DKRZ Tech Talk:
DKRZ Tape archive -- archival basics

18th June 2024, hybrid

Daniel Heydebreck (HSM Support)

Online in the Q&A Document
Hadieh Monajemi (Systems, StrongLink Administration)
Michael Kurtz (Datamanagement, Substitute HSM Support)

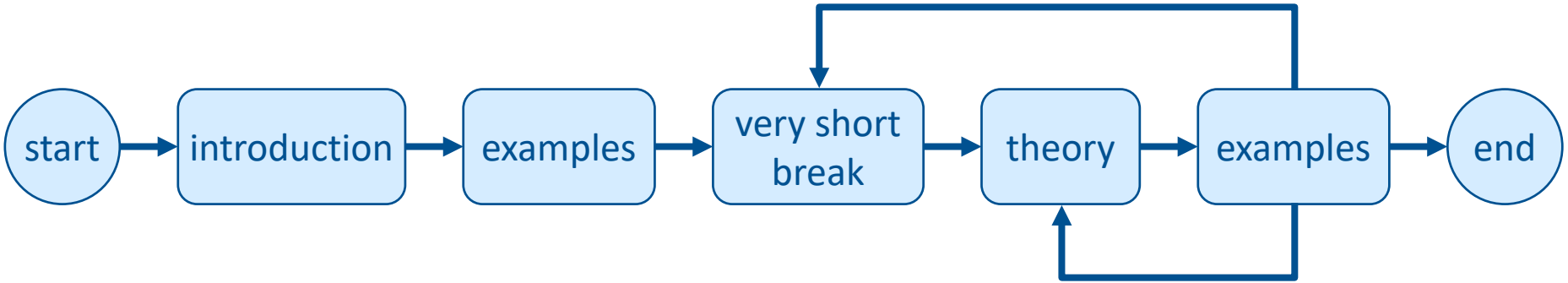
Deutsches Klimarechenzentrum (DKRZ)

Important links

- **Q&A document**:
  https://pad.gwdg.de/szSeKyicRvC_3A-y3IrVPg#
- **Documentation** for Archivals:
  https://docs.dkrz.de/doc/datastorage/hsm/archivals.html
- Questions / Feedback / Wishes:

  support@dkrz.de

Structure


start
introduction
examples
very short break
theory
examples
end

introduction

HSM tools

slk

- command line tool for basic interaction with StrongLink
- version: 3.3.91
- `module load slk`

packems

- Command line tool to simplify packing and achival by MPI-M and DKRZ
- Version 2.0.3 (soon)
- `module load packems`

slk_helpers

- command line tool, extension for slk made by DKRZ
- v1.12.10 (soon v1.13.0)
- `module load slk`

pyslk

- Python library for interaction with StrongLink
- wrappers around `slk` and `slk_helpers`
- v2.0.0 (soon)
- `module load python3/2023.01-gcc-11.2.0`

Memory footprint of slk archive and retrieve

- slk archive/retrieve use up to 4 GB of memory
- please do not run these commands on login nodes
  - `slk archive`
  - `slk retrieve`
  - `packems`
- instead:
  - interactive SLURM session: `salloc ...`
  - submit SLURM batch script: `sbatch ... ./a_script.sh`
- recommendation:
  allocate 6 GB memory (`--mem=6GB`)

examples

example 0: training data and path structure

training data available:

```
# folder with training data
$ cd /work/bm0146/k204221/material_hsm_workshop
$ cd data/many_small_files/a/0
$ ls
file_00.txt  file_03.txt  file_06.txt  file_09.txt
file_01.txt  file_04.txt  file_07.txt  file_10.txt
file_02.txt  file_05.txt  file_08.txt
```

sandbox namespace in tape archive

```
# folder to archive training data to
$ slk list /dkrz_test/techtalk
```

Namespaces for normal “self-service” archival

```
$ slk list /arch/<project>
$ slk list /double/<project>
```

example 1: basic archival command

archive a file

```
$ slk archive file_00.txt /dkrz_test/techtalk/001
Non-recursive Archive completed
```

- no progress printed
- target folder is automatically created

example 2: memory footprint

slk killed due to memory shortage

```
$ slk archive large_file.nc /dkrz_test/techtalk/000
/sw/spack-levante/slk-3.3.91-wuylnb/bin/slk: line 16: 3448766
Killed ...
```

run slk archive interactively via salloc

```
[k204221@levante6 ~]$ salloc -A bm0146 -p interactive --mem=6GB
...
[k204221@l40048 ~]$ slk archive file_00.txt
/dkrz_test/techtalk/001
Non-recursive Archive completed
```

