DKRZ Tech Talk: searching and optimizing the retrieval of data from tape

12th July 2023, hybrid

Daniel Heydebreck

Deutsches Klimarechenzentrum (DKRZ)

Important links

- Q&A document: https://pad.gwdg.de/XMk-C0ikTiaFwCQzlLTpYA#
- HSM Documentation: https://docs.dkrz.de/doc/datastorage/hsm/index.html
- Questions / Feedback / Wishes: support@dkrz.de

HSM tools

- `slk`:
  cli provided by StrongLink (`module load slk`)
- `slk_helpers`:
  cli provided by DKRZ (`module load slk_helpers`)
  new release (not default): `slk_helpers/1.9.5`

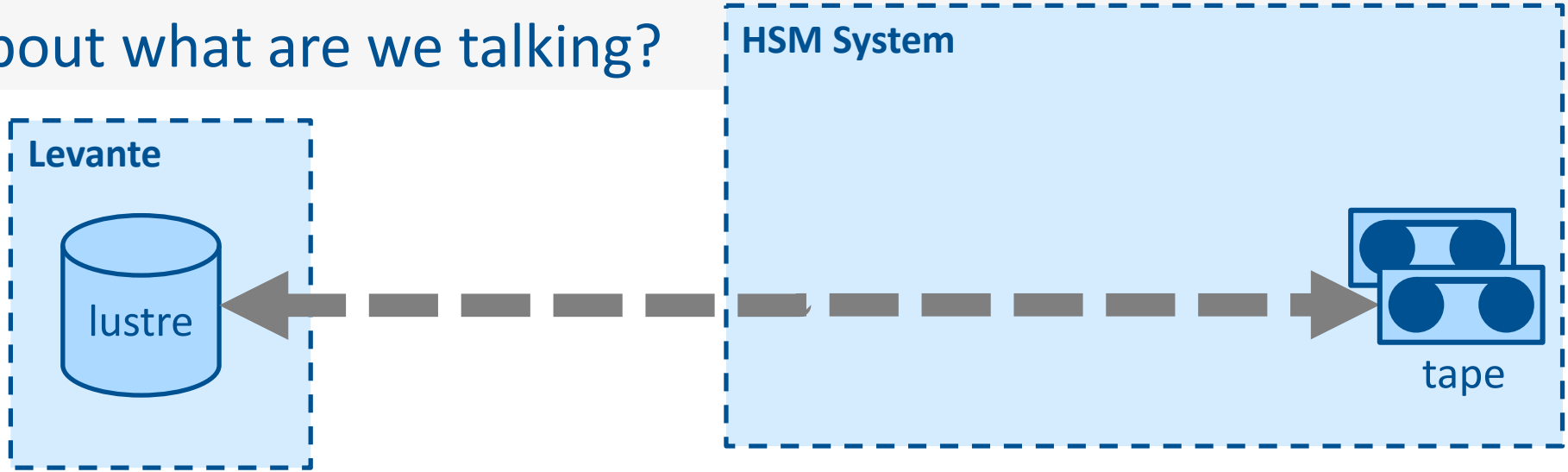
  preferably use this version
- `Pyslk`:
  Python wrappers for `slk` and `slk_helpers`
  - v1.8.1 via `module load python3/2023.01-gcc-11.2.0`
    (rather simple/inconvenient wrappers, but stable)
  - v1.9.1 not installed on Levante yet; manual installation
    https://hsm-tools.gitlab-pages.dkrz.de/pyslk/availability.html#download
    (testing phase but rather stable; many convenient wrappers)

    preferably use this version

Overview

- Tape archive in general
- Basic Retrievals
- Optimizing Retrievals
- Searching Files

About what are we talking?

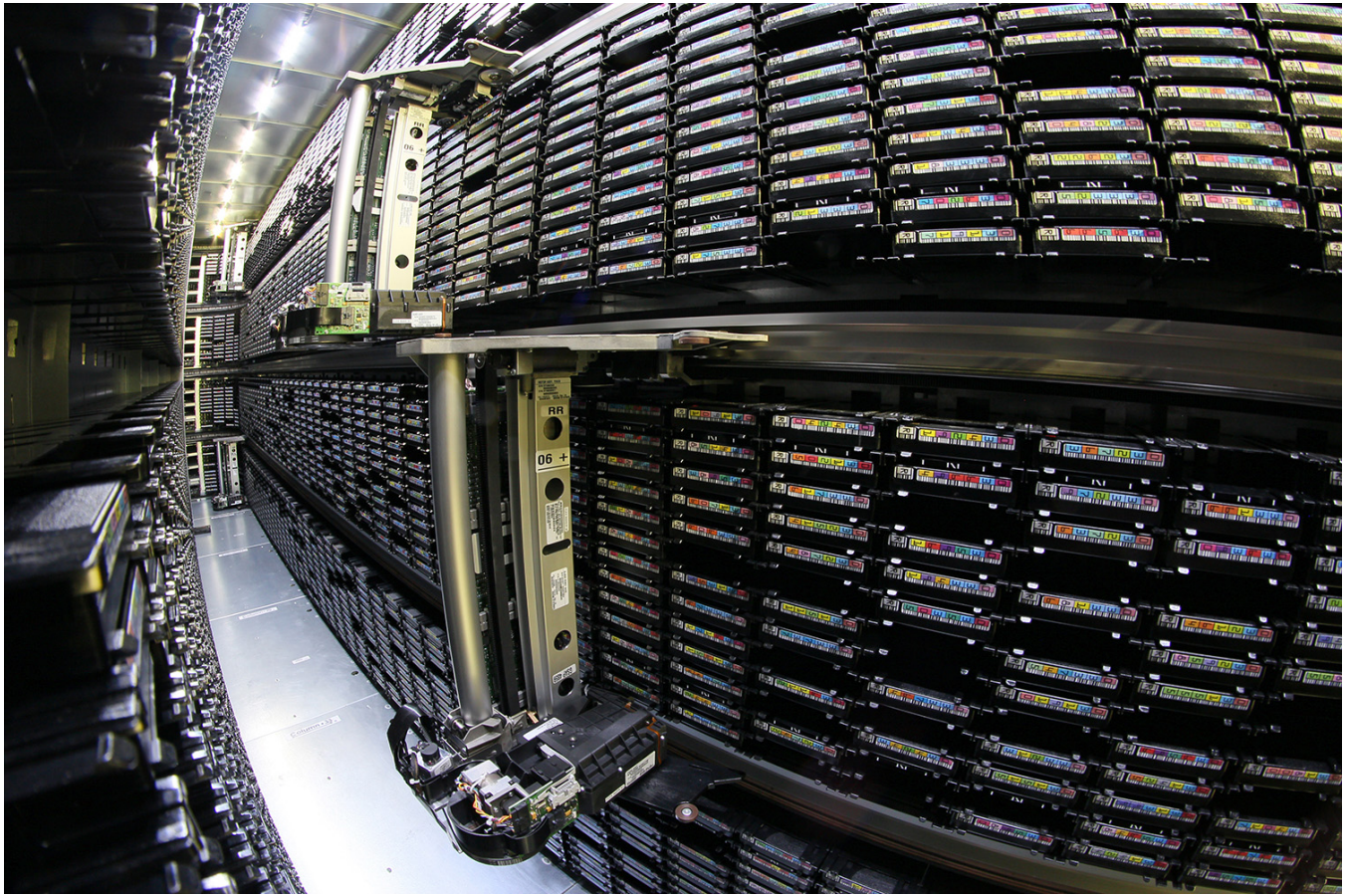


Why storage tape and not buy more harddisks?

