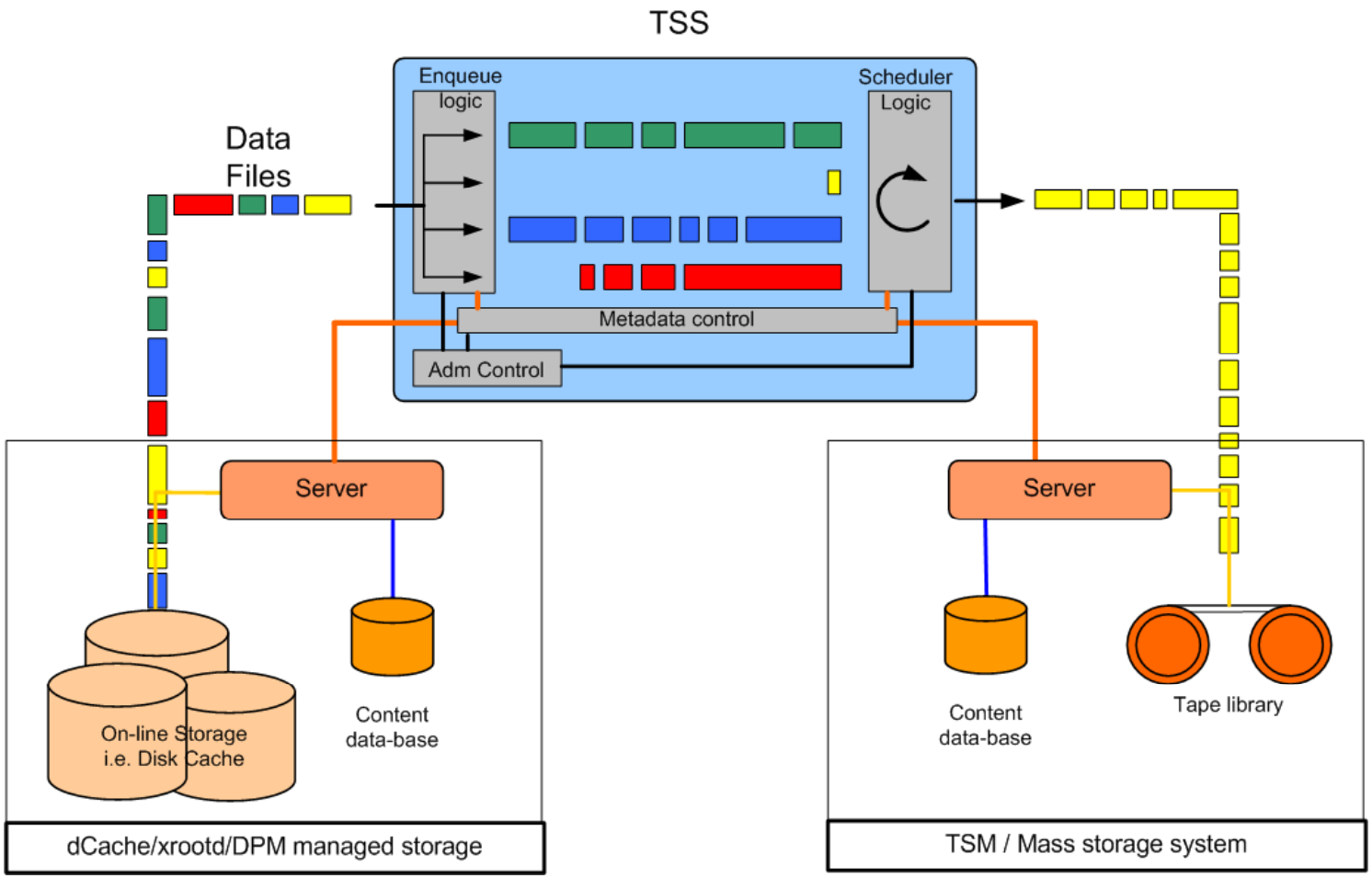
TSS, an interface to TSM for storage-management middleware

and

the GridKa mass storage system

Jos van Wezel / GridKa

[Tape|TSM] staging server



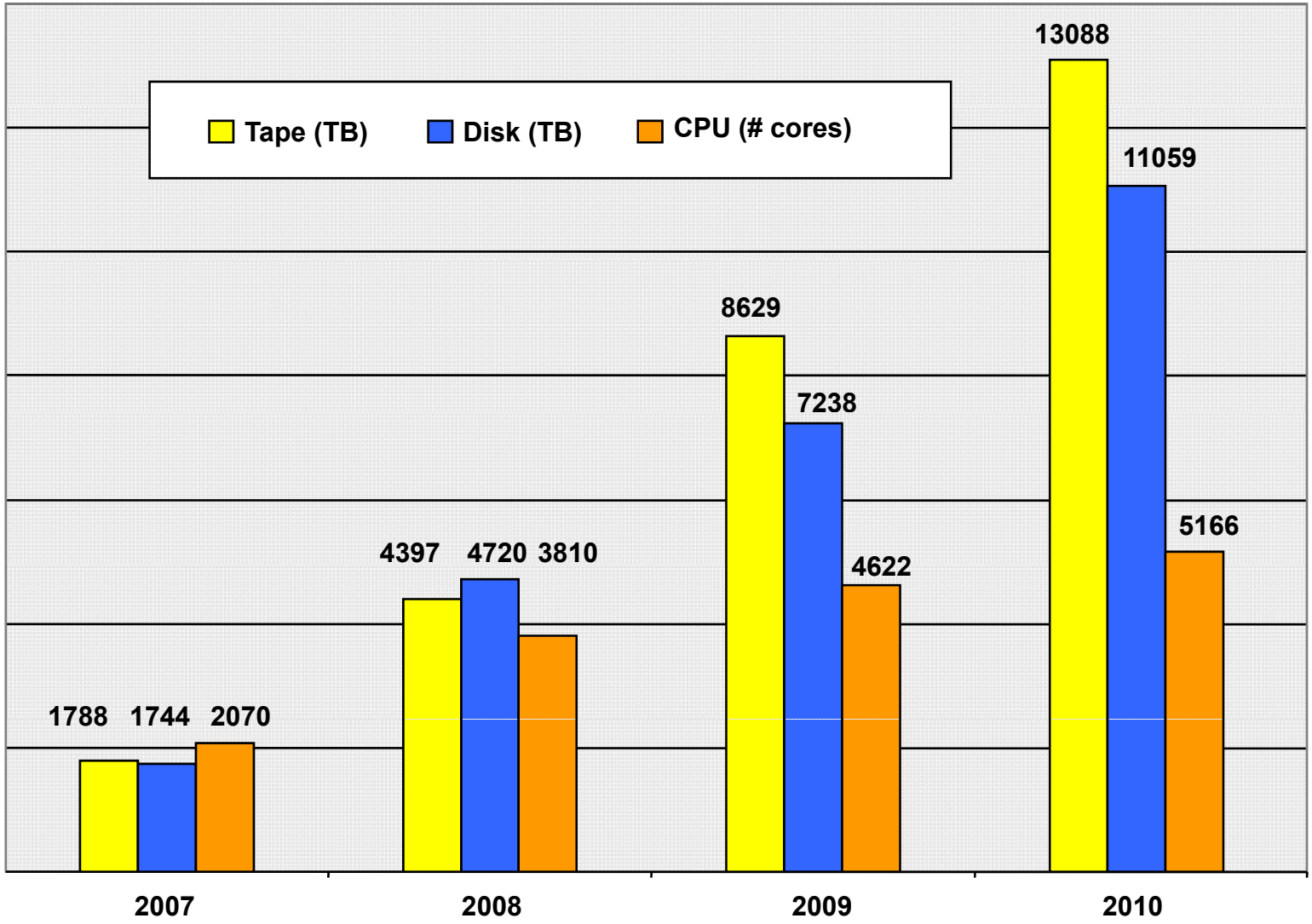
- *Introduction*
- *Grid storage and storage middleware*
- *dCache and TSS*
- *TSS internals*
- *Conclusion and further work*

