of the project and number of involved people!

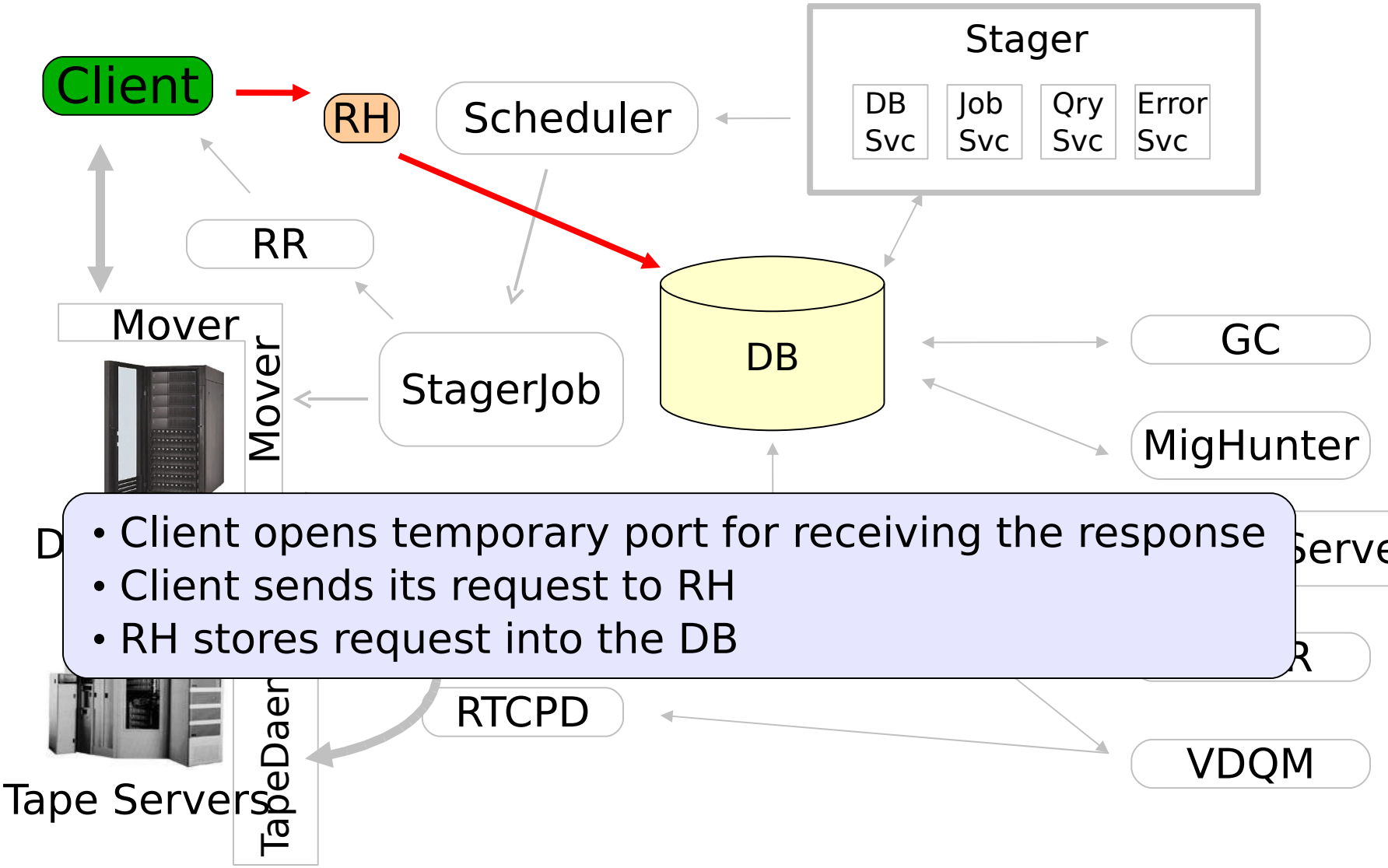
Lifecycle of a GET + recall

- Client connects to the RH
- RH stores the request into the db
- Stager polls the db and checks for file availability
- If the file is not available, the recall process is activated
- Once the file is available, stager asks the scheduler to schedule the access to the file
- Client gets a callback and can initiate the transfer
- The commandline is stager_get

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

stager_get (1)

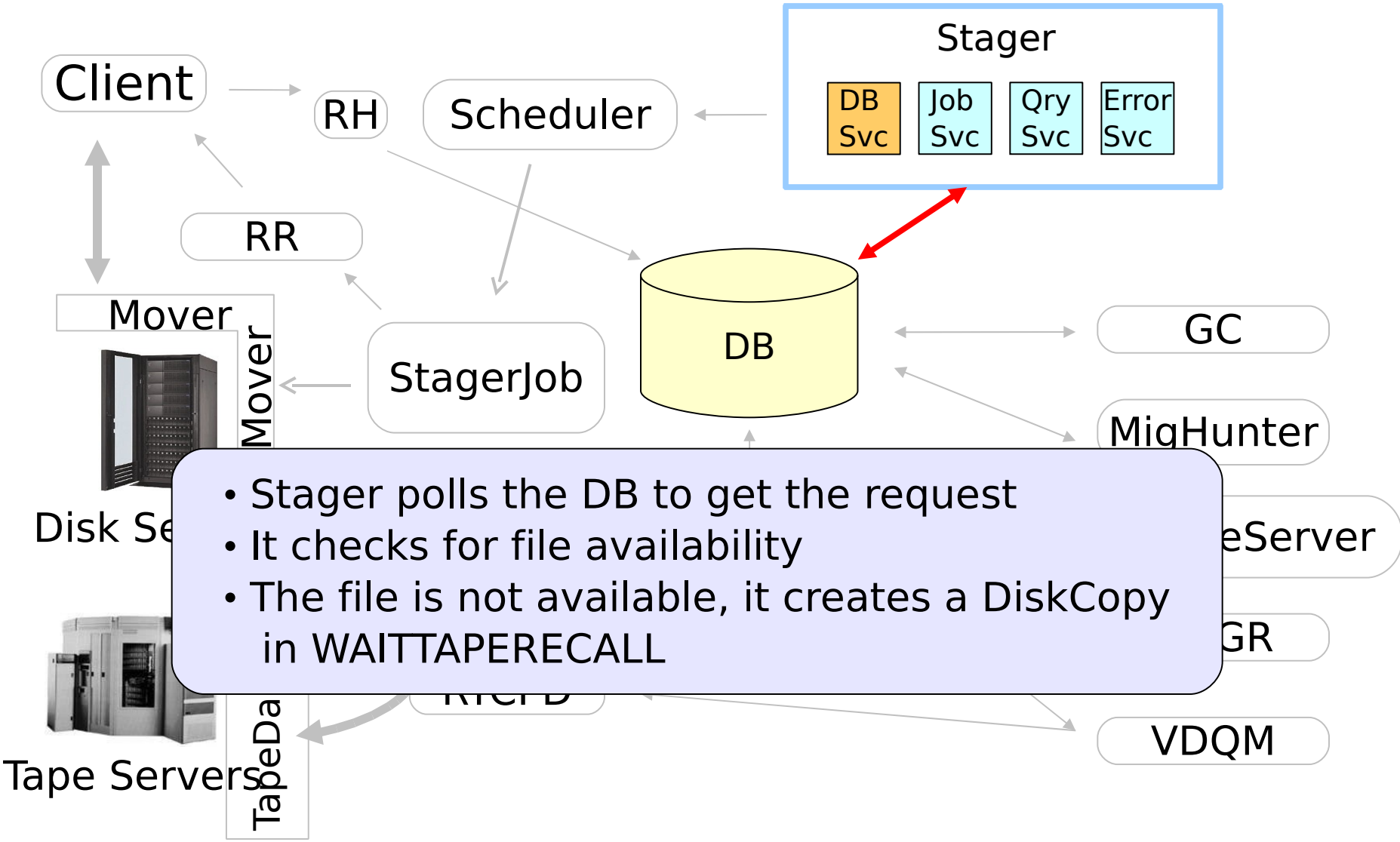
CERN IT Department



- Client opens temporary port for receiving the response
- Client sends its request to RH
- RH stores request into the DB

stager_get (2)

CERN IT Department

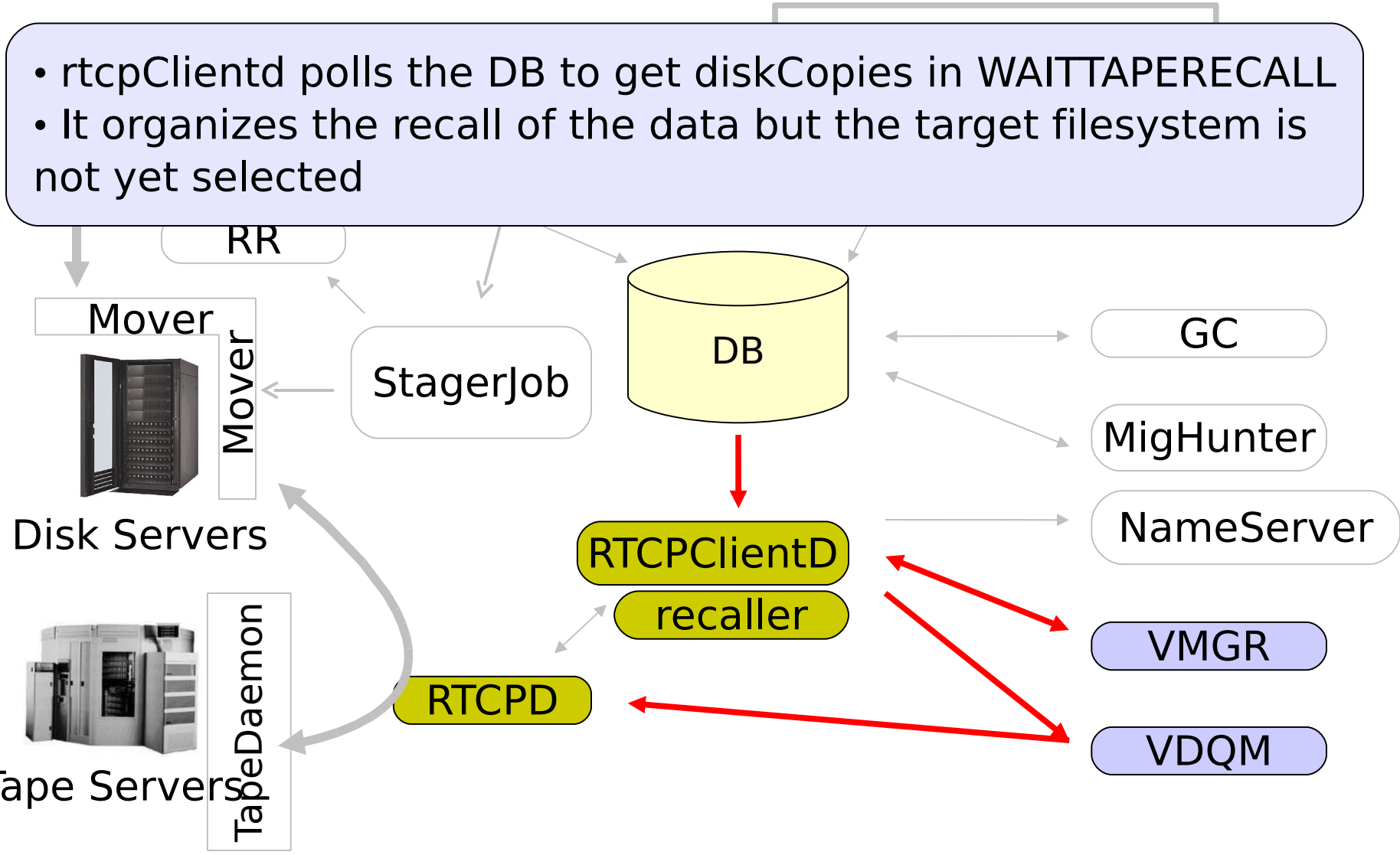


- Stager polls the DB to get the request
- It checks for file availability
- The file is not available, it creates a DiskCopy in WAITTAPERECALL

stager_get (3)