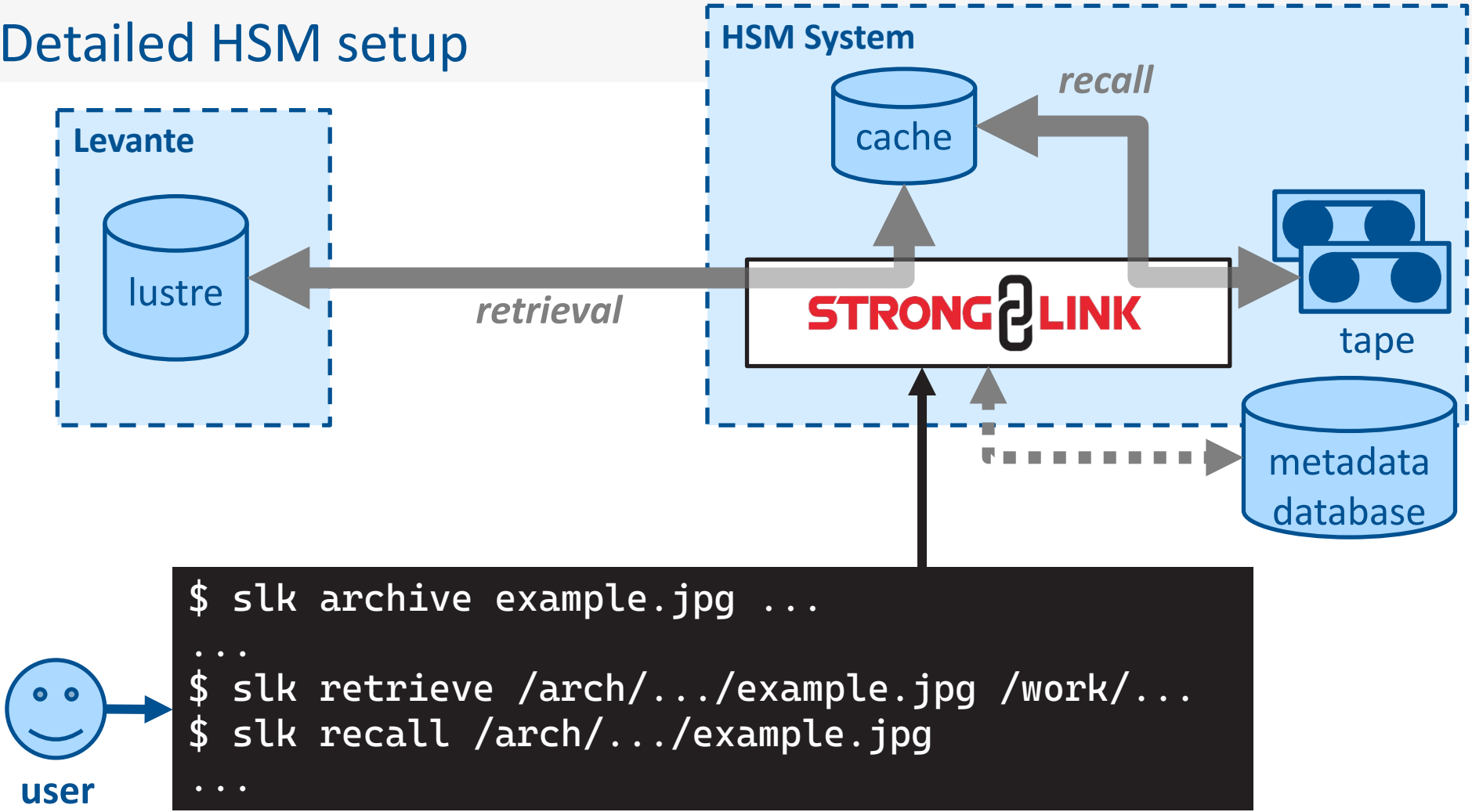
- cheaper (pay less EUR/TB)
- lower energy consumption

But: slower access compared to harddisk

tape library from inside



Detailed HSM setup




```
$ slk archive example.jpg ...
...
$ slk retrieve /arch/.../example.jpg /work/...
$ slk recall /arch/.../example.jpg
...
```

Practice session Ia: get files from tape

- we need two files:

```
/arch/bm0146/k204221/iow/INDEX.txt
/arch/bm0146/k204221/iow/iow_data_001.tar
```

- see whether files in cache or not (=> only on tape):

```
slk list /arch/bm0146/k204221/iow/...
slk_helpers iscached /arch/bm0146/k204221/iow/...
```

- commands also work with search ids:

```
slk list <search id>
slk_helpers iscached --search-id <search id>
```

Practice session Ia: get files from tape – output (I)

```
$ slk list /arch/bm0146/k204221/iow
-rwxr-xr-x- k204221     bm0146          1.2M   10 Jun 2020 08:25 INDEX.txt
-rw-r--r--- k204221     bm0146         19.5G   05 Jun 2020 17:36 iow_data2_001.tar
-rw-r--r--- k204221     bm0146         19.0G   05 Jun 2020 17:38 iow_data2_002.tar
-rw-r--r--- k204221     bm0146         19.4G   05 Jun 2020 17:38 iow_data2_003.tar
-rw-r--r--t k204221     bm0146         19.3G   05 Jun 2020 17:40 iow_data2_004.tar
-rw-r--r--t k204221     bm0146         19
-rw-r--r--t k204221     bm0146          7
-rw-r--r--t k204221     bm0146        186
-rw-r--r--t k204221     bm0146         24
-rw-r--r--- k204221     bm0146          4
-rw-r--r--t k204221     bm0146         10.5G   05 Jun 2020 19:46 iow_data4_002.tar
-rw-r--r--- k204221     bm0146         19.5G   10 Jun 2020 08:21 iow_data5_001.tar
-rw-r--r--t k204221     bm0146         19.0G   10 Jun 2020 08:23 iow_data5_002.tar
-rw-r--r--t k204221     bm0146         19.4G   10 Jun 2020 08:23 iow_data5_003.tar
-rw-r--r--- k204221     bm0146         19.3G   10 Jun 2020 08:24 iow_data5_004.tar
-rw-r--r--t k204221     bm0146         19.1G   10 Jun 2020 08:25 iow_data5_005.tar
-rw-r--r--t k204221     bm0146          7.8G   10 Jun 2020 08:25 iow_data5_006.tar
-rw-r--r--t k204221     bm0146         19.5G   05 Jun 2020 17:53 iow_data_001.tar
-rw-r--r--t k204221     bm0146         19.0G   05 Jun 2020 17:53 iow_data_002.tar
-rw-r--r--t k204221     bm0146         19.4G   05 Jun 2020 17:56 iow_data_003.tar
-rw-r--r--t k204221     bm0146         19.3G   05 Jun 2020 17:56 iow_data_004.tar
-rw-r--r--t k204221     bm0146         19.1G   05 Jun 2020 17:58 iow_data_005.tar
-rw-r-----t k204221     bm0146          7.8G   05 Jun 2020 17:57 iow_data_006.tar
Files: 23
```