FZK/GridKa

The GridKa project at Research Center Karlsruhe:

- Project start in 2002
- Construct a compute cluster for use in computing Grids
- Current capacity: ~3000 cores, 1.5 PB disk, 2 PB tape
- Commenced as central compute service to the particle physics community in Germany
- Now serves several other Virtual Organizations (VOs)
- Main focus at the moment is data processing for the LHC (Large Hadron Collider)
  - LCG: LHC Computing grid
  - WLCG: World wide LCG

Planning numbers



Planned transfers rates to and from tape

T0: CERN/data source

T1: GridKa/data reprocessing

T2: data analysis

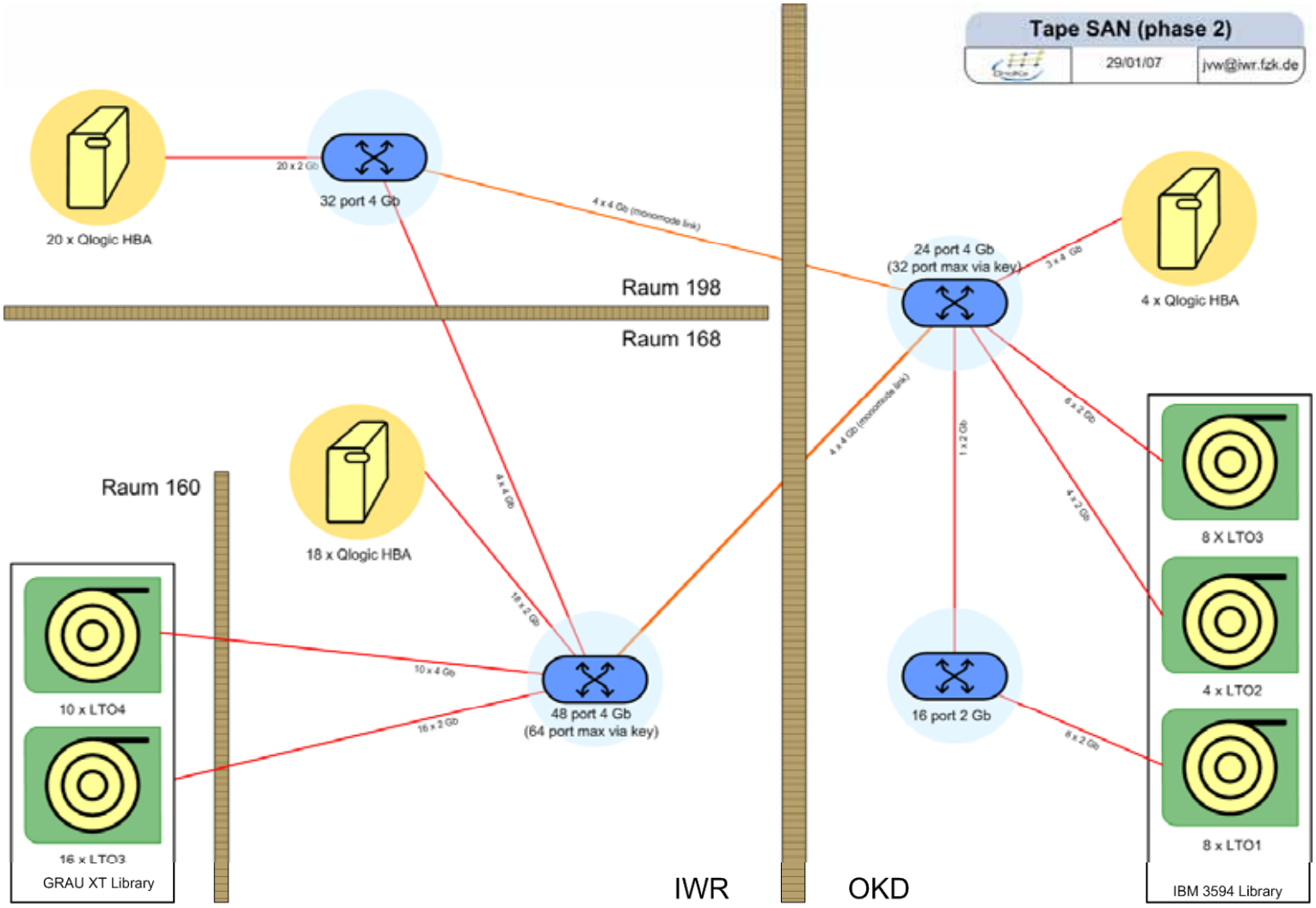
	T0⟶T1	T2 ⟶ T1	T1 ⟶ T1	T1 ⟶ T2	T1⟵T1
VO	**MB/s**	**MB/s avg.**	**MB/s In**	**MB/s avg.**	**MB/s out**
ALICE	14.9	44.1	15.2	17.8	16.0
ATLAS	88.2	34.1	93.3	99.0	110.3
CMS	26.3	7.0	91.2	63.0	48.0
LHCb	6.3	0.2	20.4	11.4	18.1
SUM	**135.6**	**85.4**	**220.1**	**191.2**	**192.4**

Tape hardware

- Libraries
  - GRAU XL (~600 slots / hour)
  - IBM 3592 (~700 slots / hour)

- Drives
  - LTO2
    - 8 installed
  - LTO3
    - 24 installed
    - 8 older drives (1 replaced)
    - IO rate 42 MB/s observed
    - mostly less ~25
    - could be 80?