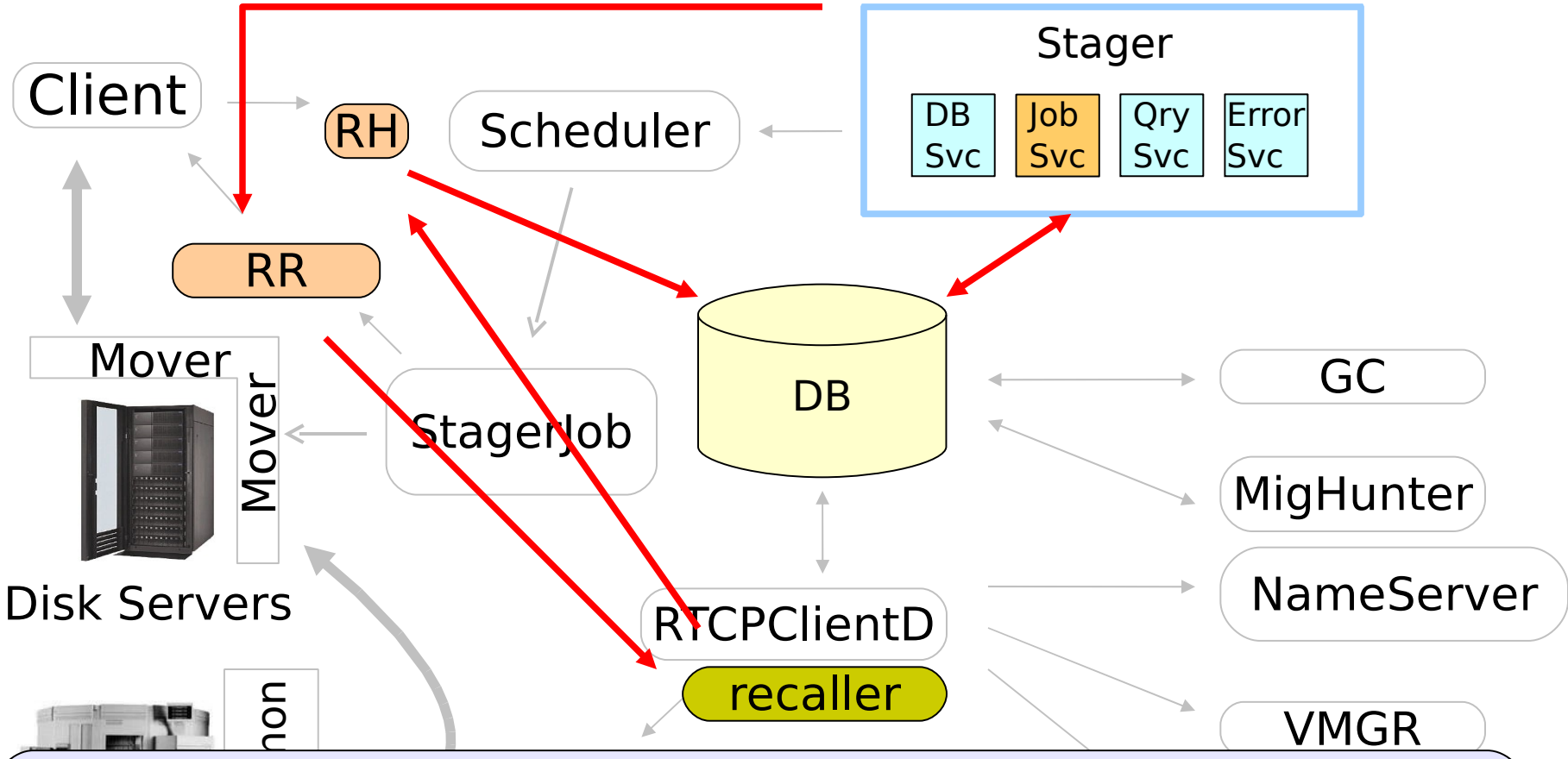
CERN IT Department

- rtcpClientd polls the DB to get diskCopies in WAITTAPERECALL
- It organizes the recall of the data but the target filesystem is not yet selected



stager_get (4)

CERN IT Department

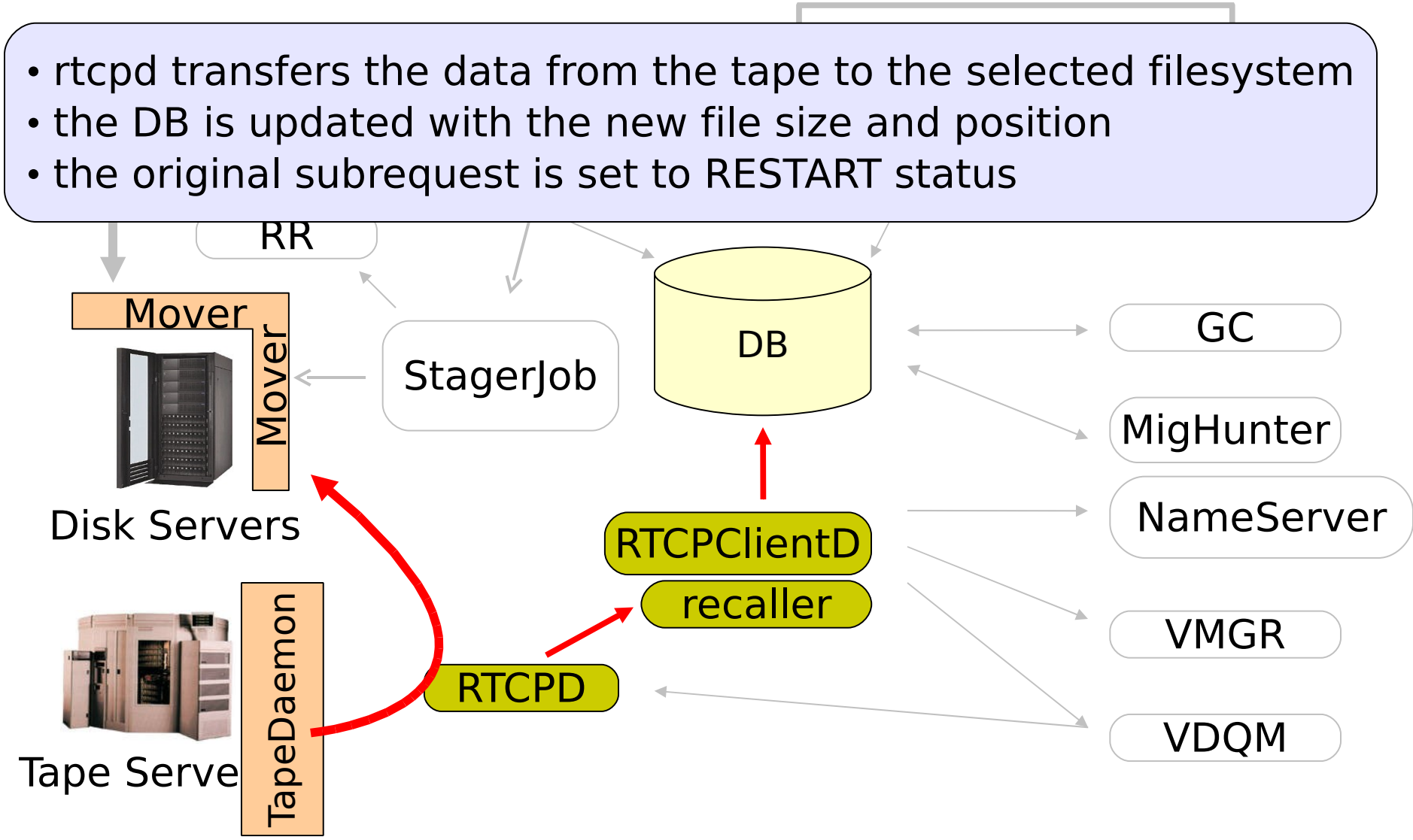


- recaller sends a request to the stager in order to know where to put the file
- the request goes through the usual way: Request Handler, DB, stager (job service), Request Replier

stager_get (5)