Caching status is highlighted in red:

- 't': has to be copied from tape
- '-': available in the cache

Practice session Ia: get files from tape – output (II)

```
$ slk_helpers iscached -R /arch/bm0146/k204221/iow
Not all files are cached.

$ slk_helpers iscached -R /arch/bm0146/k204221/iow -v
/arch/bm0146/k204221/iow/iow_data_002.tar is not cached
/arch/bm0146/k204221/iow/iow_data_001.tar is not cached
/arch/bm0146/k204221/iow/iow_data5_006.tar is not cached
/arch/bm0146/k204221/iow/iow_data5_005.tar is not cached
/arch/bm0146/k204221/iow/iow_
/arch/bm0146/k204221/iow/iow_
/arch/bm0146/k204221/iow/iow_
/arch/bm0146/k204221/iow/iow_
/arch/bm0146/k204221/iow/iow_
/arch/bm0146/k204221/iow/iow_
/arch/bm0146/k204221/iow/iow_
/arch/bm0146/k204221/iow/iow_data5_003.tar is not cached
/arch/bm0146/k204221/iow/iow_data5_002.tar is not cached
/arch/bm0146/k204221/iow/iow_data2_006.tar is not cached
/arch/bm0146/k204221/iow/iow_data2_005.tar is not cached
/arch/bm0146/k204221/iow/iow_data2_004.tar is not cached
Number of files stored in the cache: 7/23
```

three different verbose modes:

- 'no -v': print one summary line
- '-v': print files which are not cached
- '-vv': print all files and their caching status

Practice session Ib: get files from tape

- if cached => `slk retrieve <src> <dst>`
  => get files in seconds
- else: `slk retrieve ...` or `slk recall <src>`
  => wait for tape
- The <src> either is
  - a path to a file
  - a path to a folder (append –R)
  - a search id

Practice session Ic: get files from tape

- recalls and retrievals might fail
- possible reasons: please see *Known Issues* in the doc
- error details might be in the log: `~/.slk/slk-cli.log`
- example log content

```
2023-07-06 20:55:04 xU22 679118 INFO  Executing command: "list /dkrz_test/netcdf/20230630b"
2023-07-06 20:55:14 xU22 679205 INFO  Executing command: "list /dkrz_test/netcdf/20230630b/a"
2023-07-06 20:55:55 xU22 679236 INFO  Executing command: "search {"path": {"$gte":
                                "/dkrz_test/netcdf/20221103a"}}"
2023-07-06 20:56:06 xU22 679326 INFO  Executing command: "search {"path": {"$gte":
                                "/dkrz_test/netcdf/20230630b/a"}}"
2023-07-06 21:03:22 xU22 681556 INFO  Executing command: "list 466066"
2023-07-09 16:40:14 xU22 2262 INFO  Executing command: "list /arch/bm0146/k204221/iow"
2023-07-09 16:40:24 xU22 2262 ERROR Unhandled error occurred, please check logs
2023-07-09 16:40:24 xU22 2262 ERROR Exiting due to error:
io.ktor.network.sockets.ConnectTimeoutException: Connect timeout has expired
[url=https://archive.dkrz.de//api/v2/whoami, connect_timeout=unknown ms]
        at io.ktor.client.features.HttpTimeoutKt.ConnectTimeoutException(HttpTimeout.kt:183)
          ~[slk-cli-tools_3.3.90.jar:?]
```