Tape SAN


Tape SAN (phase 2)
29/01/07
jvw@iwr.fzk.de
20 x Qlogic HBA
32 port 4 Gb
Raum 198
Raum 168
24 port 4 Gb
(32 port max via key)
4 x Qlogic HBA
Raum 160
18 x Qlogic HBA
8 X LTO3
4 x LTO2
8 x LTO1
10 x LTO4
16 x LTO3
GRAU XT Library
48 port 4 Gb
(64 port max via key)
16 port 2 Gb
IBM 3594 Library
IWR
OKD

- *Introduction*
- *Grid storage and storage middleware*
- *dCache and TSS*
- *TSS internals*
- *Conclusion and further work*

Accessing ‘Grid’ data

- Network access via uniform protocol: SRM
- SRM is software on top of storage management system
- Connect grid storage islands through a grid Storage Service
- Network interface to storage
- You provide data and a combination of:
  - Acess protocol: dcap, rfio, nfs
  - Retention policy: recover time needed (custodial, replica, output)
  - Access latency: nearline, online, (offline) / tape, disk, shelf
- Location management takes care of duplication
- The storage system (dCache) does the rest for you

WLCG storage

- WLCG uses the Storage Resource Manager
- From SRM the following storage classes are inferred for the WLCG data management:
  - T1D0: files moved to tape directly
  - T1D1: files migrated to tape but kept on disk as long as there is space or ‘pin’ times out.
  - T0D1: files on disk only
- Tape (or mass storage system) is considered ‘custodial’ storage. Meaning: data is to be kept indefinitely.
- We do not delete data on tape.

dCache

Think of it a as a filesystem. It gives you:

A: interface to the grid via SRM data storage interface
- data placement based on SRM classes
- disk, tape or both

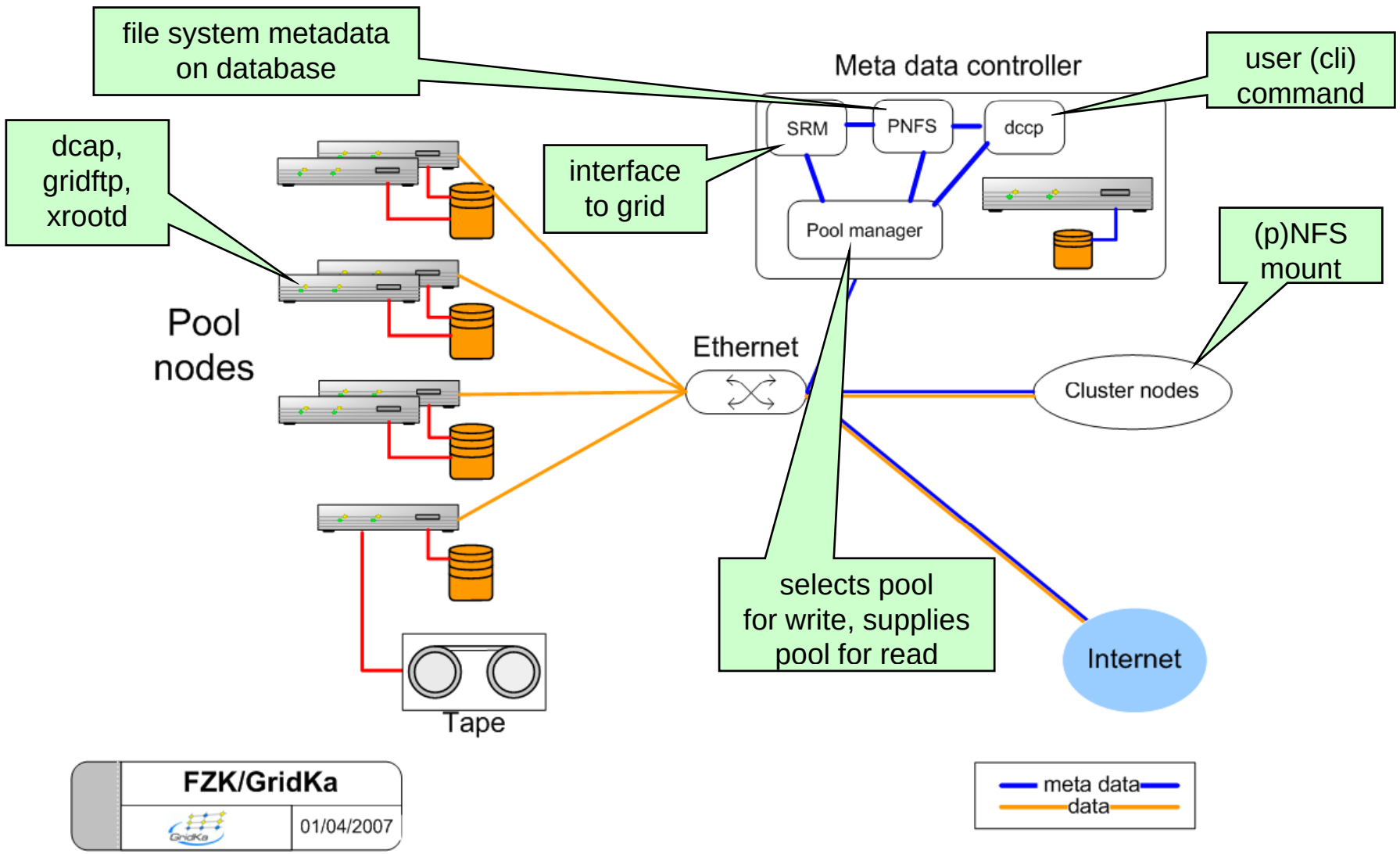
B: manages disk storage
- ‘global’ name space (within the domain)
- load balancing
- access control (via certificates)

C: backend to write to and read from permanent storage (tape or other Mass Storage System)
- GridKa is ‘custodian’ for, part of, the raw detector data
- Some computed data also goes to tape