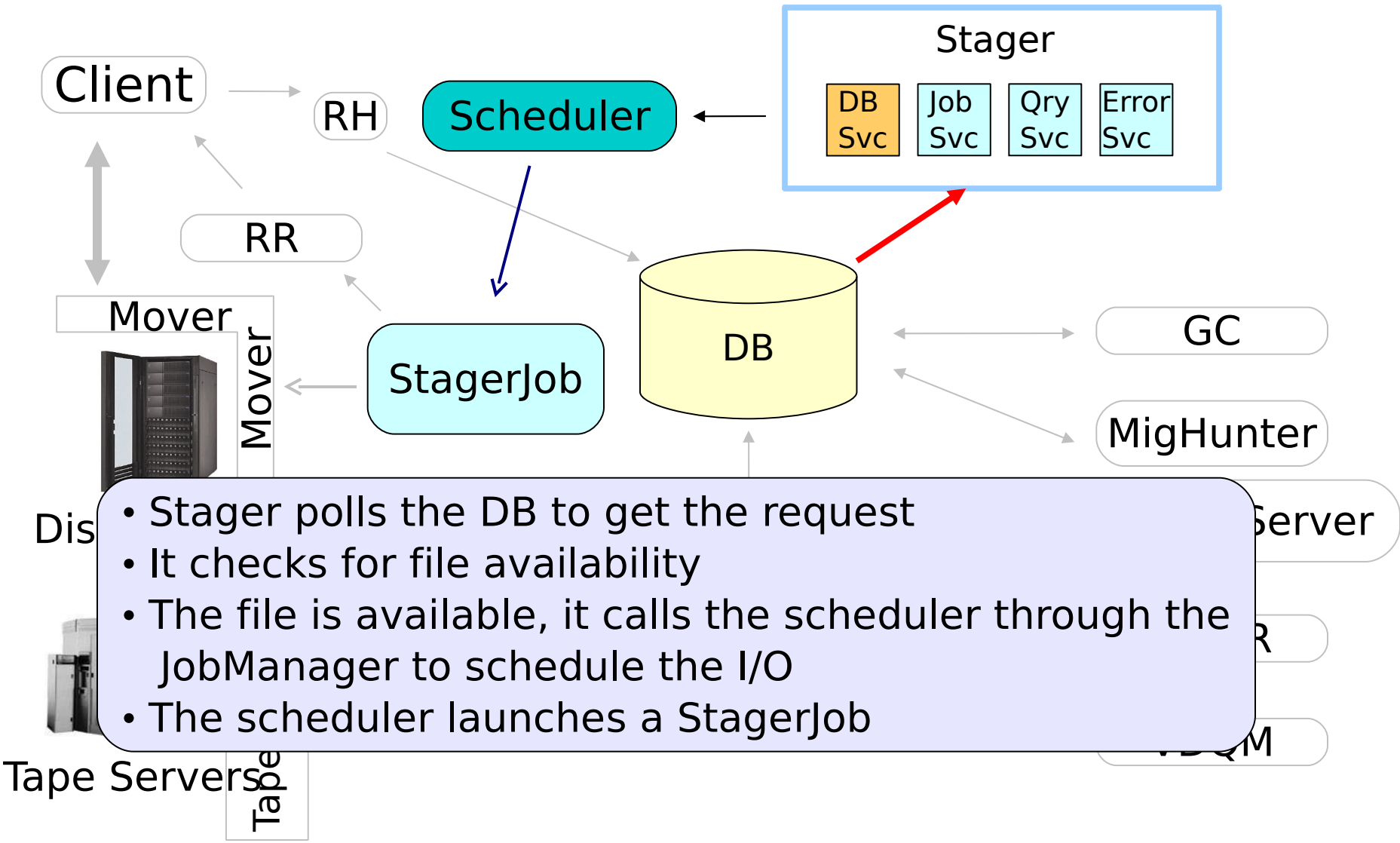
CERN IT Department

- 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



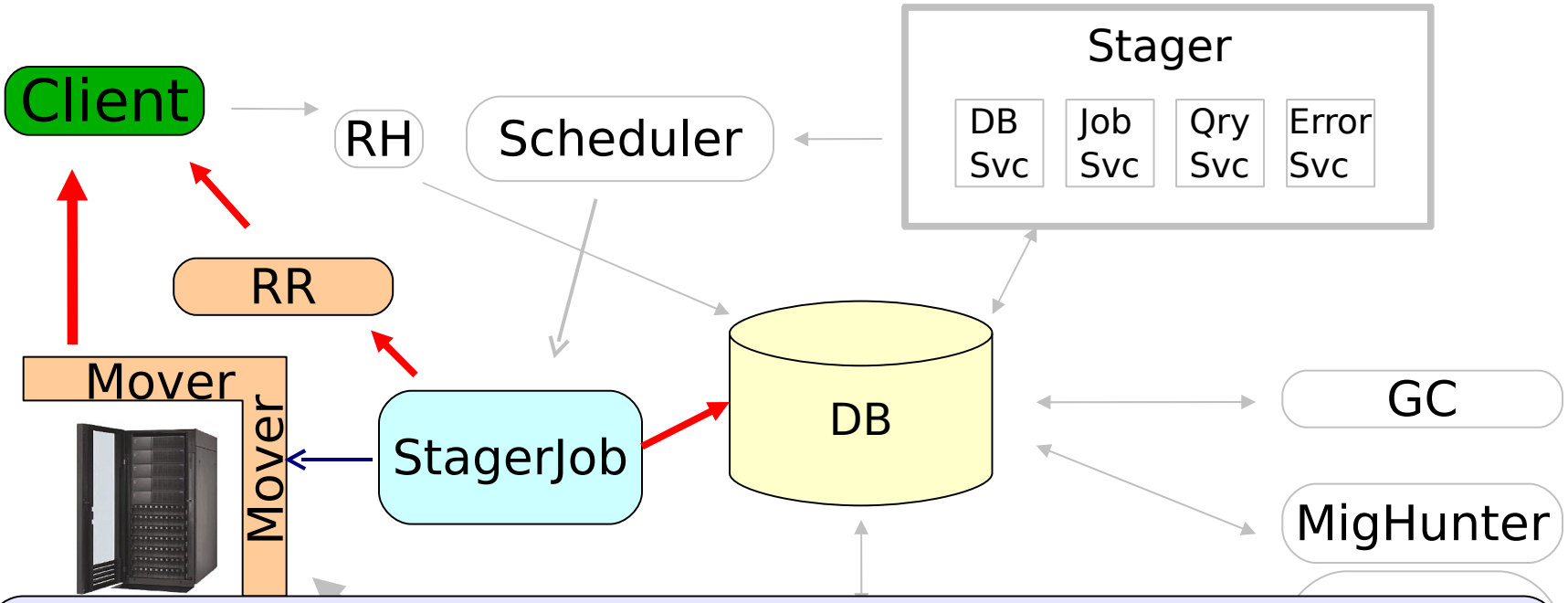
stager_get (6)

CERN IT Department



stager_get (7)

CERN IT Department



- 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

Lifecycle of a PUT + migration

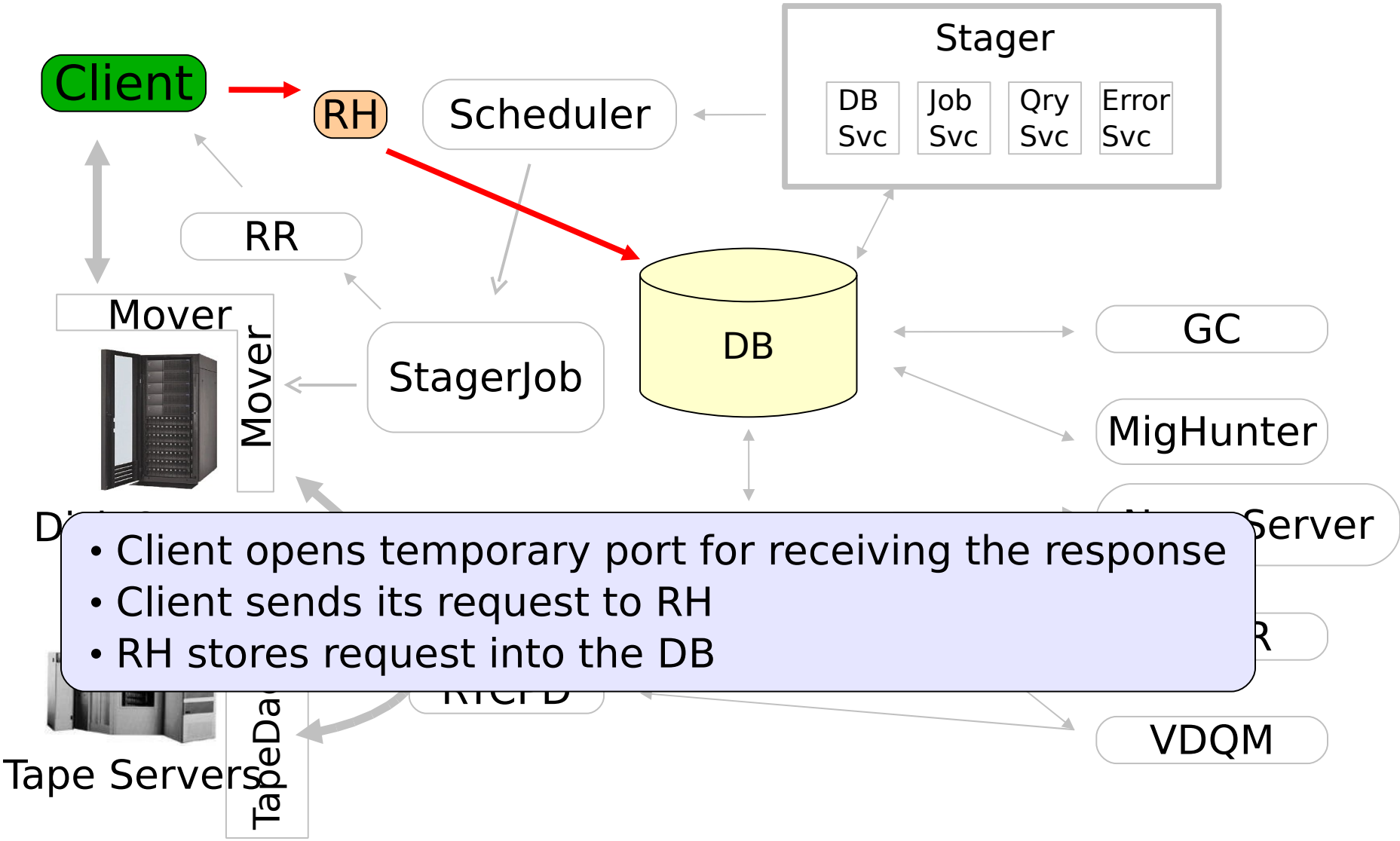
- Client connects to the RH
- RH stores the request into the db
- Stager polls the db and looks for a candidate filesystem for the transfer
- Client gets a callback and can initiate the transfer
- After the transfer is completed, migration to tape is performed

- The commandline is stager_put

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

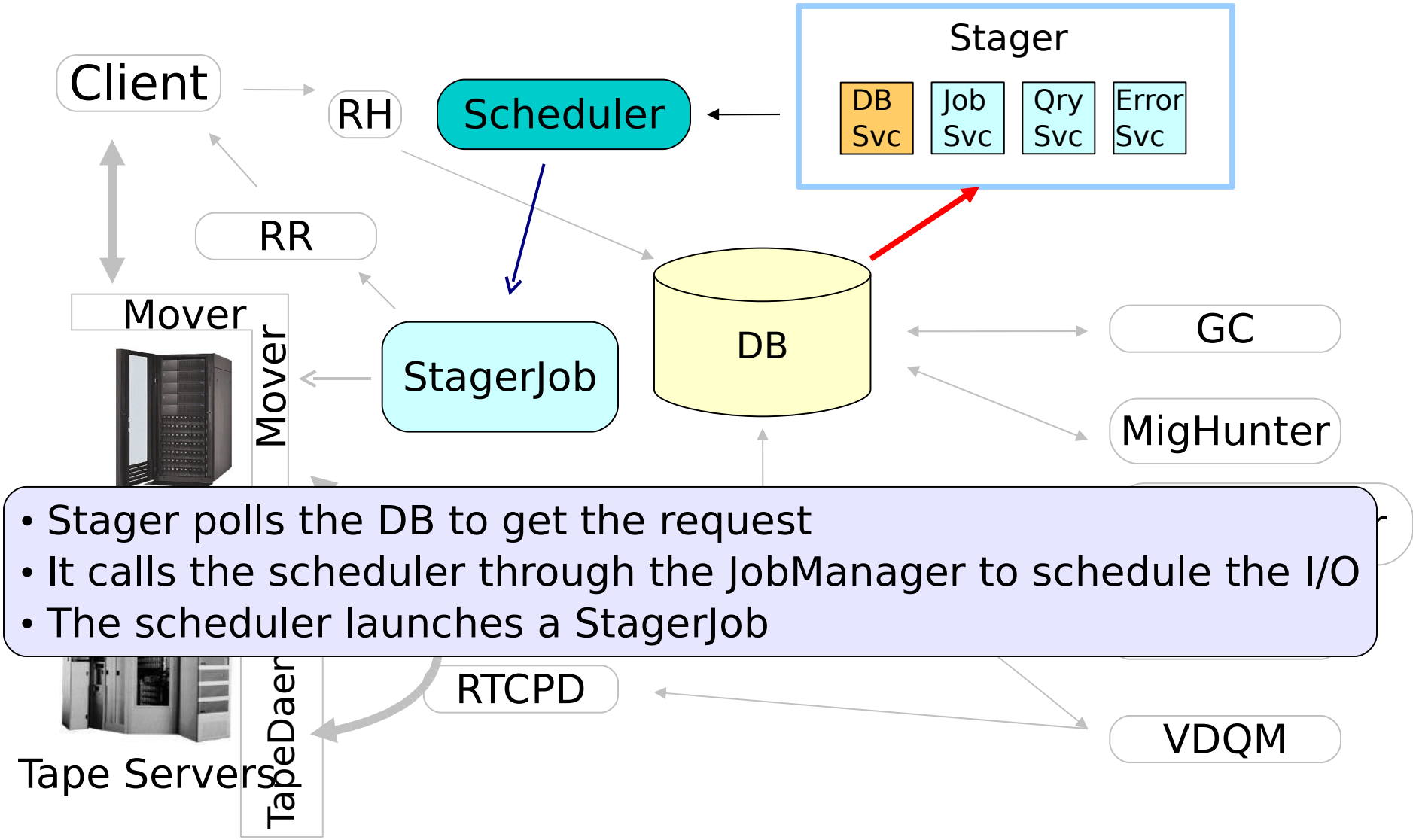
stager_put (1)

CERN IT Department



stager_put (2)