- a compute time account / project
- interactive partition (in contrast to shared or compute)
- 6GB of memory

example 3: verbose mode

archive a file in normal verbose mode

```
$ slk archive file_01.txt /dkrz_test/techtalk/001 -v
[=====================================\] 100%
       complete. Files archived: 1/1, [3B/3B].
Non-recursive Archive completed
```

- progress bar during archival
- progress bar not captured by SLURM logs

Recommendation for scripts!

archive a file in double verbose mode

```
$ slk archive file_02.txt /dkrz_test/techtalk/001 -vv
file_02.txt SUCCESSFUL
Non-recursive Archive completed
```

- one status line per file:
  SUCCESSFUL, FAILED, SKIPPED

example 4: success and failure

archive same file twice

```
$ slk archive file_03.txt /dkrz_test/techtalk/001 -vv
file_03.txt SUCCESSFUL
Non-recursive Archive completed

$ slk archive file_03.txt /dkrz_test/techtalk/001 -vv
file_03.txt SKIPPED
Non-recursive Archive completed
```

let the archival fail

```
$ slk archive file_03.txt /dkrz_test/no_permissions -vv
file_03.txt FAILED
Non-recursive Archive failed
```

theory

skip / overwrite rules

file already exists in destination location

- `slk archive`

  size and **modification** date of source and destination file are **equal**

  - yes => skip
  - no => overwrite

- `slk retrieve`

  - default => overwrite
  - -s is set => skip
  - -d is set => create duplicate file (existing duplicate is overwritten)

examples

example 5: archive file multiple times

archive a file multiple times

```
$ slk archive -vv file_01.txt /dkrz_test/techtalk/002
test_netcdf_a.nc SUCCESSFUL
Non-recursive Archive completed

$ slk archive -vv file_01.txt /dkrz_test/techtalk/002
test_netcdf_a.nc SKIPPED
Non-recursive Archive completed

$ touch file_01.txt
$ slk archive -vv file_01.txt /dkrz_test/techtalk/002
test_netcdf_a.nc SUCCESSFUL
Non-recursive Archive completed
```

- archival of existing file skipped if unchanged
- file archived again of size or timestamp do not match

example 6: slk archive in scripts => exit codes

archive a file

```
$ slk archive file_03.txt /dkrz_test/techtalk/001 -vv
file_03.txt SUCCESSFUL
Non-recursive Archive completed
$ echo $?
0

$ slk archive file_20.txt /dkrz_test/techtalk/001 -vv
Non-recursive Archive failed
$ echo $?
1
```

- $? contains exit code
- exit code == 0 => success or skipped
- exit code > 0 => failed

slk log file

```
$ less ~/.slk/slk-cli.log
```

successful archival:

```
2024-06-12 11:10:53 levante.dkrz.de 197384 INFO  Executing command: "archive ..."
2024-06-12 11:11:00 levante.dkrz.de 197384 INFO  Non-recursive Archive completed

Archive report
==============
Status: success
Total files uploaded: 1/1 files [3B/3B]
```

failed archival:

```
2024-06-12 11:17:01 levante.dkrz.de 205127 INFO  Executing command: "archive ..."
2024-06-12 11:17:02 levante.dkrz.de 205127 ERROR Unexpected exception
java.nio.file.NoSuchFileException: file_20.txt
[...]

Archive report
==============
Status: incomplete
Total files uploaded: 0/0 files [0B/0B]
```

example 7: list archived content

slk list

```
$ slk list /dkrz_test/techtalk/001
-rw-r--r--- k204221   bm0146    3   13 Oct 2022 9:52 file_00.txt
-rw-r--r--- k204221   bm0146    3   13 Oct 2022 9:52 file_01.txt
-rw-r--r--- k204221   bm0146    3   13 Oct 2022 9:52 file_02.txt
Files: 3
```

- like `ls`
- file size
- modification date and time (mtime)
- user id instead of username if user has never logged in before to StrongLink