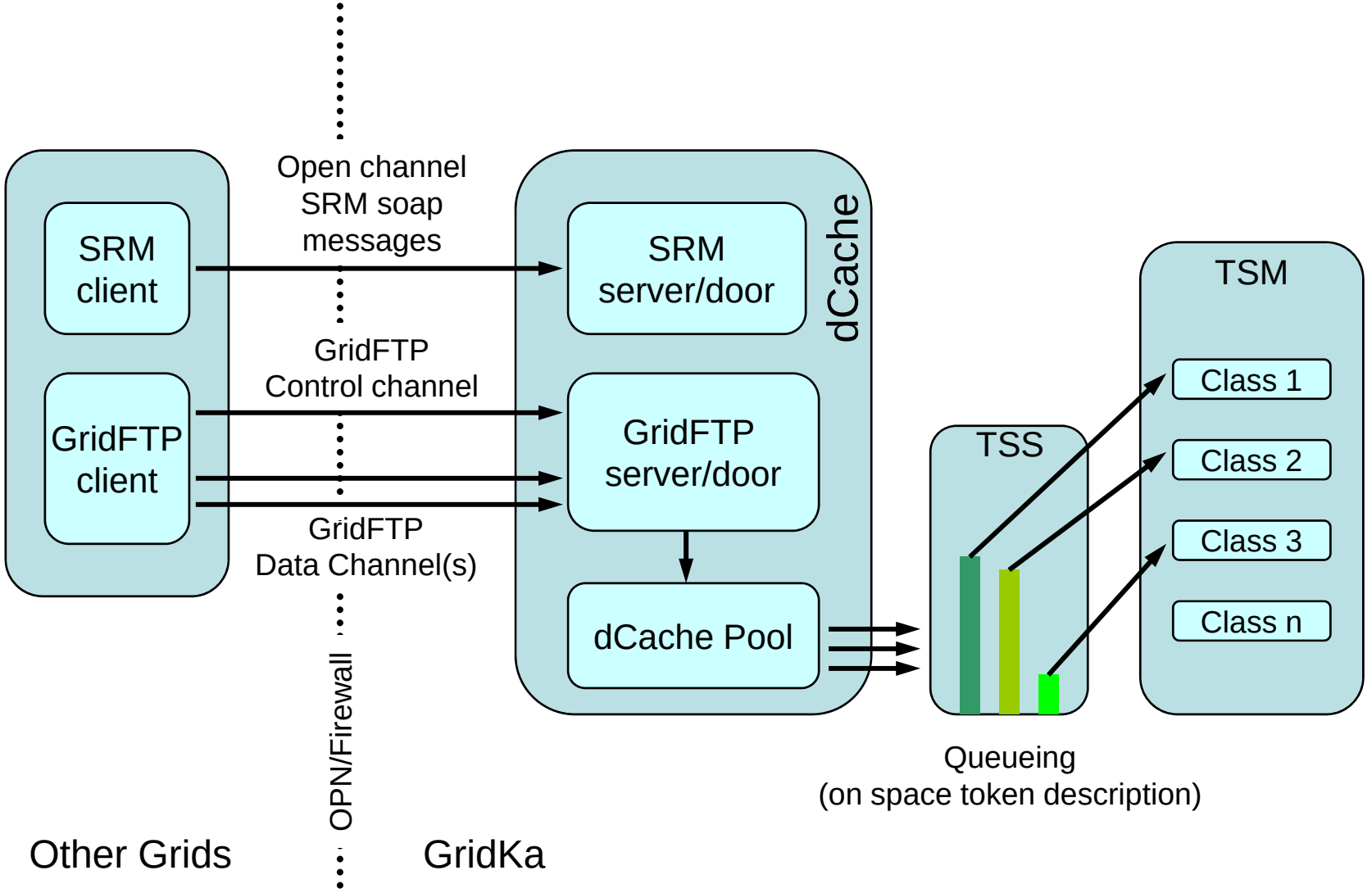
Disk pool managers

- dCache
  - interfaced with TSS/TSM, HPSS, ENSTOR, OSM
- DPM : the disk pool manager
  - has no mass / archival storage support yet
- Storm: an SRM on top of GPFS
  - efficiency of interface to TSM is investigated
- Xroot: in use at particle physics labs
  - HPSS, TSS/TSM

dCache components



Data flow to the T1 with dCache and SRM

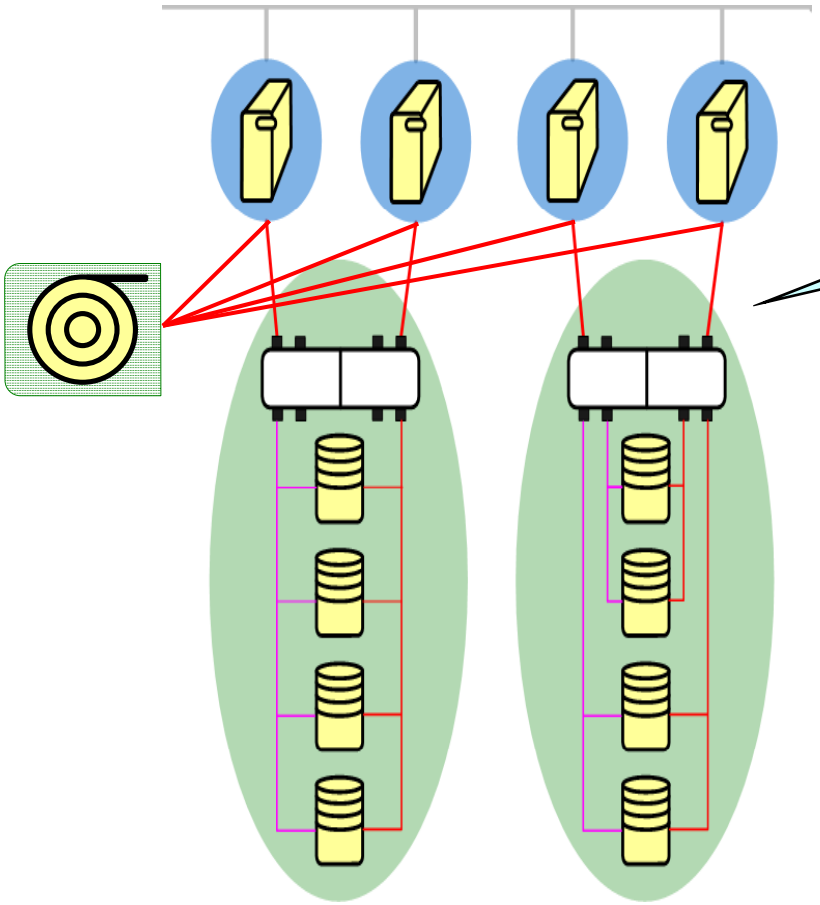


Summary 1

- SRM is the entry point for data exchange between grid sites.
- Disk pool managers offer an SRM interface to disk (and tape) storage
- The disk pool manager in use at GridKa is dCache
- dCache intelligently places files on distributed disk storage and coupled mass storage (tape)