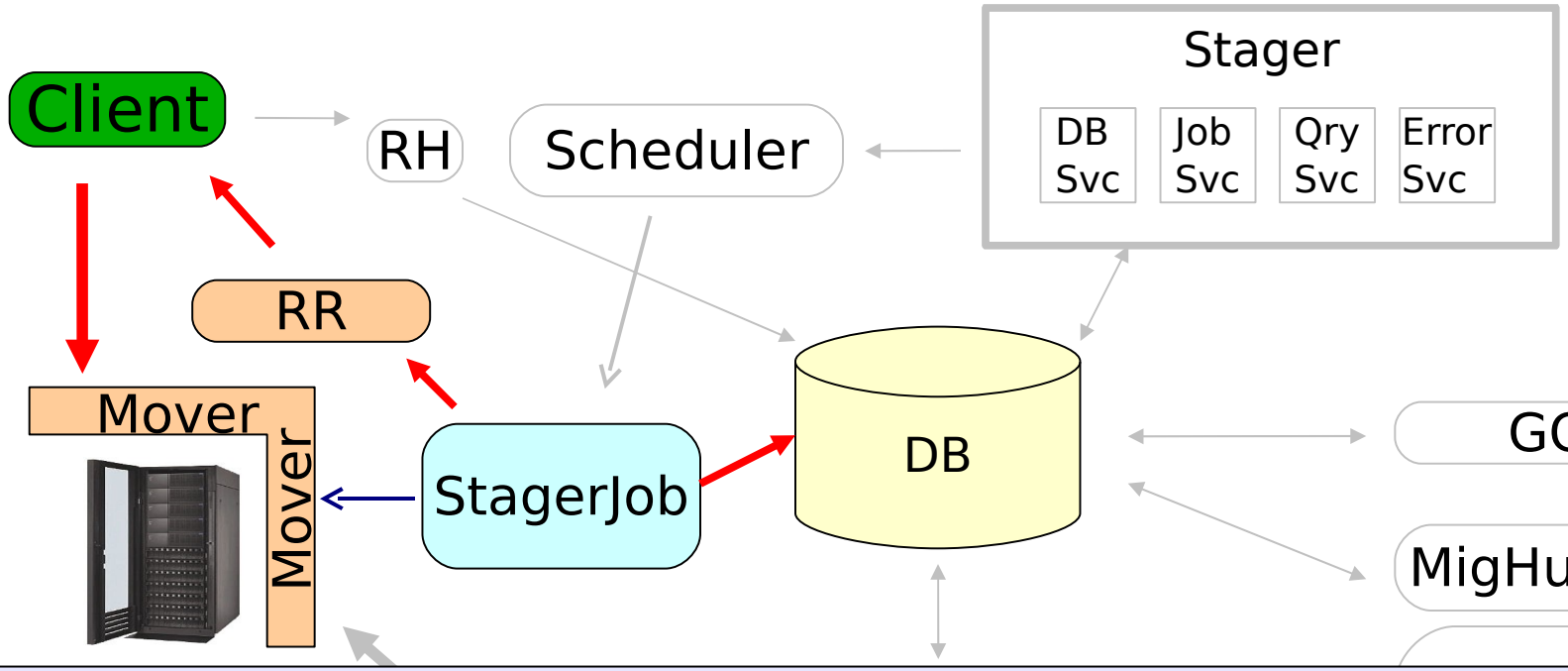
CERN IT Department



- Stager polls the DB to get the request
- It calls the scheduler through the JobManager to schedule the I/O
- The scheduler launches a StagerJob

stager_put (3)

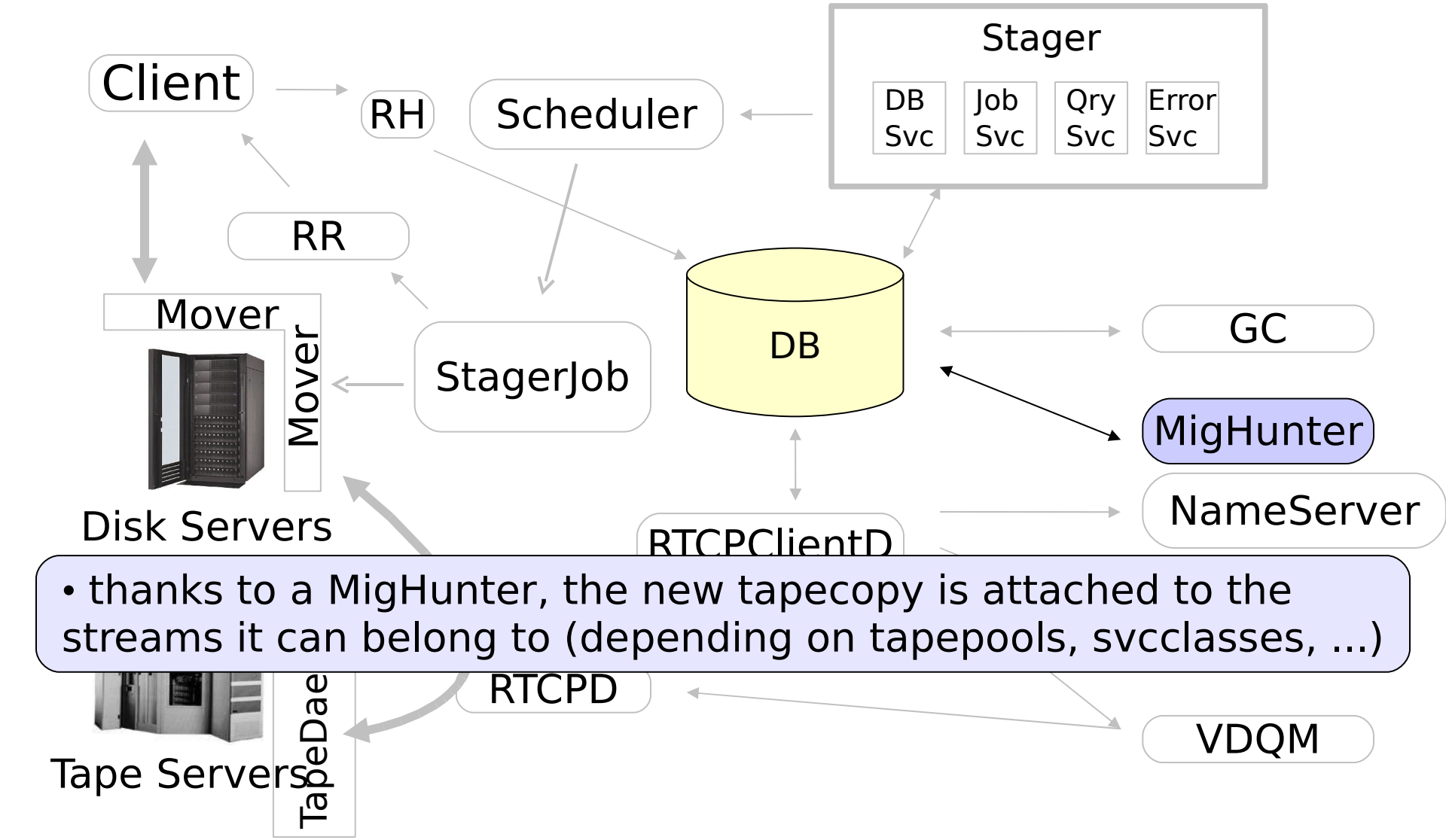
CERN IT Department



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

stager_put (4)

CERN IT Department

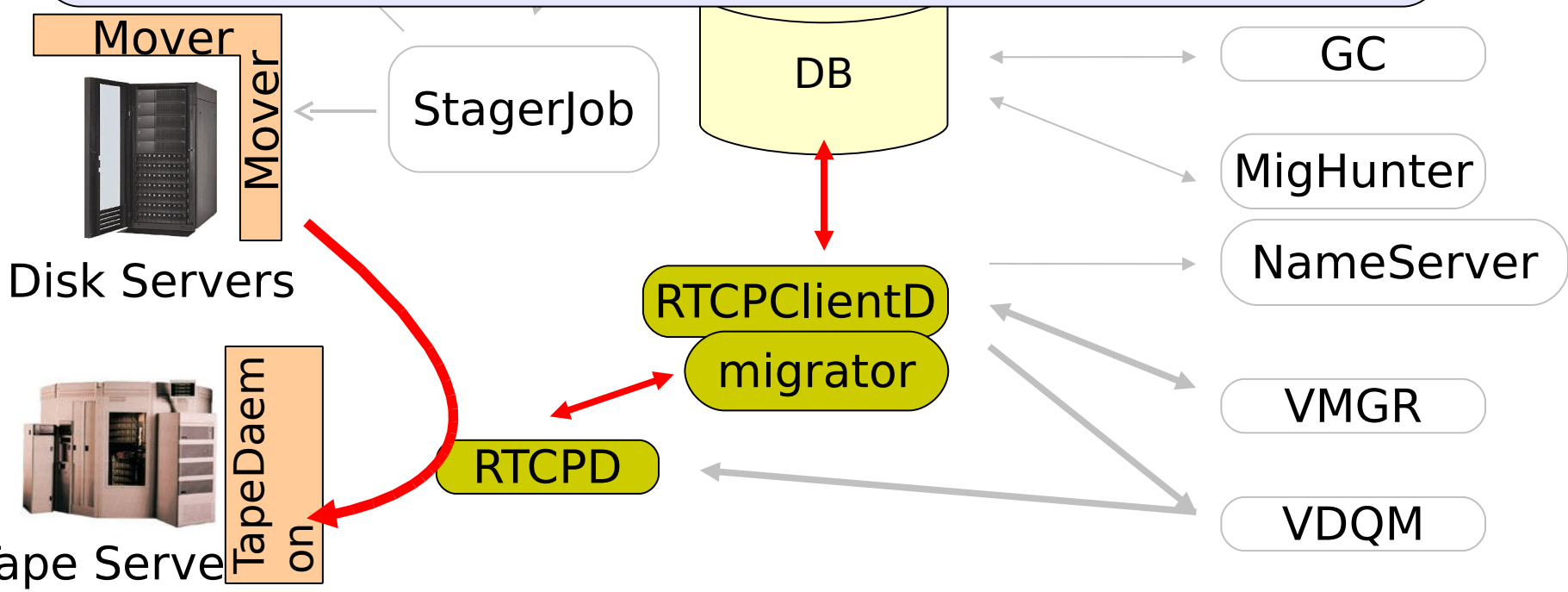


• thanks to a MigHunter, the new tapecopy is attached to the streams it can belong to (depending on tapepools, svcclasses, ...)

stager_put (5)