date time hostname pid

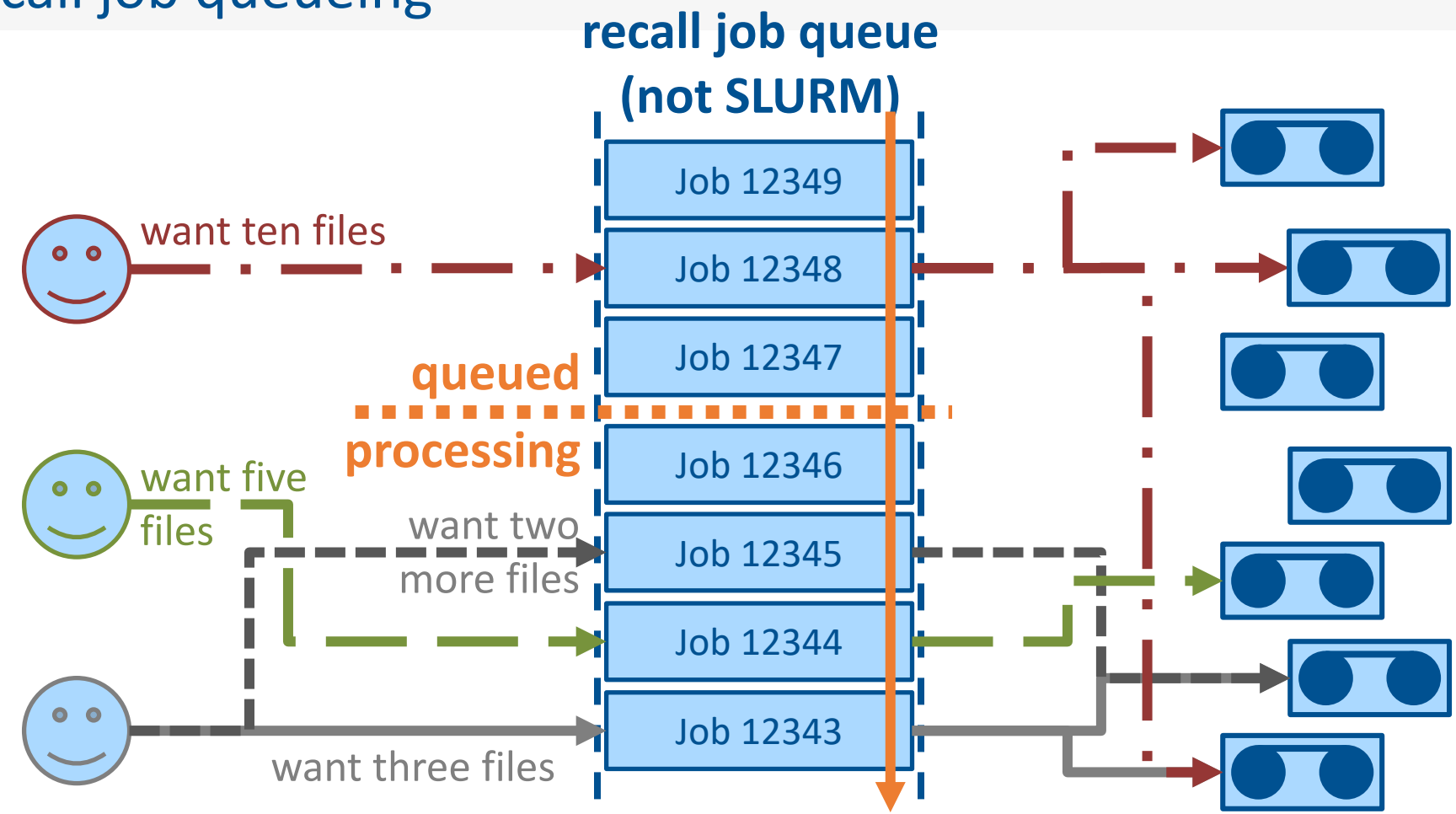
The end

If StrongLink was running perfectly.

Note

This should not indicate that StrongLink is a bad system. StrongLink has more features than HPSS and with respect to some admin features it is easier to maintain. However, certain basic tasks are not working smoothly. One could discuss whether a system with more features has a higher complexity and, naturally, has more bugs.

Recall job queueing



Practice session IIa: checking recall jobs

- How do I find out the job id?
  - slk log file: `~/.slk/slk-cli.log`
  - log entry contains process id and hostname:

```
[DATE TIME] [HOSTNAME] 2244893 INFO Executing command: "recall -R 466080"
[DATE TIME] [HOSTNAME] 2244893 INFO Created copy job with id: '137871'
                for - "recall -R 466080"
```

- How do I find out what my job is doing?

```
$ slk_helpers job_status 137871
SUCCESSFUL
```

https://docs.dkrz.de/doc/datastorage/hsm/slk_helpers.html#job-stati

job stati

Practice session IIb: checking recall jobs

- How do I get the queue status?

```
$ slk_helpers job_queue
total read jobs: 1
active read jobs: 0
queued read jobs: 0

$ slk_helpers job_queue --format d
no queue, waiting time in the queue: none
```

- Does the slk recall need to run the whole time? No!

Get the job id from the slk log file

- If you run slk and check the process id:

```
$ ps -ef | grep slk
k204221  1332938 1034014  0 10:48 pts/36   00:00:00 /bin/bash /sw/.../slk recall ...
k204221  1332963 1332938 93 10:48 pts/36   00:00:05 java -Xmx4g -jar /sw/.../slk-cli-tools-
3.3.91.jar recall ...
```

  - 1332938 is the process id of slk which you would get when you do `$$` or send it to the background
  - 1332963 is the pid of the JVM which is a child pid to 1332938 and which is used in the slk log
  - => To get the pid, which is used in the slk log, you need to get the child pid of the JVM while slk is running
- When you have the pid then you can grep for the job id

```
1332963 INFO Created copy job with id: '[0-9]*'
```

Questions?

Please get up!

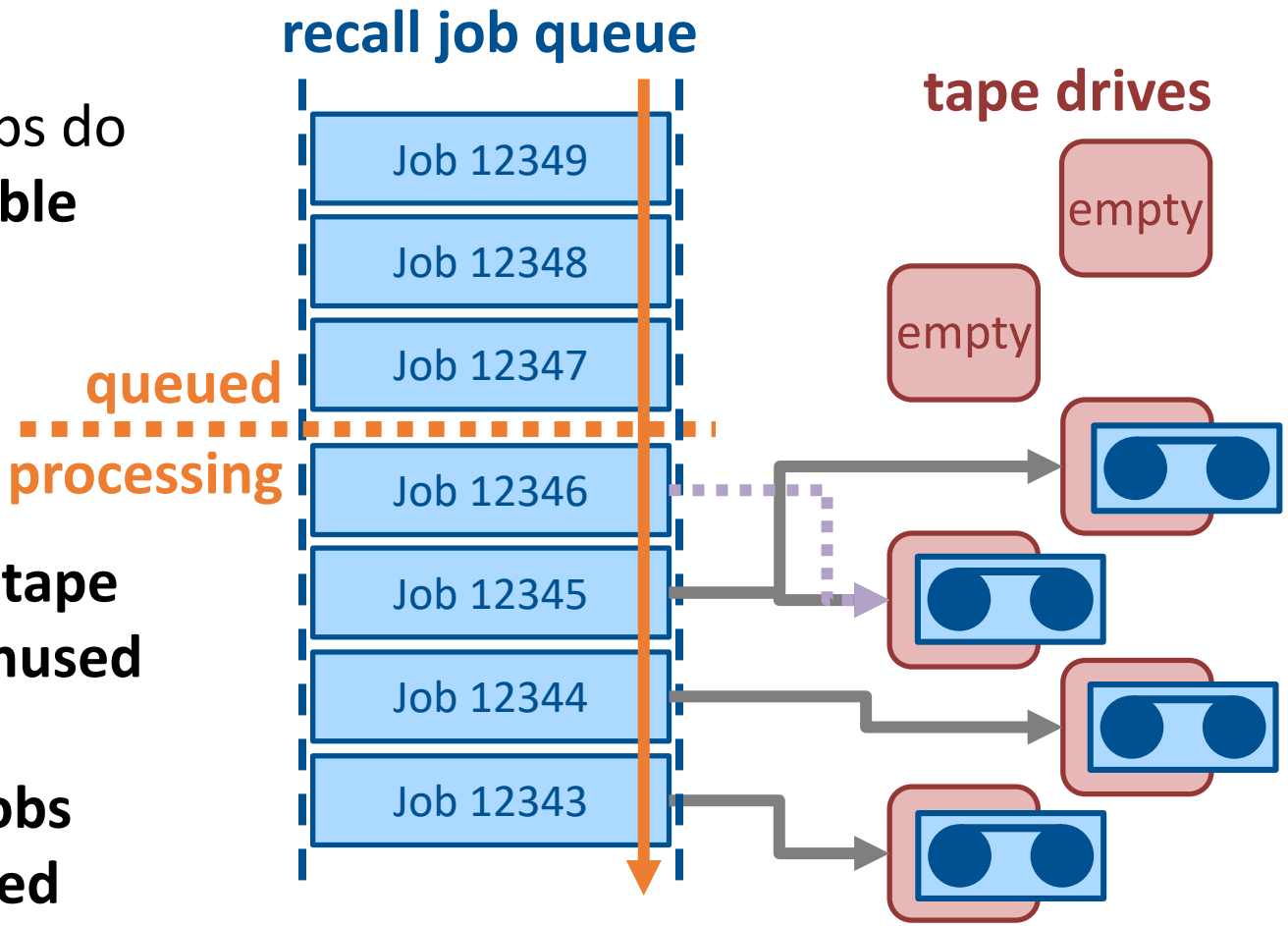
What does limit the number of recalls?