- ☑ *Introduction*
- ☑ *Grid storage and storage middleware*
- ☐ *dCache and TSS*
- ☐ *TSS internals*
- ☐ *Conclusion and further work*

dCache disk pools and pool nodes



Disk pools

Disk pools on dCache

- trigger a callout
  - number of files
  - total size
  - wait time
- the callout runs on recalls and on migrate requests
- synchronous to dCache activities

Calling sequence

dCache provides

- physical filename: name on the disk pool
- unique ID: pnfsid
- storage info: detailed variables
  - logical file name: name as seen by user
  - administrator defined ‘tag’
    - tag is set per directory
    - follows parent
    - e.g.

```
/pnfs/gridka/vos/ atlas/disk
                        /tape
                 /cms/disk
                        /tape
```

dc_atlas

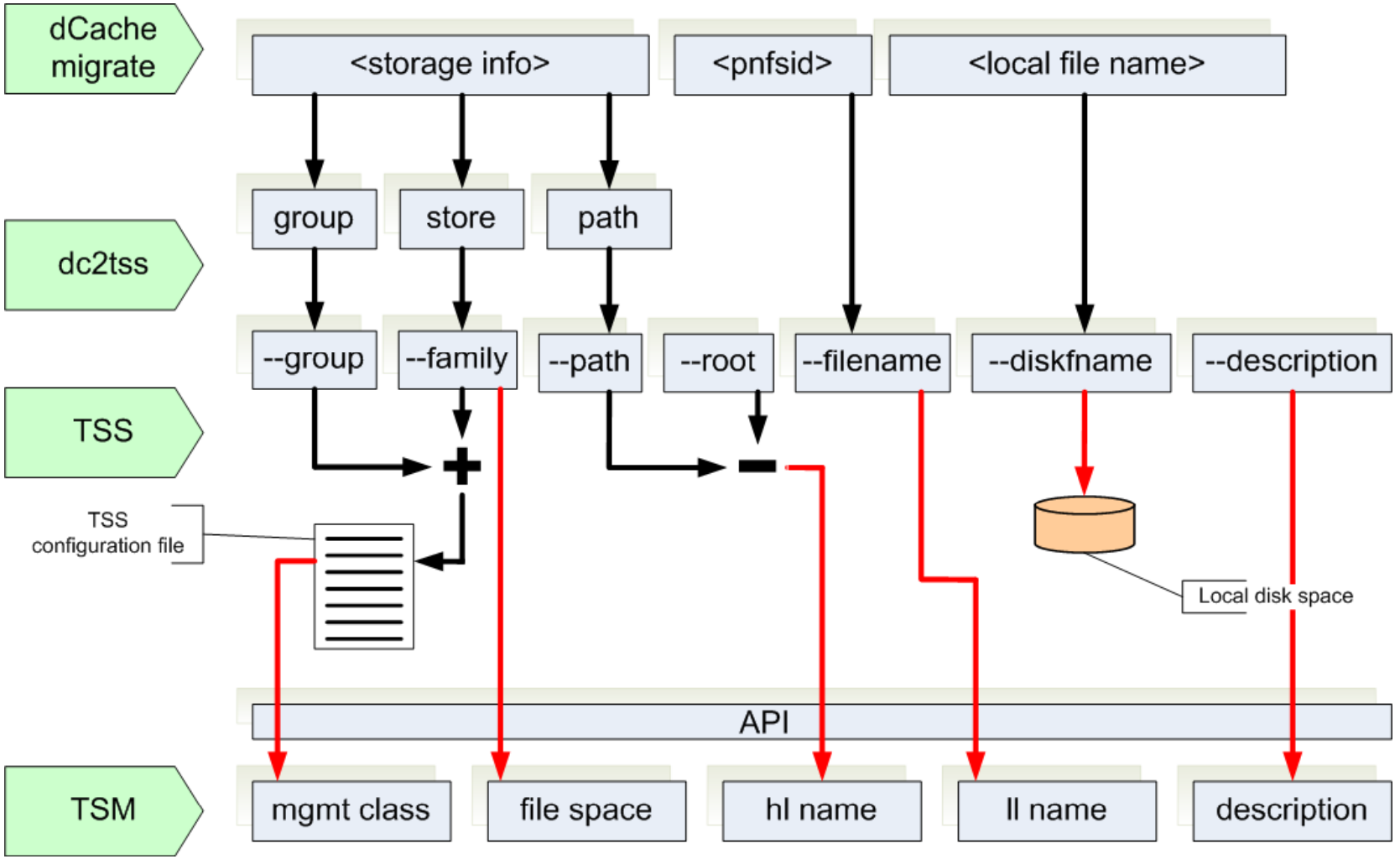
dc_cms

- callout runs UNIX script

Directory tag is set here

Output on callout

dCache to TSS / TSM



- ☑ *Introduction*
- ☑ *Grid storage and storage middleware*
- ☑ *dCache and TSS*
- ☐ *TSS internals*
- ☐ *Conclusion and further work*

dCache to TSM previously

Original TSM backend

- 1 file results in 1 store or recall
  - large overhead. Session startup time takes inordinate amount of time
  - when storage agents are used: TSM volume selection algorithm starts cartridge juggle. Efficiency nears zero.
- No data classes
  - everything goes to one and the same tape
  - no policies or quota for particular data
- On recalls
  - no control over tape file order: recalls will be virtually impossible)
  - dCache cannot provide queues (for recalls)

Remember: tape allows only sequential access!