CERN IT Department

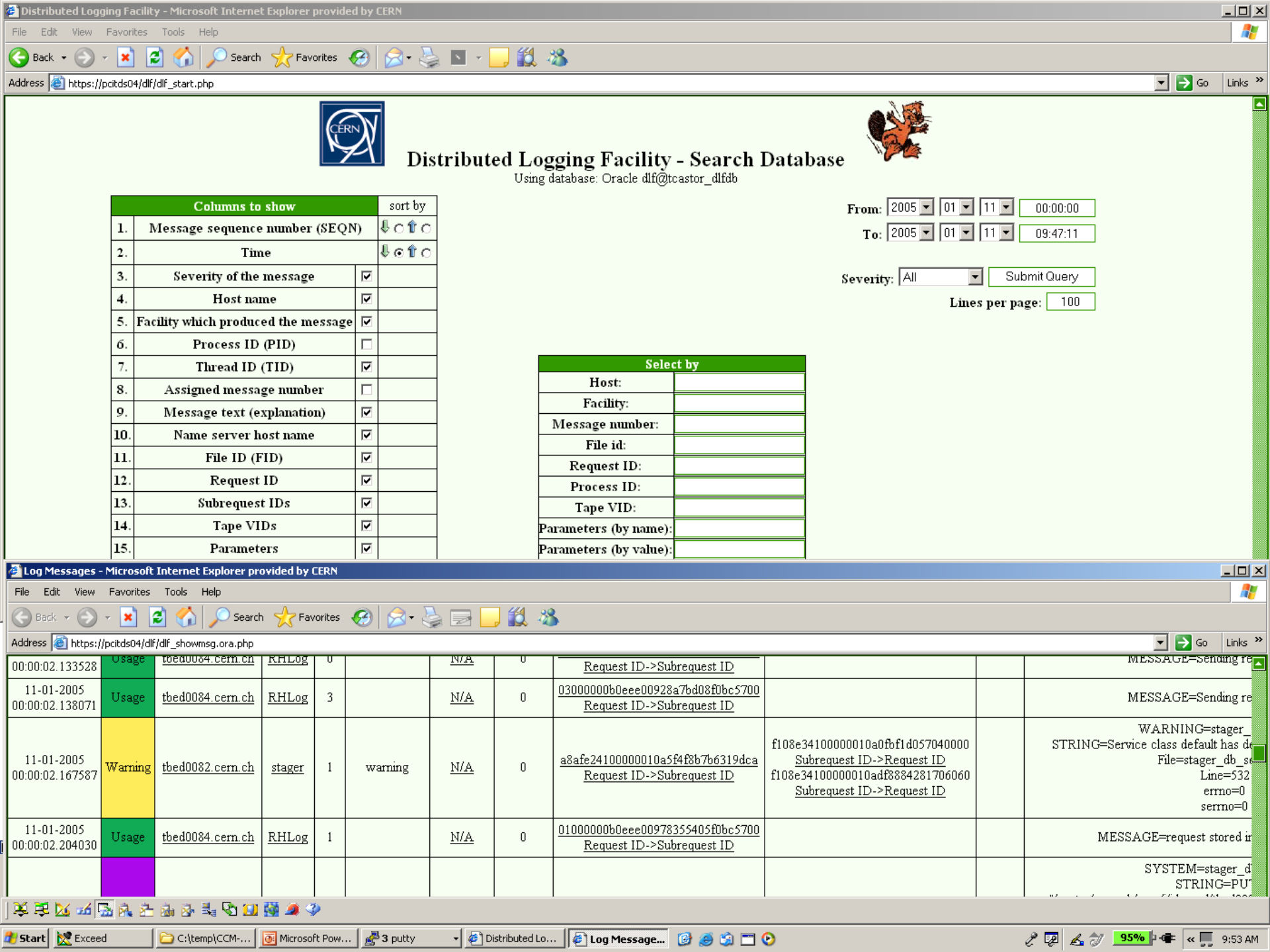
- rtcpclientd will launch a migrator
- this one asks the DB for the next migration candidate
- the DB takes the best candidate in the stream (based on filesystems availability)
- the file is written to tape and the DB updated



Extra components

- Monitoring daemons
  - rmNodeDaemon runs on all diskservers
  - rmMasterDaemon collects the information
- JobManager
  - a wrapper around the LSF API
    - LSF API is non thread safe
- LSF Plugin
  - Select best candidate resource (file system) among the set proposed by LSF
- Distributed logging facility (DLF)
  - a central DB allowing easy log browsing

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it


Distributed Logging Facility - Microsoft Internet Explorer provided by CERN
File Edit View Favorites Tools Help
Back Search Favorites
Address https://pcitds04/dlf/dlf_start.php
Go Links
Distributed Logging Facility - Search Database
Using database: Oracle dlf@tcastor_dlfdb
Columns to show
sort by
1. Message sequence number (SEQN)
2. Time
3. Severity of the message
4. Host name
5. Facility which produced the message
6. Process ID (PID)
7. Thread ID (TID)
8. Assigned message number
9. Message text (explanation)
10. Name server host name
11. File ID (FID)
12. Request ID
13. Subrequest IDs
14. Tape VIDs
15. Parameters
From: 2005 01 11 00:00:00
To: 2005 01 11 09:47:11
Severity: All
Submit Query
Lines per page: 100
Select by
Host:
Facility:
Message number:
File id:
Request ID:
Process ID:
Tape VID:
Parameters (by name):
Parameters (by value):
Log Messages - Microsoft Internet Explorer provided by CERN
File Edit View Favorites Tools Help
Back Search Favorites
Address https://pcitds04/dlf/dlf_showmsg.ora.php
Go Links
00:00:02.133528 Usage tbed0084.cern.ch RHLog 0 N/A 0 Request ID->Subrequest ID MESSAGE=Sending re
11-01-2005 00:00:02.138071 Usage tbed0084.cern.ch RHLog 3 N/A 0 03000000b0eee00928a7bd08f0bc5700 Request ID->Subrequest ID MESSAGE=Sending re
11-01-2005 00:00:02.167587 Warning tbed0082.cern.ch stager 1 warning N/A 0 a8afe24100000010a5f4f8b7b6319dca Request ID->Subrequest ID f108e34100000010a0fbf1d057040000 Subrequest ID->Request ID f108e34100000010adf8884281706060 Subrequest ID->Request ID WARNING=stager_ STRING=Service class default has d File=stager_db_s Line=532 errno=0 serrno=0
11-01-2005 00:00:02.204030 Usage tbed0084.cern.ch RHLog 1 N/A 0 01000000b0eee00978355405f0bc5700 Request ID->Subrequest ID MESSAGE=request stored i
SYSTEM=stager_d STRING=PU
Start Exceed C:\temp\CCM-... Microsoft Pow... 3 putty Distributed Lo... Log Message... 95% 9:53 AM

Extra components (2)

- Garbage collection
  - gcDaemon running on every diskServer
  - central SQL code taking garbage collection decisions based on SQL polices
- Python policy service
  - allows easy execution of python policies
  - still quite efficient thanks to on the fly precompilation
  - used for recall, stream, migration and scheduling policies

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Outline

- CASTOR2 context
  - Scope and constraints
  - History
    - SHIFT, CASTOR 1
  - Requirements
- CASTOR 2 architecture overview
  - blocks and components
  - practical example : lifecycle of get/put requests
  - extra components
- Technology choices
  - about languages
  - UML and the code generation
  - DB centric

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

About languages

- Object Orientation
  - Java like inheritance
    - Simple, multiple only for interfaces
  - Heavy use of (pure) abstract interfaces
    - Each service/component has one
    - Some services even have several implementations
      - IGCSvc → RemoteGCSvc and OraGCSvc
- Implementation language is C++
  - Still some subparts in C
  - Happy mix, special thanks to code generation
- Castor 1 is pure C, no OO

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

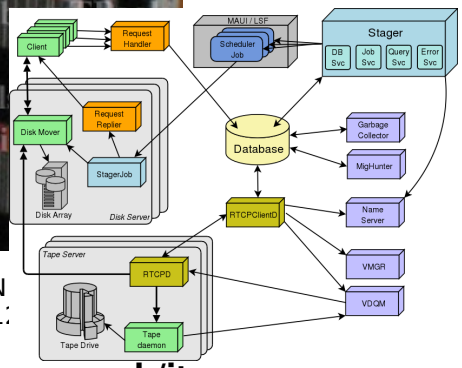
Still around languages

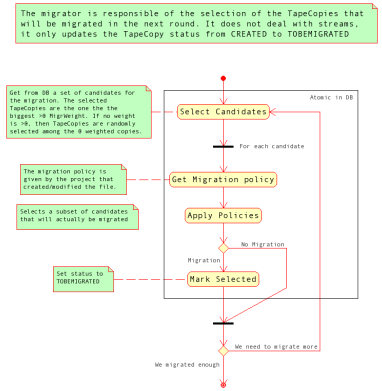
- Admin scripts are a mix of perl and python
  - New ones in python, future is python
- No imposed dev environment
  - Wide range of tools among developpers
    - vi(m), (x)emacs
    - jedit, kdeveloper
    - nedit
  - Even different linux distributions
- Documentation
  - Recent documents are tex
  - Older are more word

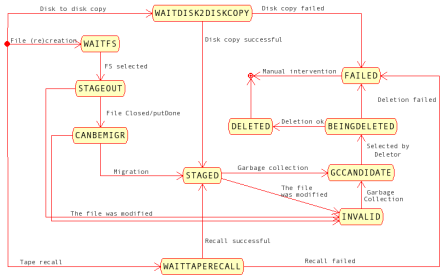
CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

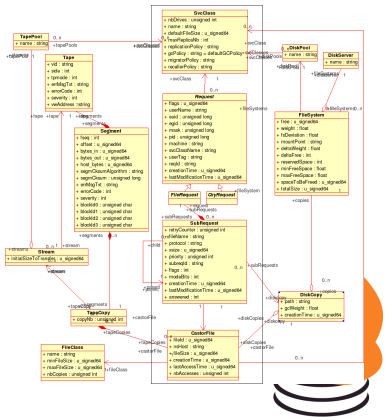
Architecture design

- Based on UML
  - Component views for the rough architecture
  - Activity diagrams per component
  - State diagrams per object
  - Sequence diagrams per use case
  - Class diagrams for implementation details, per compenent
- Using umbrello
  - Essentially because of its code generation