- number of tape drives is limited
- additional limiting boundary conditions
  - three different generations of tape and tape drives
    => tape drives of type A read/write only tapes of type A
  - tapes are located in different libraries and can only be read by tape drives in their library(-complex)
- StrongLink has static job limit

static job limit: tape drives not used

When running jobs do **not use all available** tape drives …

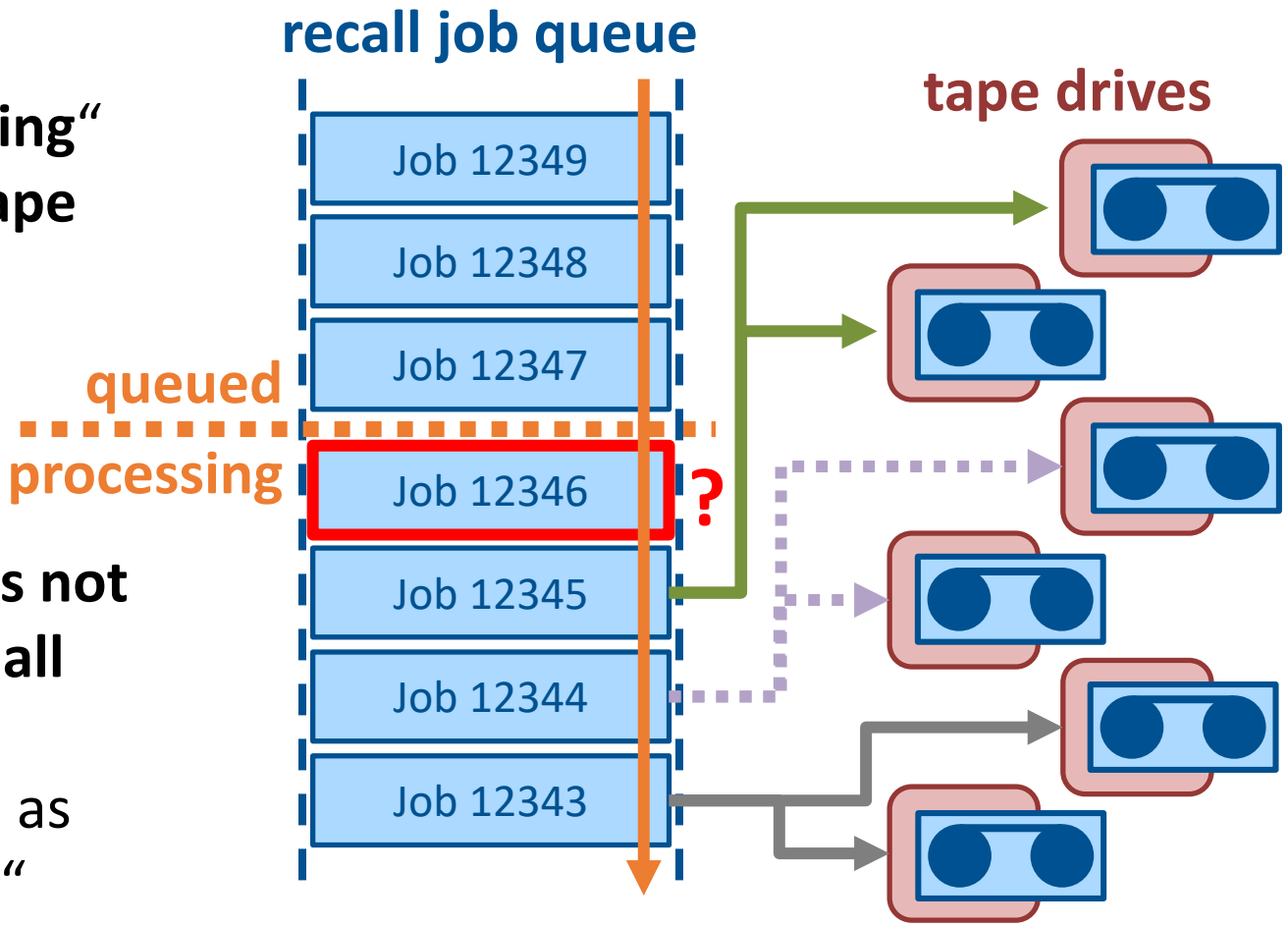
… then **unused tape drives remain unused** and the **queued jobs remain queued**



static job limit: jobs not running because tape drive missing

When a „**processing**“ job has **no free tape drive** available …

… then **the job is not processed at all**
but
it is **displayed** as „**processing**“