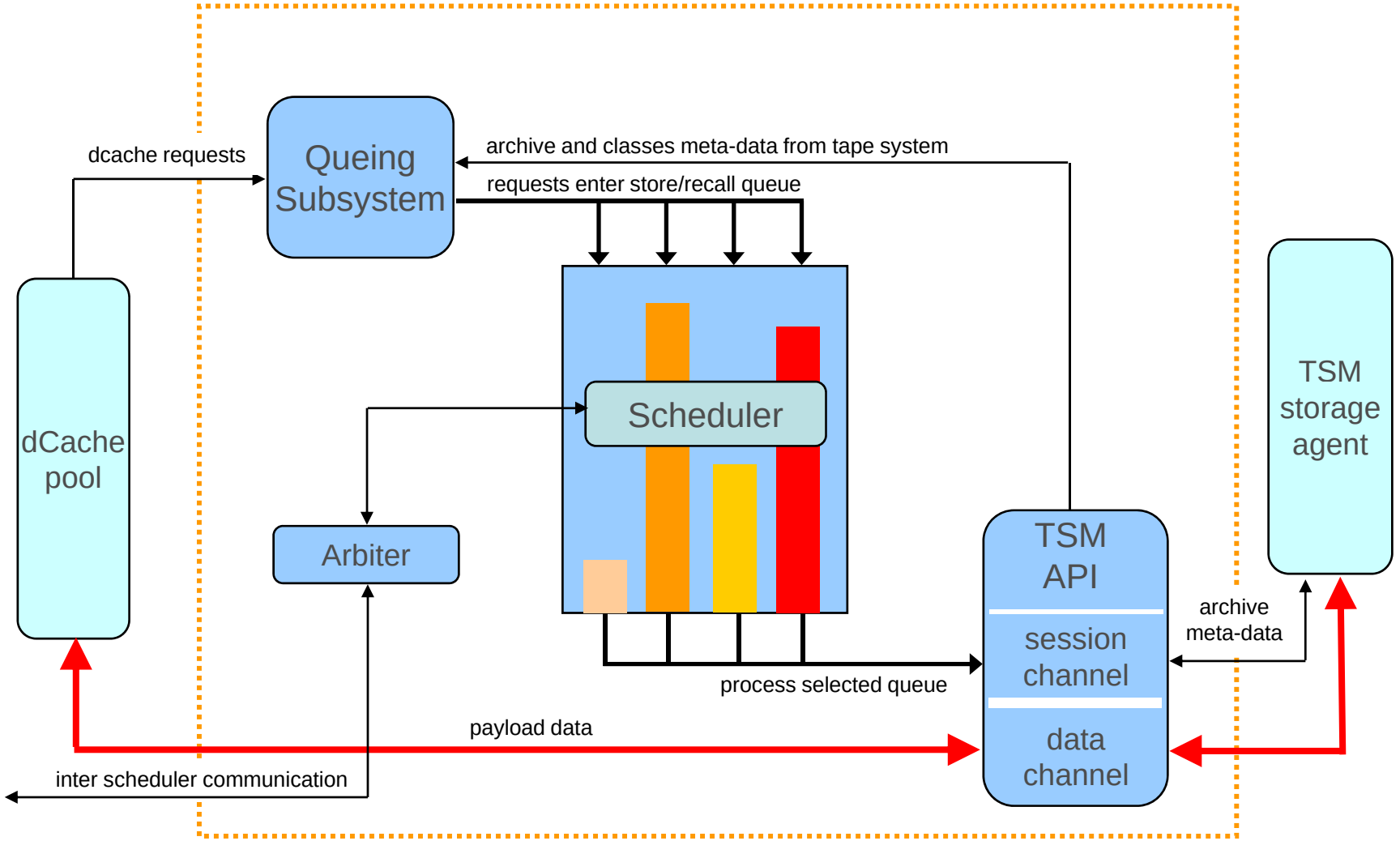
Requirements for dCache to tape interface

- Use available TSM base at Forschungszentrum Karlsruhe
- Improve throughput
- Reduce number of tape mounts
- Use different tape sets for different data classes

TSS properties

- Interface directly with TSM via the API
- Fan out for all dpm/dCache to tape activities
  - mutiple operations: recall, migrate, rename, delete, query
- Runs on the TSM clients, storage agent or on the server proper
- Plug-in replacement for the TSM backend that comes with dCache
- Sends different type of data to different tape sets
- Two level data classes (with dCache)
- Queues requests on tape sequence order
- No persistent state is kept
- Allows to store an exact image of the logical global name space on tape
- command line interface to set running parameters, monitor the processing, run db queries (think of it as an alternative dsmc)

TSS command and data flow



Major components

Queuing engine
- data management: input output files, set data classes
- enqueue: creates queues

Scheduler
- Select queue to process based on trigger
- Starts threads to process queue(s)

TSM DMI interface
- handle sessions
- queries TSM DB
- setup data transfers
- sends and receives data

Admin interface
- separate thread to return status information of queues and clients
- stopping and starting the subsystems
- changing running parameters

TSS Scheduler

- Scheduler starts request processing per queue
- More than one queue may be processed concurrently (allows for ‘big’ hosts that handle 2 or more tape drives)
- Queue is determined ‘runnable’ based on:
  - time: elapsed time since first job entry)
  - size: summation of the number of bytes of all files
  - length: number of requests in the queue
- Communicates with arbiter to prevent ‘drive collisions’ in next version

TSS Queue engine

- On (recall) entry query the TSM DB
  - if ‘object’ exists, its id is put in the queue
- On (migrate) entry select management class
  - Unknown classes are not migrated
- Renames, deletes etc are forwarded directly
- Caller waits until TSS returns with the data or a non-zero error code