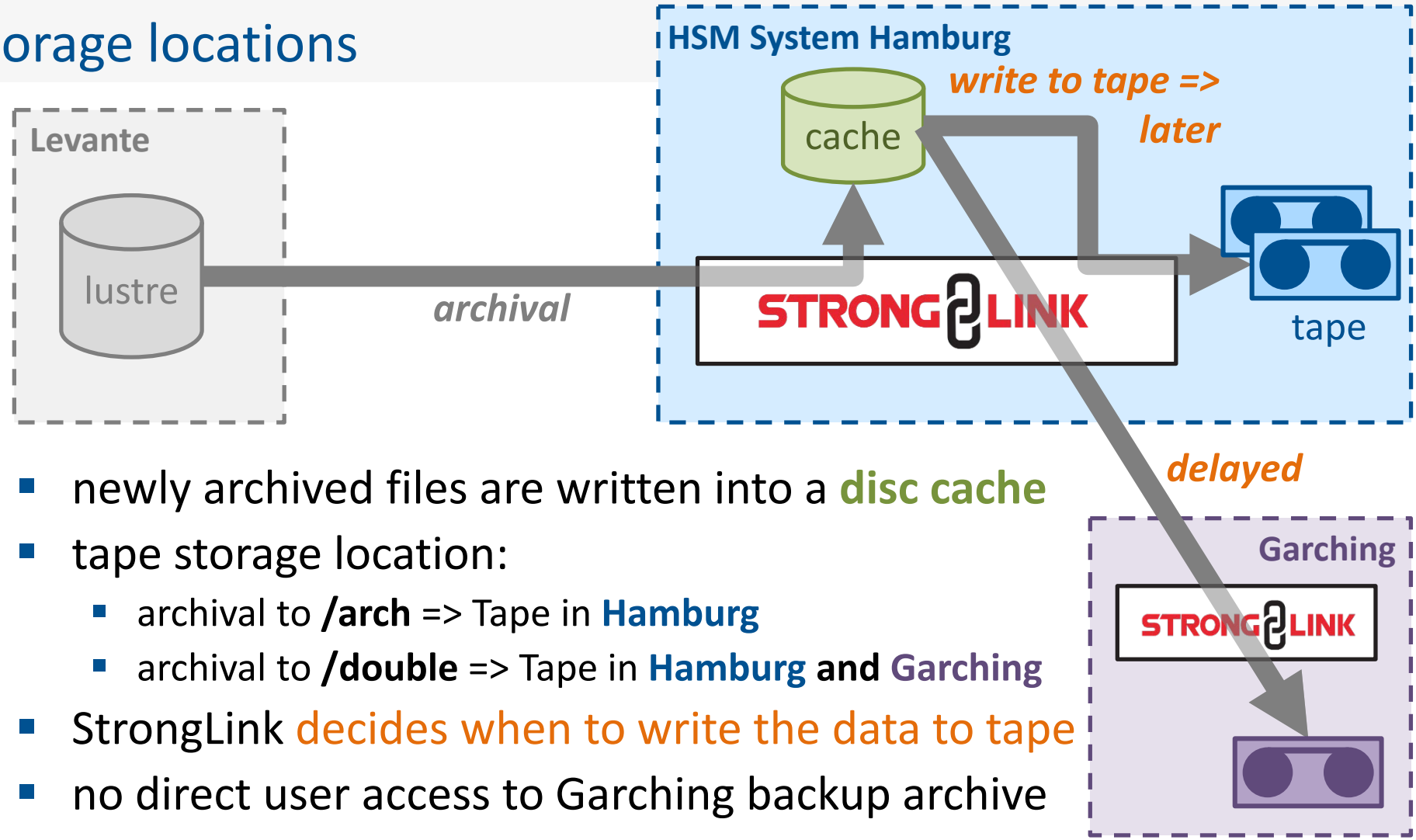
example 7b: slk_helpers list_clone_file

```
$ slk_helpers list_clone_file /arch/bm0146/k204221/iow/INDEX.txt
-rwxr-xr-x   25301    1076          1268945
        2020-06-10T08:25
        2021-10-17T12:33
        2021-10-17T15:52
        2024-05-28T17:28
        56325 /arch/bm0146/k204221/iow/INDEX.txt
Resources: 1
```

```
$ slk_helpers list_clone_file /arch/bm0146/k204221/iow/INDEX.txt
<permissions> <uid>    <gid>         <size>
        <mtime of the file>
        <timestamp: first file version in StrongLink>
        <timestamp: archival of the current file version>
        <timestamp: last internal copy process, e.g. recall>
        <tape id> <path>
Resources: 1
```