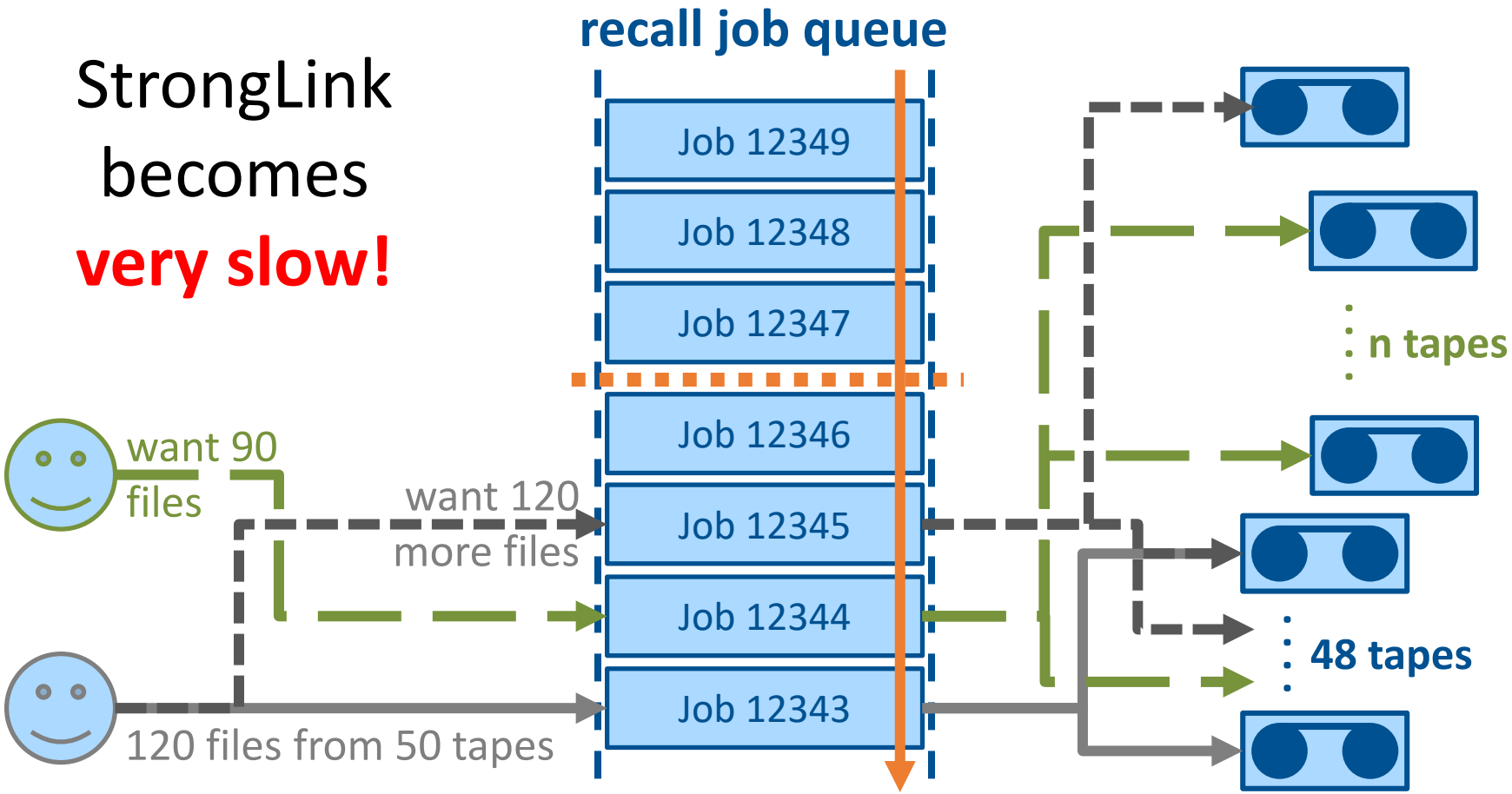


Questions?

Please get up!

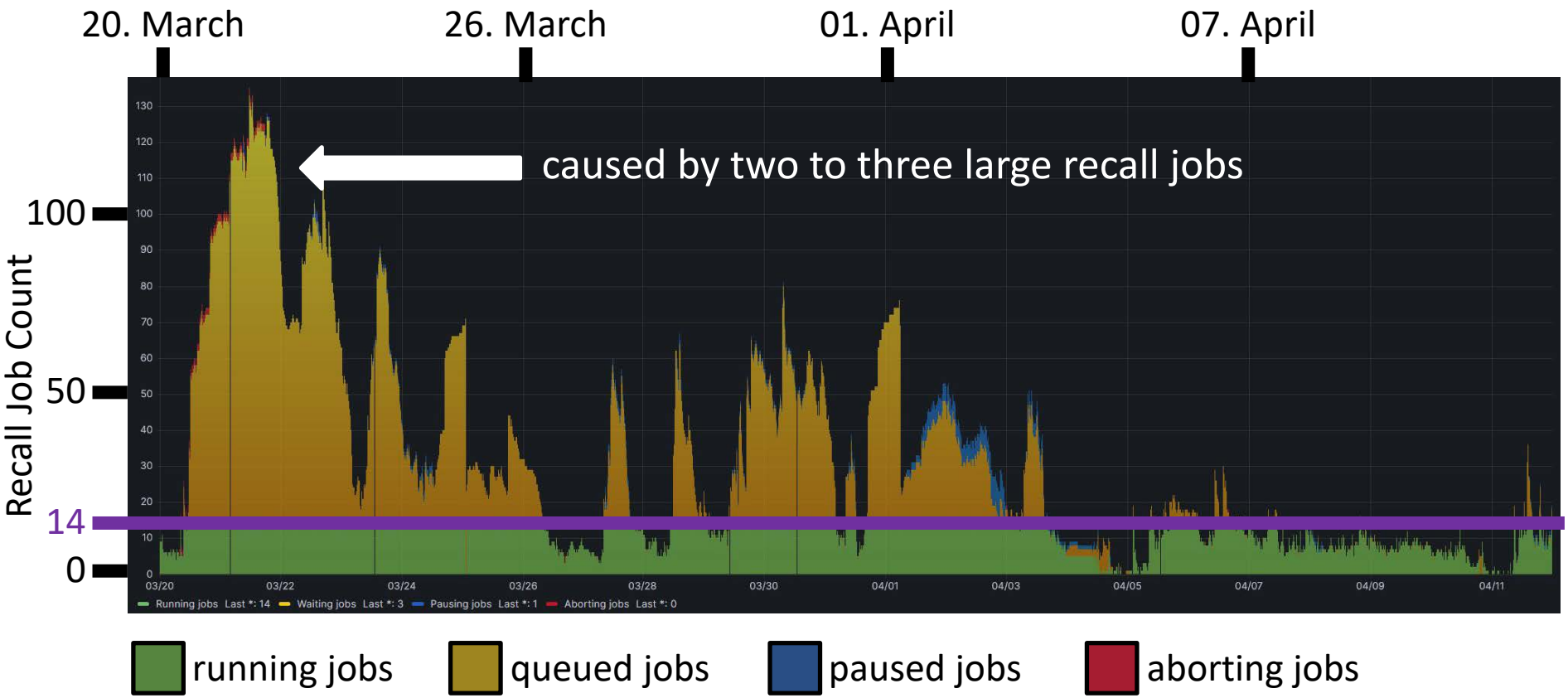
problem

StrongLink becomes **very slow!**



problem: recall jobs with many tapes (I)