DMI API

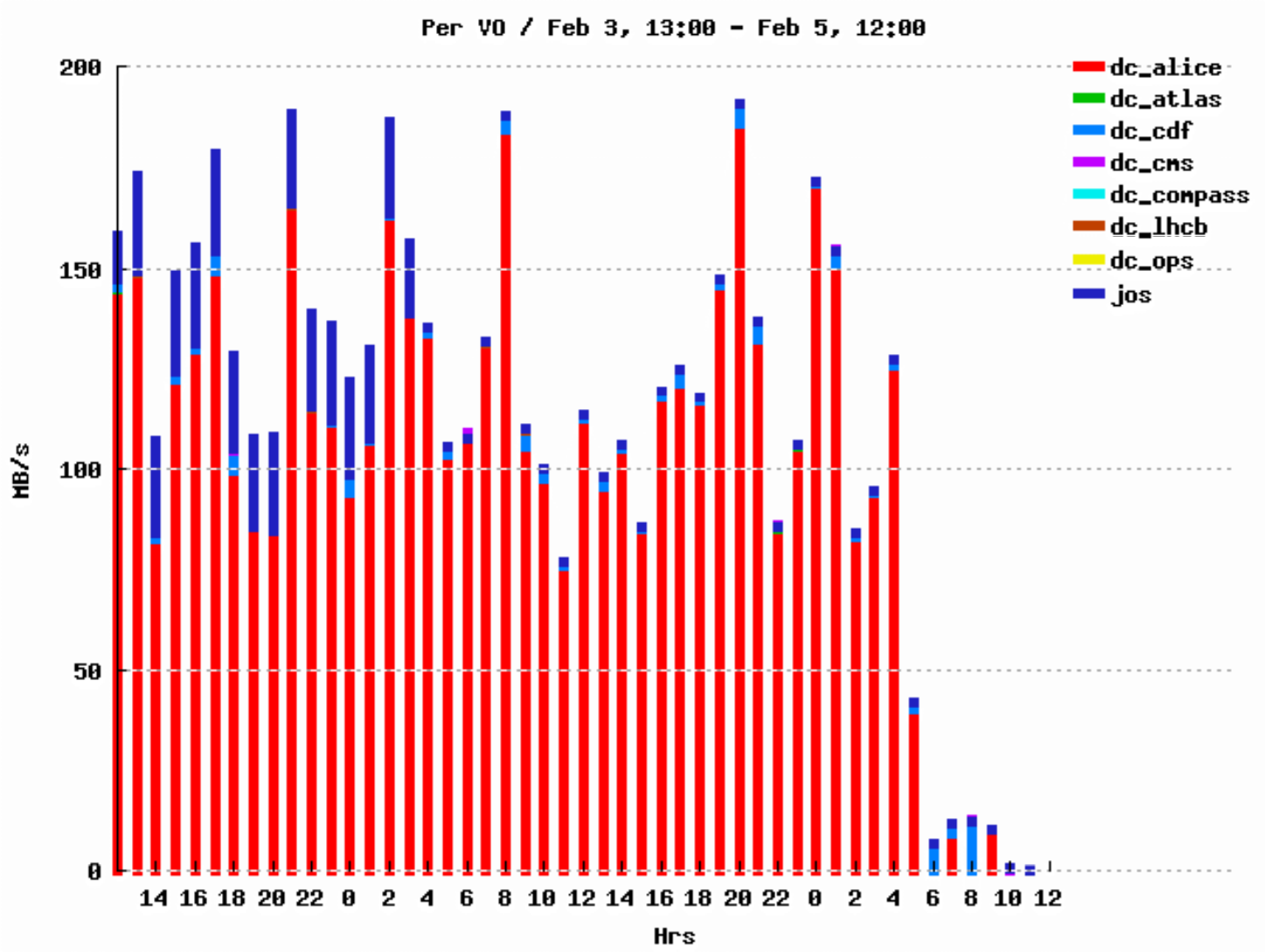
Wrapper around the API

- Library of the library
- Keeps track of open sessions/handles
- Simplifies queries and Send/Get data calls
- Utility functions dmi_query_mc(), dmi_log, dmi_session_info() etc.
- Callbacks separate API lib from rest of the code
- Example: data Get becomes:

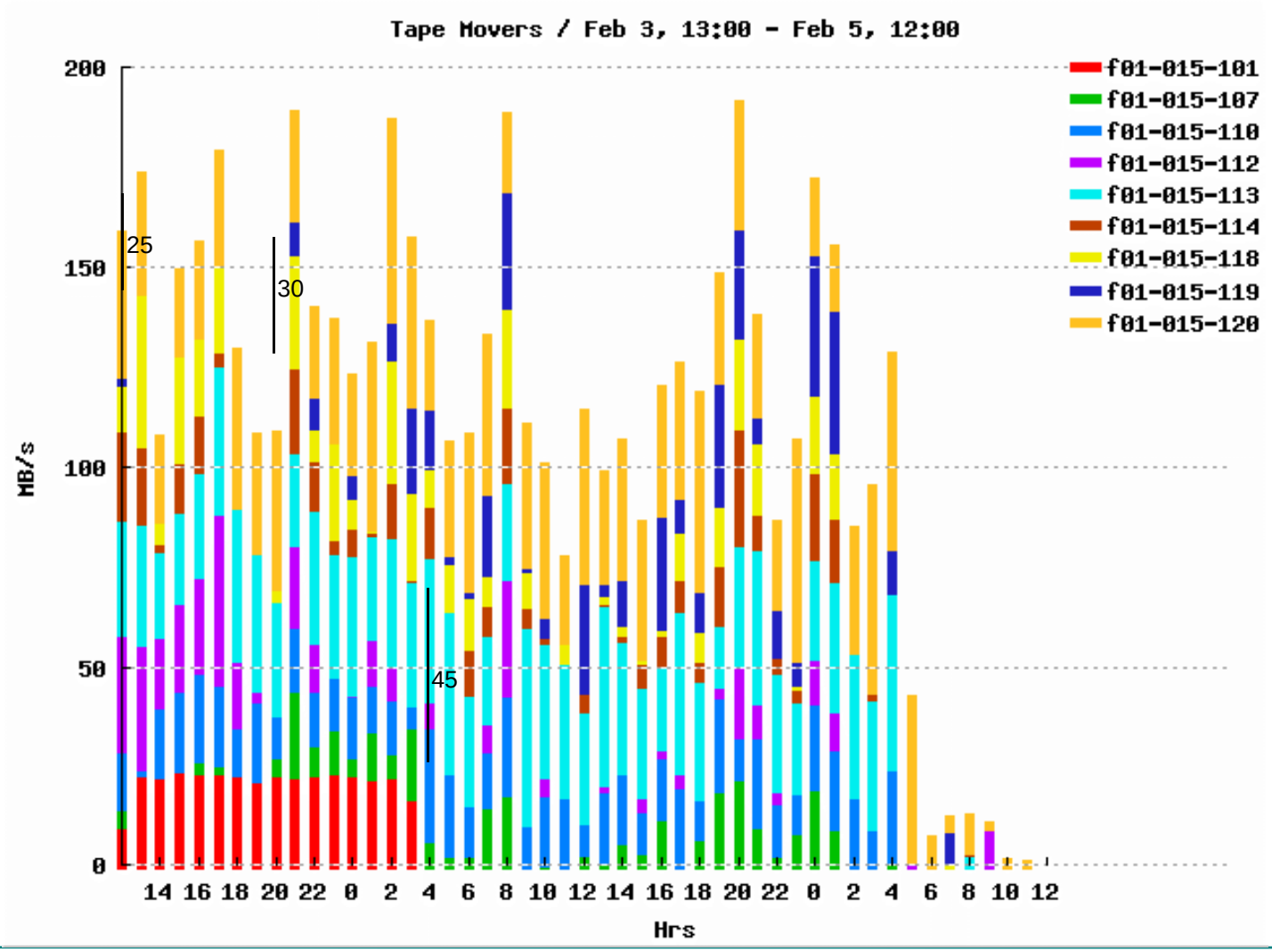
```
if( dmi_init(p1 *, …) == 0)
   if (dmi_query(p1 *, …) ==0)
       if (dmi_get(p1 *, …) == 0)
              return(0);
```

- Regretfully no API support for library and or volume handling.