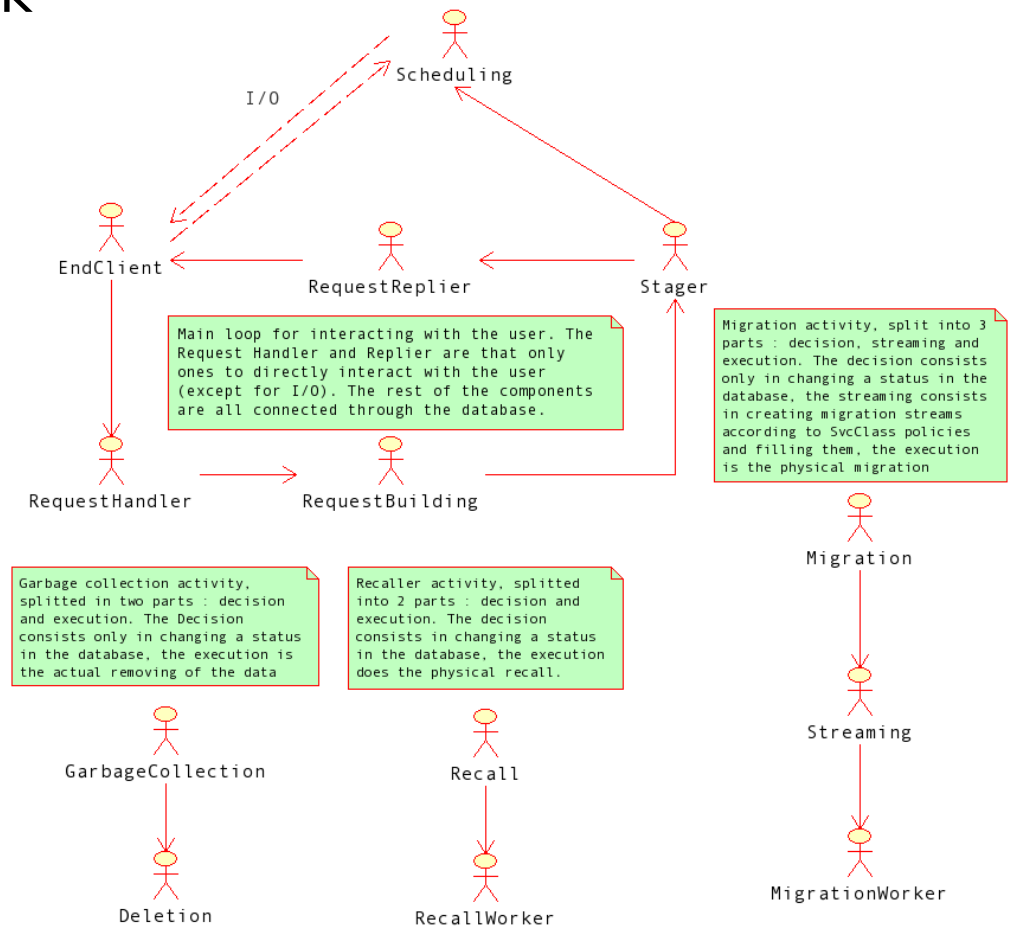






Component views

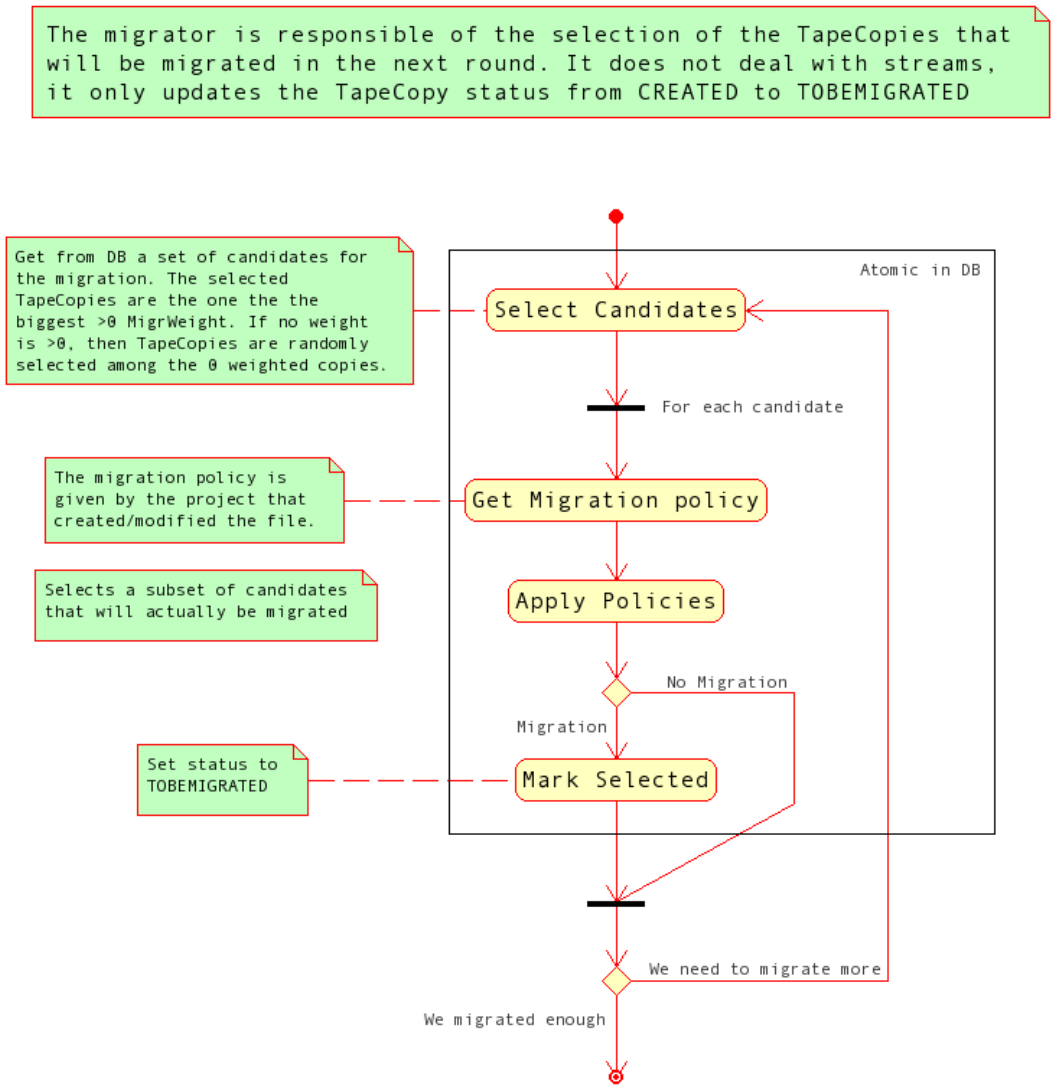
- Used for designing the overall architecture and the relations between components
- The building block is a component
- The relations mean that the 2 components communicate at some stage



CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Activity diagrams

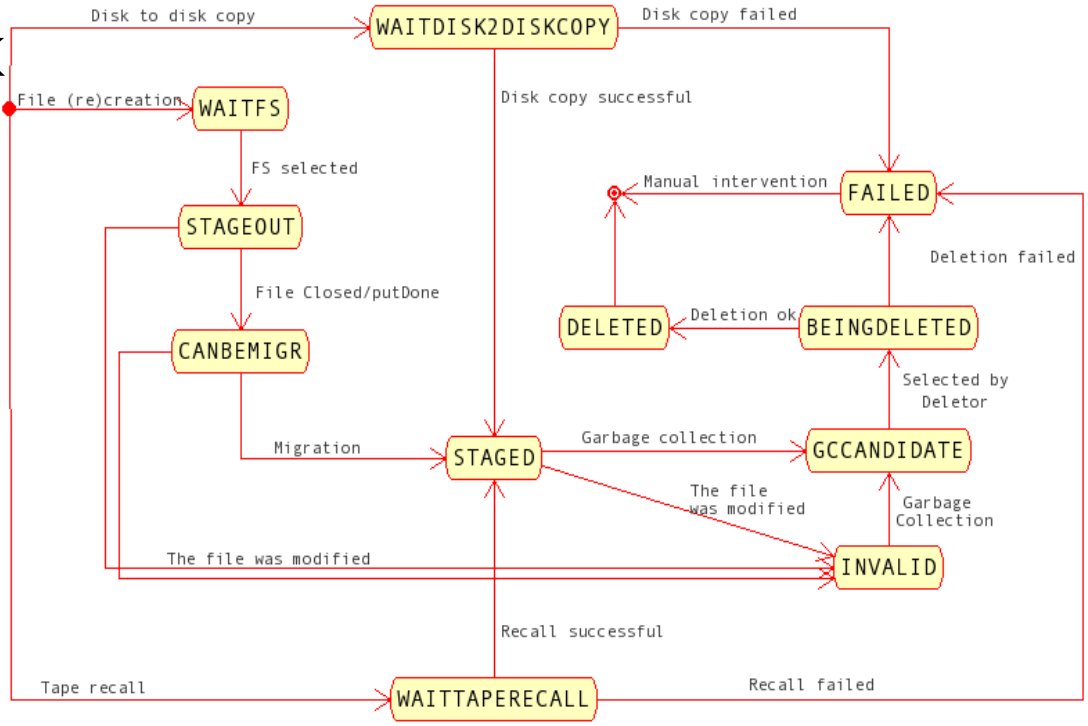
- Used for describing the work flow of a component
- The building block is a simple (atomic) operation
- Arrows indicate the flow of time



CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

State Diagrams

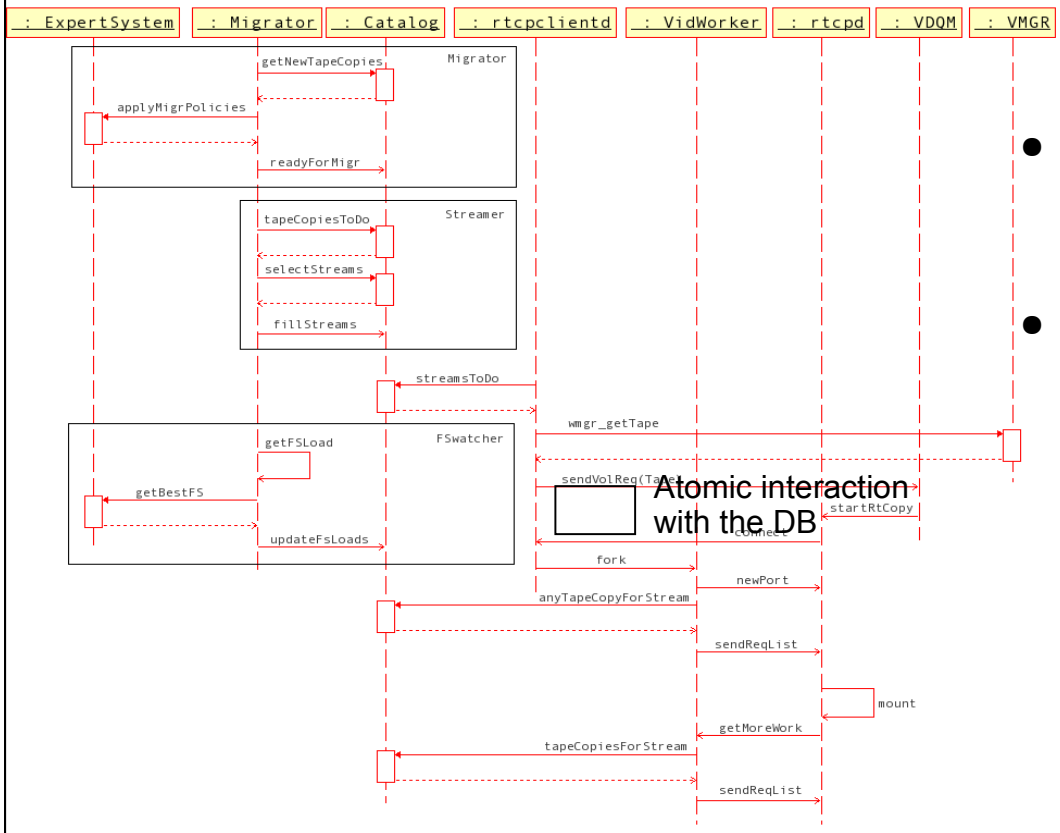
- Used for describing the possible states and state transitions of objects
- Allows to easily find out implications of adding a new state or state transition
- Building block is a state
- Arrows indicate state transitions



CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Sequence diagrams

- Used for analyzing interactions between components and their timing
- Useful to avoid deadlocks and inefficiencies due to bad granularity

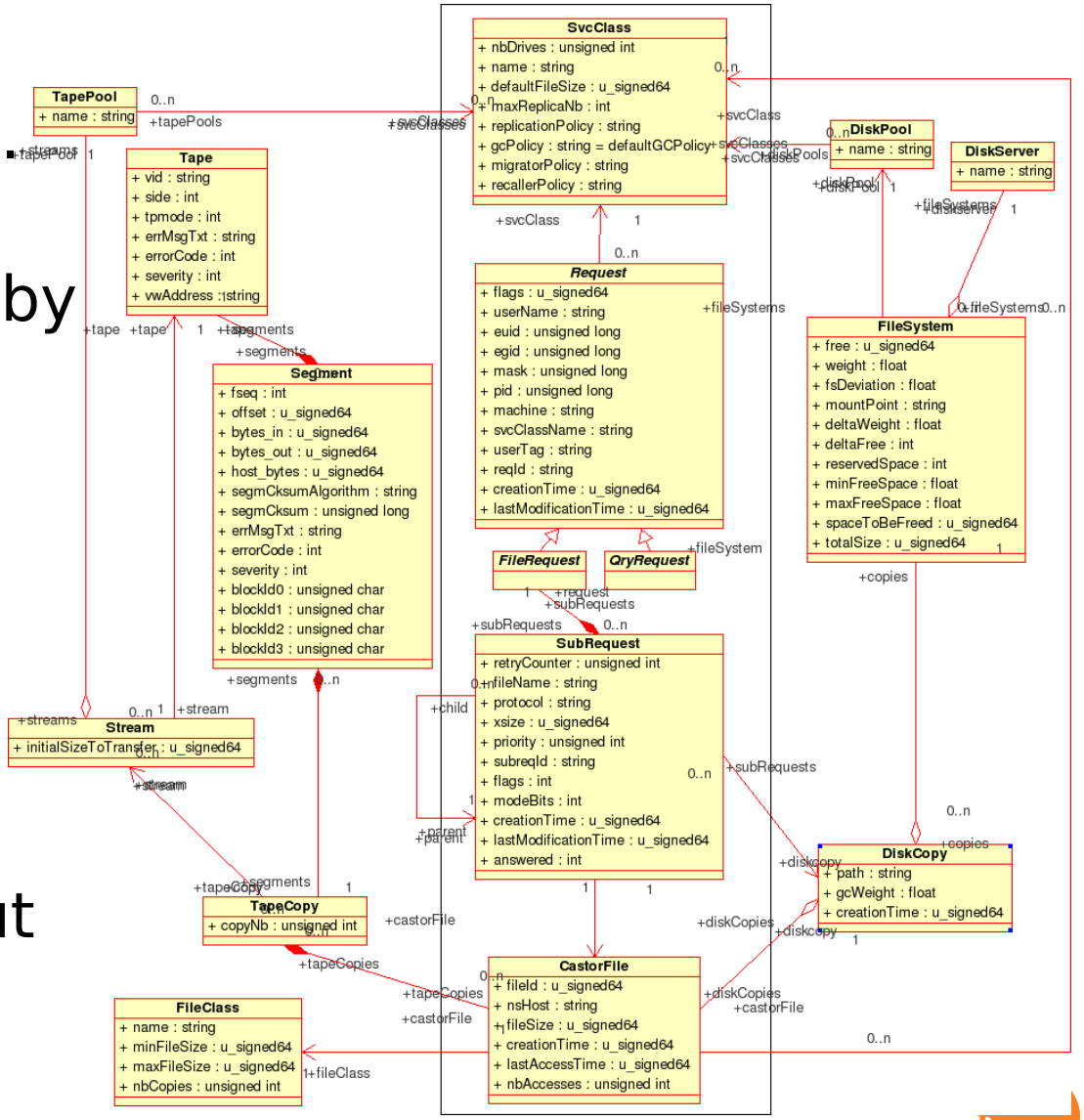


- Blocks are actions
- Arrows mean communication between components

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Class diagrams

- Used of fine grained design.
- Blocks are the items handled by the code (mapped to classes)
- Relations are links from one object to another
- This is the input of the code generation



CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Usage of UML diagrams

- For the design
  - At the project level
  - At the code level
  - For the database schema
    - class diagram maps to the DB schema
- For code generation
  - Of header files and object implementation (data objects)
  - Of DB scripts : schema creation/deletion
  - Of a DB access layer in the code
    - allows to store/retrieve/update/(un)link objects
  - Of I/O libraries dealing with objects

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Goal of code generation

- Saves typing for class declaration and implementation
- Automates some facilities (object printing, introspection)
- Automates streaming and DB access
- Generates C interface to C++ code
- Allows easy maintenance

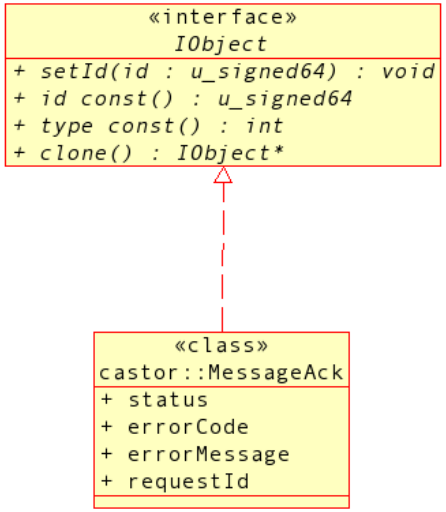
```
castor::IObject* obj = sock->readObject()
…
castor::BaseAddress ad;
ad.setCnvSvcName("DbCnvSvc");
…
svcs()->createRep(&ad, obj);
```

Core of the Request Handler code

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Class implementation

C++ code




```
class MessageAck : public virtual IObject {
…
private:
    /// Status of the request
    bool m_status;
    /// Code of the error if status shows one
    int m_errorCode;
```

.hpp

```
void castor::MessageAck::print(…) const {
  …
  stream << indent << "status : "
         << m_status << std::endl;
  stream << indent << "errorCode : "
         << m_errorCode << std::endl;
```

.cpp

```
int C_MessageAck_create(struct C_MessageAck_t**);
int C_MessageAck_print(struct C_MessageAck_t*);
```

.h

And C interface

```
int C_MessageAck_create(castor::MessageAck** obj) {
  *obj = new castor::MessageAck();
  return 0;
}
int C_MessageAck_print(castor::MessageAck* instance) {
  instance->print();
  return 0;
}
```

CInt.cpp

Converters

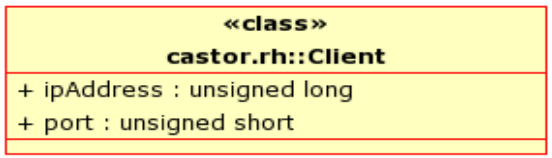
CERN IT Department

Stream*Cnv.h/cpp

Streaming code

```
virtual void createRep(IAddress*, IObject*);
```

```
void … StreamMessageAckCnv::createRep(…) {
  …
  ad->stream() << obj->type();
  ad->stream() << obj->ipAddress();
  ad->stream() << obj->port();
  ad->stream() << obj->id();
```



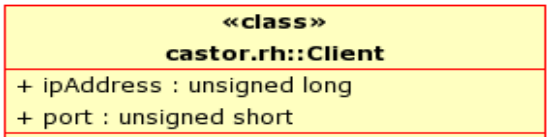
Ora*Cnv.h/cpp

And DB code

```
virtual void createRep(IAddress*, IObject*);
```

```
void … StreamMessageAckCnv::createRep(…) {
  …
  const std::string insertStmStr =
"INSERT INTO Client (ipAddress, port, id)…";
  …
 insertStmt = createStatement(insertStmStr);
  …
 insertStm->executeUpdate();
```

DB scripts



castor_oracle_create.sql

```
/* SQL statements for type Client */
CREATE TABLE Client (ipAddress NUMBER, port NUMBER, id
INTEGER PRIMARY KEY) INITRANS 50 PCTFREE 50;
```

castor_oracle_drop.sql

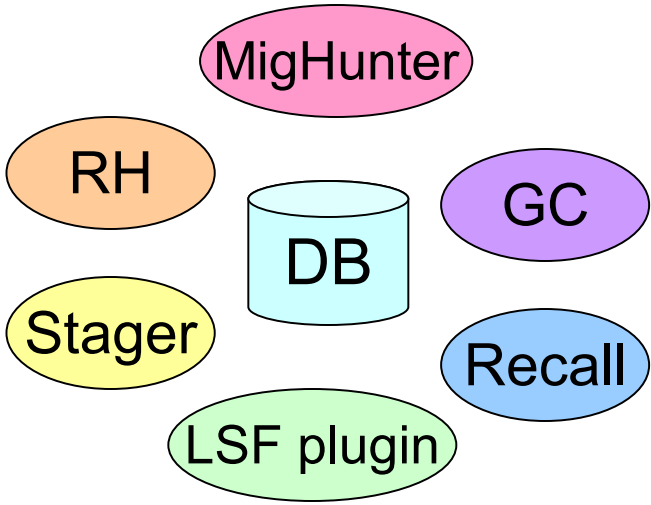
```
/* SQL statements for type Client */
DROP TABLE Client;
```

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

DM Database Centric soft

CERN IT Department

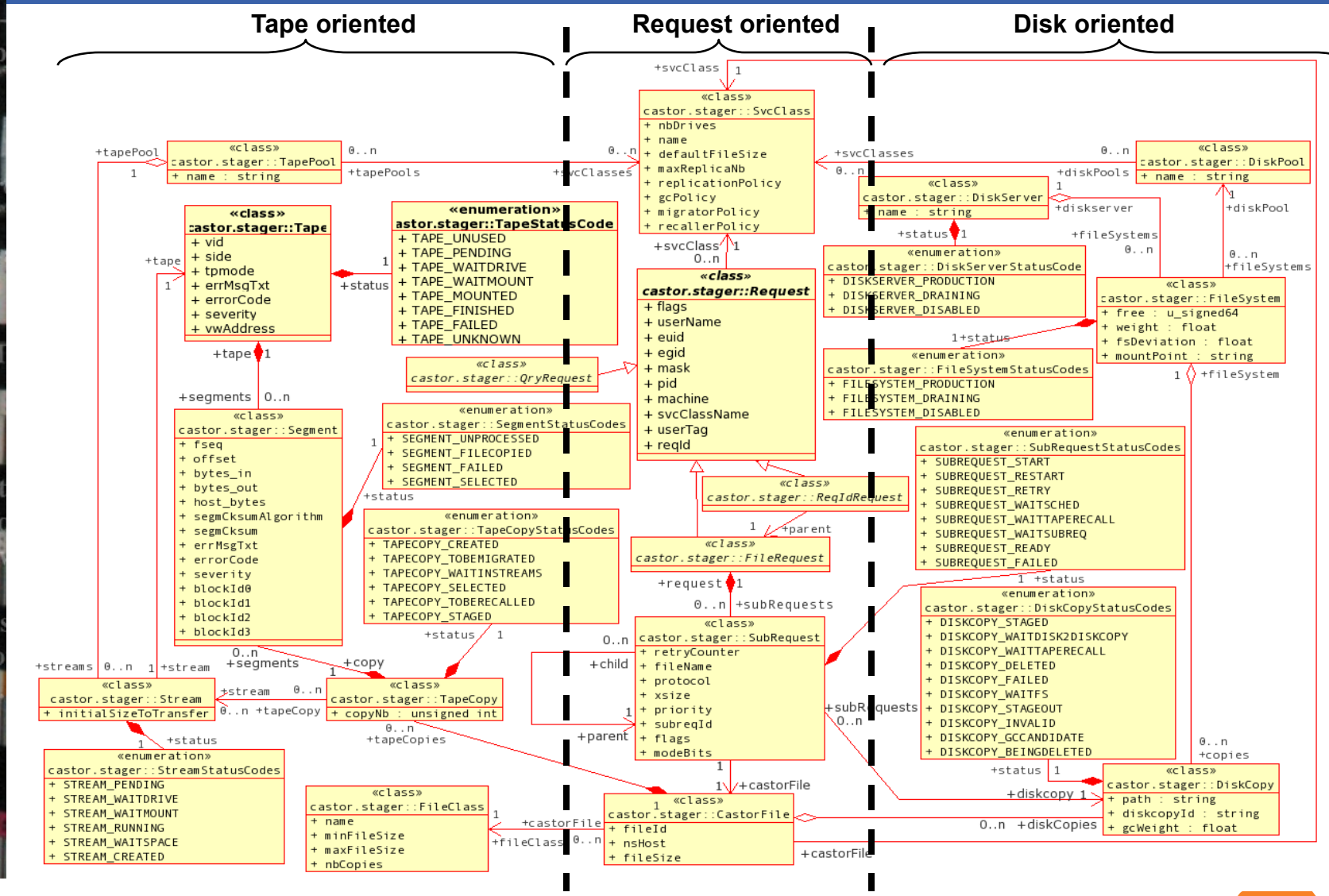
- Reliability
  - No single point of failure
    - replication of components
  - Locking handled by the DB
  - Backups handled by the DB
- Scalability
  - All component can be replicated
  - No catalog in memory
  - Limited by DB scalability
    - No risk from the space point of view
    - CPU is the limit. A lot of tuning done.
    - No fear so far, CPU usage in a production instance running stress testing is ~10%



CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Catalogue DB schema

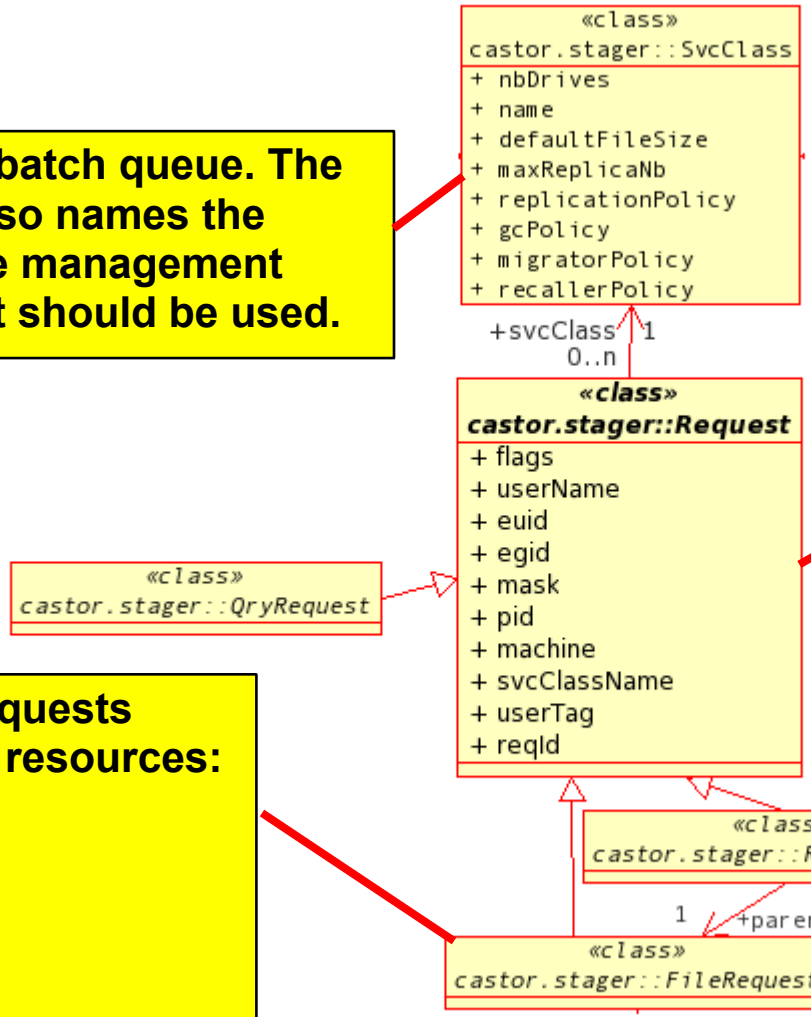
CERN IT Department



CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Request classes

SvcClass ≈ batch queue. The SvcClass also names the CASTOR file management policies that should be used.



Requestor information

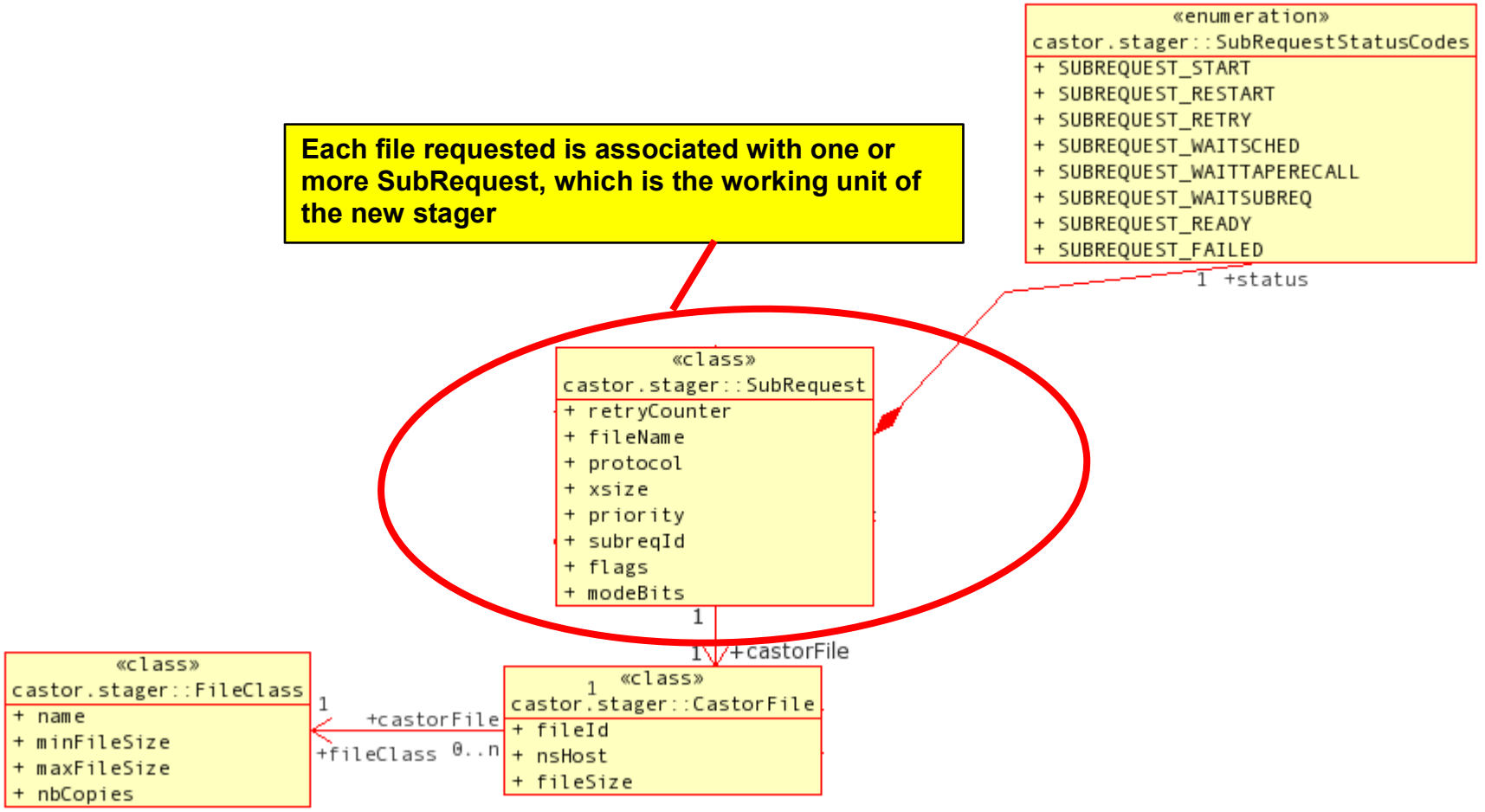
FileRequests are requests requiring access to resources:

- **stageGet**
- **stagePut**
- **stageUpdate**
- **prepareToXXX**

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

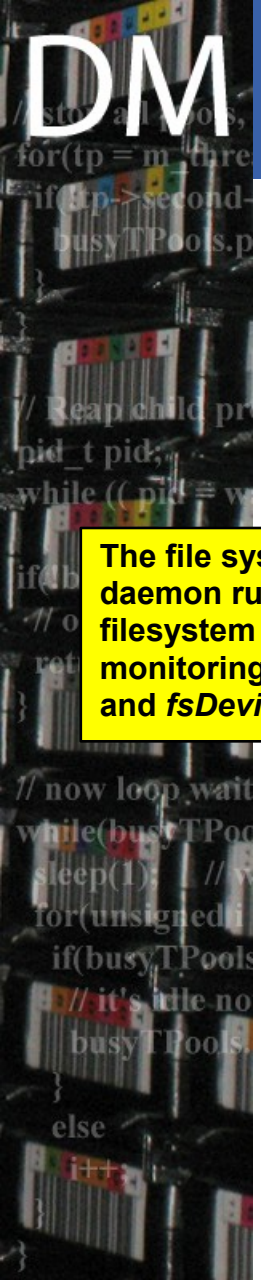
File requests

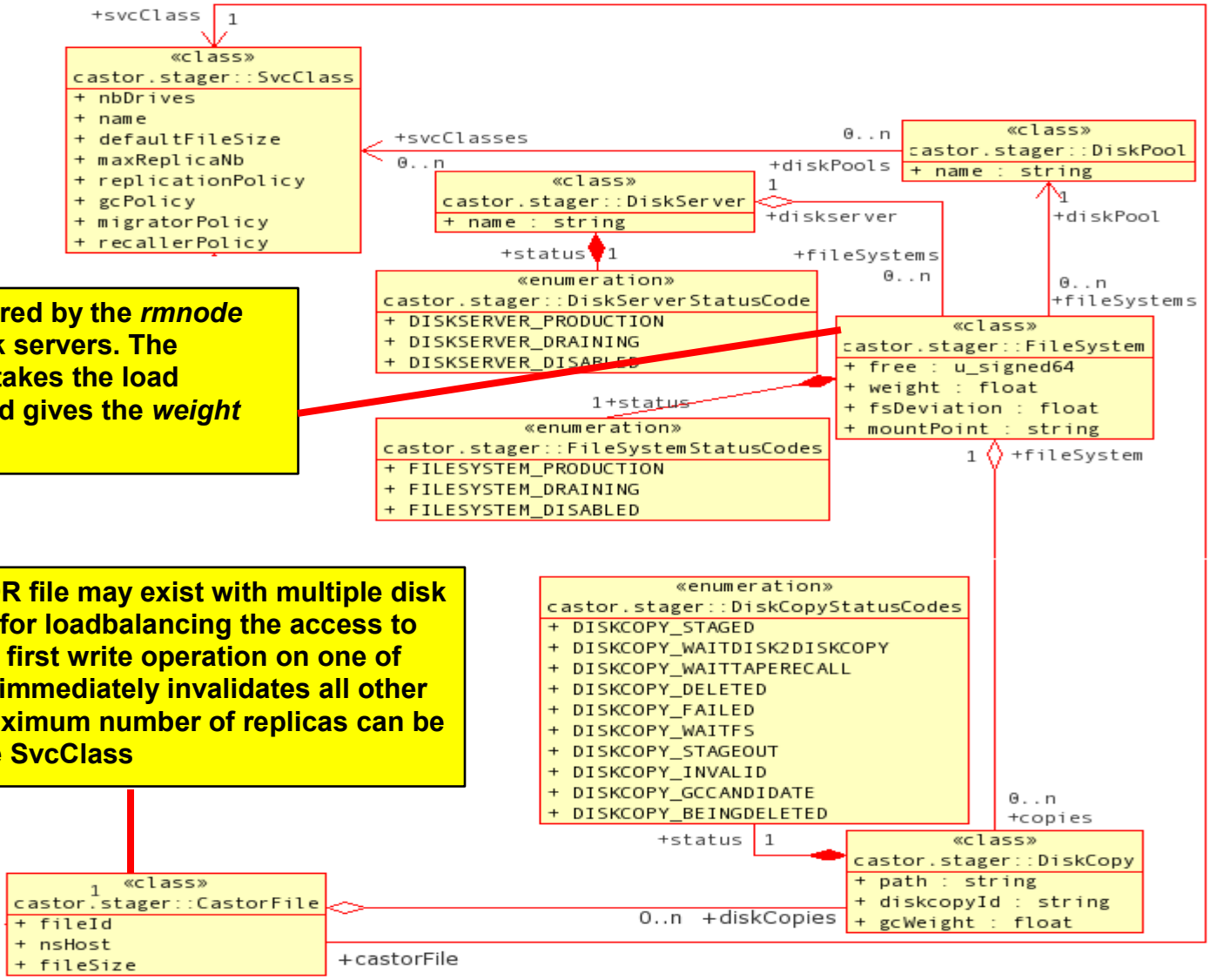
CERN IT Department



Disk residence

CERN IT Department





The file systems are monitored by the *rmnode* daemon running on the disk servers. The filesystem selection policy takes the load monitoring data as input and gives the *weight* and *fsDeviation* as output.

A given CASTOR file may exist with multiple disk copies. Allows for loadbalancing the access to "hot" files. The first write operation on one of the copies will immediately invalidates all other copies. The maximum number of replicas can be specified in the SvcClass

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

That's it for today

CERN IT Department

What we covered :

- history and requirements
- global architecture
- components and request cycle
- main technology choices
  - usage of UML
  - Db centric
  - code generation

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

That's it for today

CERN IT Department

What we did not cover (not exhaustive) :

- CASTOR features
- CASTOR users (Tier 0, Tier 1s)
- the tape layer and its specificities
- the castor 2 core framework
- the supported protocols and their interfaces to CASTOR
- scheduling
- building and testing infrastructures
- anatomy of a CASTOR instance
- hardware and network requirements
- operational aspects
- configuration, monitoring
- performances (data challenges)
- ……

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it