Recall Queue Length in StrongLink



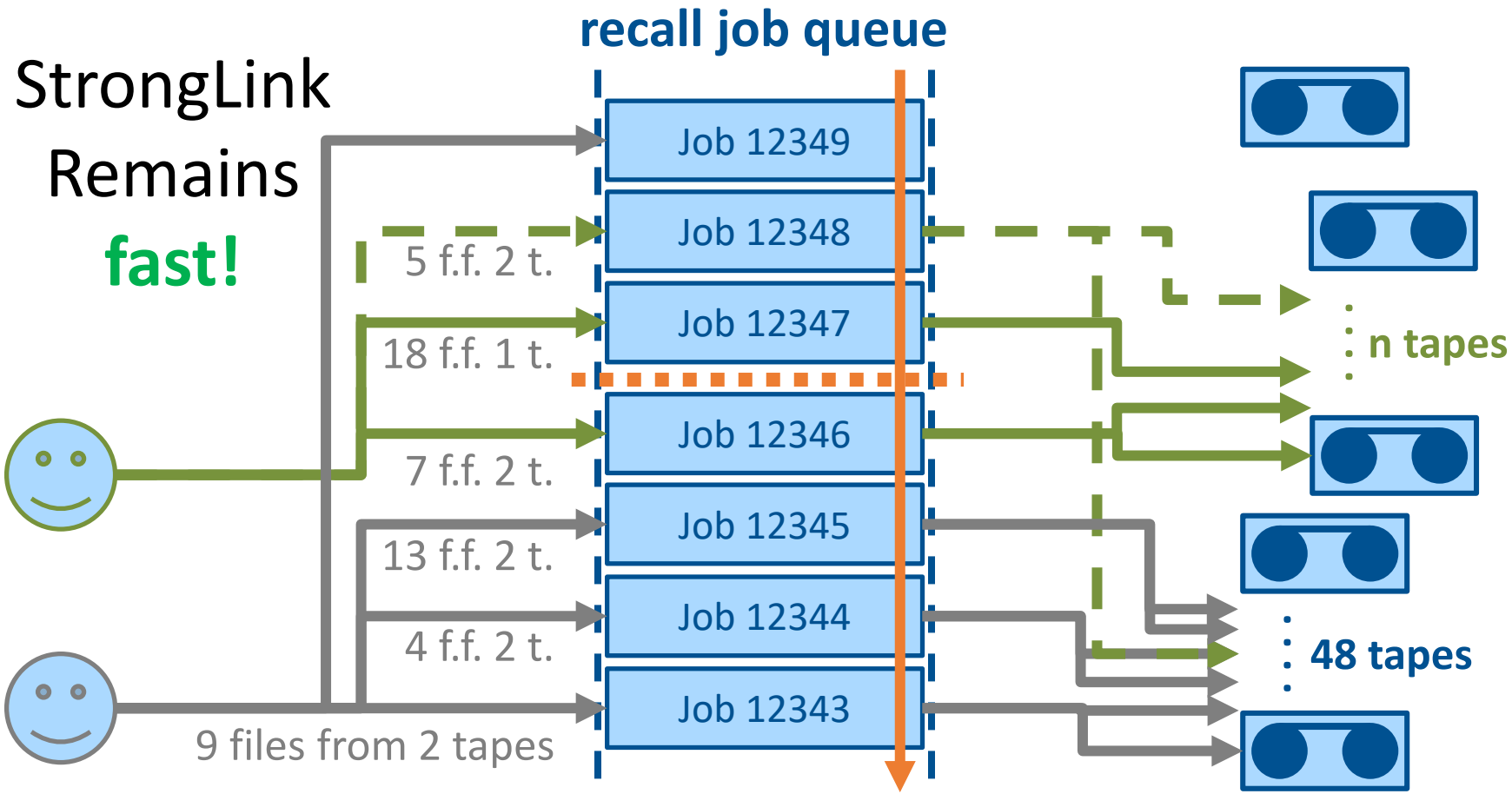
problem: recall jobs with many tapes (II)

Solutions

a) Run a recall for each file
   - Will not cause a slow system
   - Will cause many recall jobs
   - We will loose synergetical effects when multiple files need to be read from one tape

b) Run a recall for each tape
   - Will not cause a slow system
   - Will cause a lot but not too many recall jobs
   - We will get data from each tape most effectively

split recalls with many tapes (I)



What now?

Imagine

- You need to get 360 files
- These files are on 71 tapes
- You plan to run 71 retrievals
- Retrieval might run long and/or fail
- Tapes might be blocked for writing
- You need to check each recall job …

Good luck!

Questions?

Please get up!

Practice session IIIa: split recalls with many tapes

- This will help

```
$ slk_helpers group_files_by_tape ...
$ slk_helpers gfbt ...
```

- Count number of tapes:

```
$ slk_helpers gfbt --count-tapes -R \
                        /arch/bm0146/k204221/iow
```

- How are the files distributed onto tapes:

```
$ slk_helpers gfbt --details -R \
                        /arch/bm0146/k204221/iow
```

Practice session IIIa: split recalls with many tapes – output (I)

- Count number of tapes:

```
$ slk_helpers gfbt --count-tapes -R /arch/bm0146/k204221/iow
8 tapes with single-tape files
0 tapes with multi-tape files
```

- How are the files distributed onto tapes:

```
$ slk_helpers gfbt --details -R /arch/bm0146/k204221/iow
 cached (AVAILABLE  ): /arch/bm0146/k204221/iow/iow_data5_006.tar ...
  C25543L6 (AVAILABLE  ): /arch/bm0146/k204221/iow/iow_data_006.tar
  C25566L6 (AVAILABLE  ): /arch/bm0146/k204221/iow/iow_data5_002.tar
  M12208M8 (AVAILABLE  ): /arch/bm0146/k204221/iow/iow_data_001.tar ...
  M12211M8 (AVAILABLE  ): /arch/bm0146/k204221/iow/iow_data_002.tar ...
  C25570L6 (AVAILABLE  ): /arch/bm0146/k204221/iow/iow_data5_003.tar
  M12215M8 (AVAILABLE  ): /arch/bm0146/k204221/iow/iow_data_004.tar
  C25539L6 (AVAILABLE  ): /arch/bm0146/k204221/iow/iow_data_003.tar ...
  M12217M8 (AVAILABLE  ): /arch/bm0146/k204221/iow/iow_data2_006.tar
```

Practice session IIIb: split recalls with many tapes

- run searches automatically (print progress: -vv)

```
$ slk_helpers gfbt --full -R /arch/bm0146/k204221/iow
$ slk_helpers gfbt --full -R -vv
                              /arch/bm0146/k204221/iow
```

- also works with search id

```
$ slk_helpers gfbt --full --search-id 12345
```

Practice session IIIc: split recalls with many tapes

```
$ slk_helpers gfbt --full -R /arch/bm0146/k204221/iow
    cached (AVAILABLE  ): 469675
  C25543L6 (AVAILABLE  ): 469676
  C25566L6 (AVAILABLE  ): 469677
  M12208M8 (AVAILABLE  ): 469678
  M12211M8 (AVAILABLE  ): 469679
  C25570L6 (AVAILABLE  ): 469680
  M12215M8 (AVAILABLE  ): 469681
  C25539L6 (AVAILABLE  ): 469682
  M12210M8 (AVAILABLE  ): 469683
```

tape bacode

tape status

search id

tape stati

https://docs.dkrz.de/doc/datastorage/hsm/slk_helpers.html#tape-stati

Questions?

Please get up!

What now?

- slk retrieve unfavorable because
  - of long waiting time
  - reasons of errors harder identify
- better:
  - run recall => get the job id => wait until job is finished => run retrieval
- slk wrapper script for this purpose
  - automatically runs SLURM jobs
  - one log file for all process steps (recall, wait, retrieve)

Practice session IVa: slk wrapper scripts

- wrapper overview

```
$ slk_wrapper_recall_wait_retrieve \
      <account> \
      <src> \
      <dst> \
      <log file suffix>
```

- wrapper with example values

```
$ slk_wrapper_recall_wait_retrieve \
      bm0146 \
      466080 \
      /scratch/k/k204221/tmp/test_test_wrappers/data4 \
      466080
```

Practice session IVb: slk wrapper scripts

- Run wrapper

```
$ slk_wrapper_recall_wait_retrieve ... ... ... 466079
```

- Look into log!

```
$ less rwr_log_466079.log
```

- PLEASE: do not run 50 at once

  => other users are blocked than => please delay

```
$ sbatch -A ... -p shared --begin="now+1hour" \
                slk_wrapper_recall_wait_retrieve ...
```

Technical summary

Please do this when I retrieve more than 10 files at once

- Check number of tapes
- If **less than five tapes or equal**
  => go on with one recall / retrieval
- If more than five tapes
  => split
  - run `slkh gfbt --full` (search two tapes each: `--smtnps 2`)
  - run slk wrapper (maybe with time delay; each 5 at once)

What else

- Please send us feedback to support@dkrz.de
  - on the wrapper script
  - on slk_helpers
  - on the „active“ breaks in this talk
  - Do you wish TechTalks on other HSM-related topics?