Tapeview (VO's)



Tapeview (storage agents)



- ☑ *Introduction*
- ☑ *Grid storage and storage middleware*
- ☑ *dCache and TSS*
- ☑ *TSS internals*
- ☐ *Conclusion and further work*

Current issues

- ‘Communication lost’ errors:
  - (ANS1026E (RC136) The session is rejected: There was a communications protocol error.
- Multiple TSM clients talking to a single TSM Agent allocate ‘multiple’ tape drives
  - multiple clients now talk to a single STA
- No load balancing, diversion for more then 1 library
- No upstream error detection: library down, no scratch tapes left, no more drives available etc.
- Interface dCache (java) to TSS is a shell script: i.e. limited signal processing.

In progress

- Queue process arbitration via volume pegboard
  - reduce concurrent drive access
  - improve recall throughput
  - synchronous updates would lame operations
- Concurrent queue processing
  - configurable number of queues
  - processed concurrently on a single host
- Per queue scheduling parameters
  - different queue triggers for read or write queues
  - finer tuning of write queues
- Remote queue entry
  - clients connect to a central TSS
  - groups requests (esp. needed for recall)
- Support for Multiple Tape Libraries

Conclusions

- TSM can handle > 300 MB/s
- TSS is working as expected
- Tape speed not the expected rates
- Need to find out the access pattern/tape mounts
- Need to have better error recovery
- Configuration is eeeeh…. pretty complex
- It would be better if TSM could do this

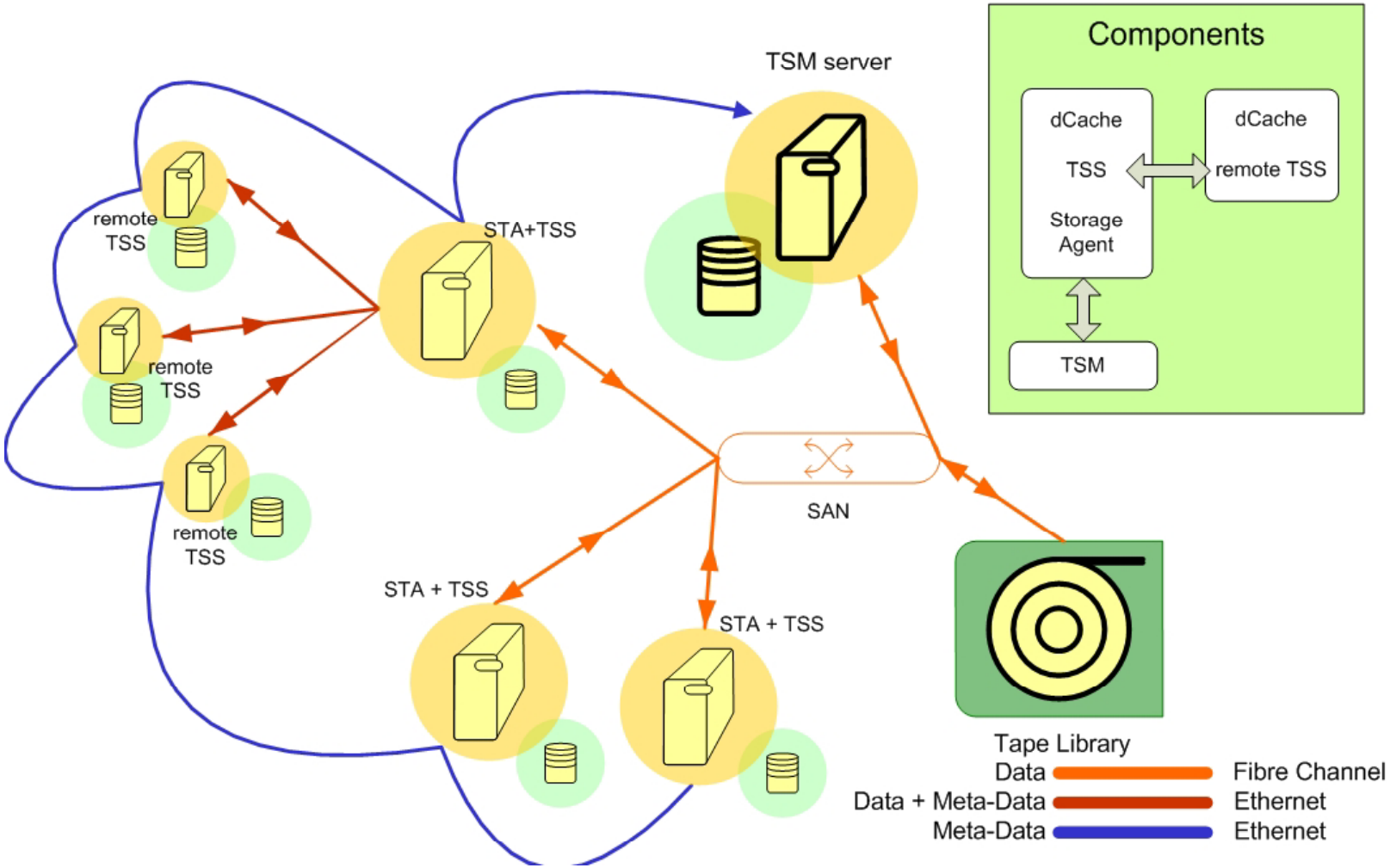
- *Introduction*
- *Grid storage and storage middleware*
- *dCache and TSS*
- *TSS internals*
- *Conclusion and further work*

Many thanks to:
Dorin Lobontu, Stephanie Boehringer, Silke Halstenberg, Doris Ressmann,

You probably have some questions?

Spare slides

Data Flow – data and meta data



storage on the grid: SRM

- hide the complexity of the local storage at a site with a uniform interface: SRM
- connect grid storage islands through a grid Storage Service
- SRM is software on top of storage management system
- provide dynamic space allocation/reservation and file management: **space management functions**
- provide dynamic information regarding storage and files: **status functions**
- takes care of authorization and authentification (in the dcache SRM via the gPlazma cell): **permission functions**
- transfer protocol negotiation: **data transfer functions**
- and many other things …..