- Please note:
  - pyslk: preparing new major release for after holiday season

Questions?

Get up!

Search? Next Time?

quick look into on searches

slk_helpers v1.9.5 needed!

- Run a search query

```
$ slk search '<JSON SEARCH QUERY>'
```

- Two commands to generate search queries
  - Generate query from file list

```
$ slk_helpers gen_file_query <FILES>
```

  - Generate query from conditions (key-value-pairs)

```
$ slk_helpers gen_search_query <FIELD>=<CONDITION>
```

examples: gen_file_query (I)

```
# search for the file iow_data_001.tar
$ slk_helpers gen_file_query iow_data_001.tar
{"resources.name":{"$regex":"iow_data_001.tar"}}

# search for these files iow_data_00[0-9].tar
# (regex, no wildcard / bash glob)
$ slk_helpers gen_file_query iow_data_00[0-9].tar
{"resources.name":{"$regex":"iow_data_00[0-9].tar"}}
```

Note:

When searching for files, always regular expressions are assumed. Thus, the first query will not only find the file '`iow_data_001.tar`' but also files with names like '`old_iow_data_001.tar`' and '`iow_data_001.tar.gz`'.

To avoid this, you either have to search for '`^iow_data_001.tar$`' or change the resulting quert string to '`{"resources.name":"iow_data_001.tar"}`'

examples: gen_file_query (II)

```
# search for all files in /arch/bm0146/k204221/iow
$ slk_helpers gen_file_query /arch/bm0146/k204221/iow -R
{"path":{"$gte":"/arch/bm0146/k204221/iow"}}

# search for these files iow_data_00[0-9].tar in the folder
# /arch/bm0146/k204221/iow
$ slk_helpers gen_file_query \
                    /arch/bm0146/k204221/iow/iow_data_00[0-9].tar
{"$and":[
      {"path":{"$gte":"/arch/bm0146/k204221/iow","$max_depth":1}},
      {"resources.name":{"$regex":"iow_data_00[0-9].tar"}}
]}
```

Note:

When you remove '`,"$max_depth":1`' then the search query becomes recursive.

examples: gen_file_query (III)

```
# search recursively for these files iow_data_00[0-9].tar
# in /arch/bm0146/k204221 (not in /arch/bm0146/k204221/iow)
$ slk_helpers gen_file_query -R \
      /arch/bm0146/k204221/iow_data_00[0-9].tar
{"$and":[
      {"path":{"$gte":"/arch/bm0146/k204221"}},
      {"resources.name":{"$regex":"iow_data_00[0-9].tar"}}
]}
```

examples: gen_file_query (IV)

```
# search recursively for all files which end on .nc
# in /arch/bm0146/k204221 (not in /arch/bm0146/k204221/iow)
$ slk_helpers gen_file_query -R /arch/bm0146/k204221/[.]nc$
{"$and":[
        {"path":{"$gte":"/arch/bm0146/k204221"}},
        {"resources.name":{"$regex":"[.]nc$"}}
]}
```

Note:

In StrongLink search queries it is not possible to escape special characters like '`.`' in regular expressions. Instead, you have to put them in square brackets '`[.]`'.

The '$' in the end indicates a line end. If not used, files like 'test`.nc.zip`' were found.

examples: gen_search_query (I)

slk_helpers v1.9.5 needed!

```
# generate search query for files in /arch/.../iow with
# size below 2 MB
$ slk_helpers gen_search_query 'resources.size<2097152' \
        path=/arch/bm0146/k204221/iow
{"$and":[
        {"resources.size":{"$lt":2097152}},
        {"path":{"$gte":"/arch/bm0146/k204221/iow","$max_iterations":1}}
]}

# generate search query for files in /arch/.../k204221
# with size above 5 GB and below 10 GB
$ slk_helpers gen_search_query 'resources.size>5368709120' \
        'resources.size<10737418240' path=/arch/bm0146/k204221/iow
{"$and":[
        {"resources.size":{"$lt":10737418240}},
        {"path":{"$gte":"/arch/bm0146/k204221/iow","$max_iterations":1}},
        {"resources.size":{"$gt":5368709120}}
]}
```

examples: gen_search_query (II)

slk_helpers v1.9.5 needed!

```
# generate search query for files in /arch/.../iow which are in the cache
$ slk_helpers gen_search_query path=/arch/bm0146/k204221/iow \
        smart_pool=slpstor
{"$and":[
        {"path":{"$gte":"/arch/bm0146/k204221/iow","$max_iterations":1}},
        {"smart_pool":"slpstor"}
]}
```

Note:

Although a `not` operator exists with which it is possible to search for not-cached files, we strongly recommand not to try this because this search runs very inefficient and takes very long.

examples: gen_file_query and gen_search_query (I)

slk_helpers v1.9.5 needed!

```
# generate search query for files in /arch/.../iow with
# size below 5 GB and the file extension .tar
$ slk_helpers gen_file_query /arch/bm0146/k204221/iow/[.]tar -R
{"$and":[
        {"path":{"$gte":"/arch/bm0146/k204221/iow"}},
        {"resources.name":{"$regex":"[.]tar"}}
]}
$ slk_helpers gen_search_query 'resources.size<5368709120' --search-query
'{"$and":[{"path":{"$gte":"/arch/bm0146/k204221/iow"}},{"resources.name":{
"$regex":"[.]tar"}}]}'
{"$and":[
        {"path":{"$gte":"/arch/bm0146/k204221/iow"}},
        {"resources.name":{"$regex":"[.]tar"}},
        {"resources.size":{"$lt":5368709120}}
]}
```

Note: `gen_search_query` cannot create queries with `regex` yet. But, you can create such a query with `gen_file_query` and extend the latter query with `gen_search_query` via the option `--search-query <existing search query>`.

examples: gen_file_query and gen_search_query (I)

slk_helpers v1.9.5 needed!

```
# generate search query for files in /arch/.../iow with
# size below 5 GB and the file extension .tar
$ slk_helpers gen_search_query 'resources.size<5368709120' --search-query
`slk_helpers gen_file_query /arch/bm0146/k204221/iow/[.]tar -R`
{"$and":[
        {"path":{"$gte":"/arch/bm0146/k204221/iow"}},
        {"resources.name":{"$regex":"[.]tar"}},
        {"resources.size":{"$lt":5368709120}}
]}
```

Note: Alternatively to the previous slide you put both commands in one line.

Questions?

support@dkrz.de