theory

Storage locations



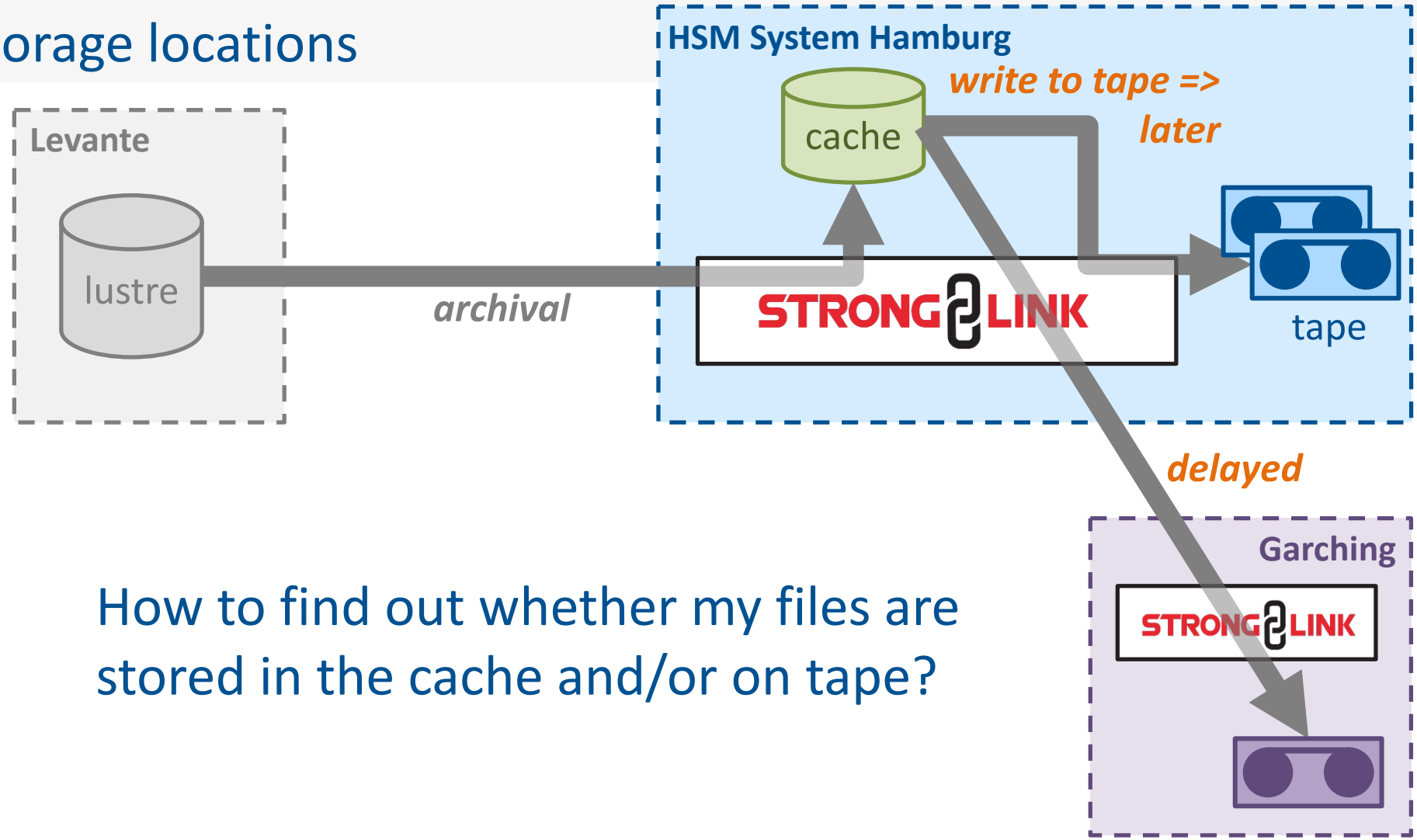
- newly archived files are written into a **disc cache**
- tape storage location:
  - archival to **/arch** => Tape in **Hamburg**
  - archival to **/double** => Tape in **Hamburg and Garching**
- StrongLink decides when to write the data to tape
- no direct user access to Garching backup archive

types of data archival at DKRZ

	Storage period	Who archives the data	access method	root path	physical storage location	more
self-service archival	project run time + 1 year	user	slk	/arch	Hamburg	
	project run time + 1 year	user	slk	/double	Hamburg and Garching	
long term archival (LTA) DOKU	10 years	DKRZ staff	slk (read-only) and WDCC Website	/doku	Hamburg and Garching	public
long term archival (LTA) WDCC	∞	DKRZ statt	WDCC Website and jblob (command line)	-	Hamburg and Garching	DOI + public

WDCC = World Data Centre for Climate (certified, FAIR data)
LTA comparison: https://www.wdc-climate.de/ui/info?site=submitinfo_comparison

Storage locations



How to find out whether my files are stored in the cache and/or on tape?

examples

example 8: storage location via slk list

```
$ 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

example 9: check if cached

```
$ 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_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

example 10: check if on tape

```
$ slk_helpers is_on_tape -R /arch/bm0146/k204221/iow -v
Number of files stored on tape: 23/23

$ slk_helpers is_on_tape -R /dkrz_test/techtalk/001 -v
/dkrz_test/techtalk/001/file_01.txt is not on tape
/dkrz_test/techtalk/001/file_02.txt is not on tape
/dkrz_test/techtalk/001/file_00.txt is not on tape
Number of files stored on tape: 0/3
```

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

theory

exit codes: slk vs. slk_helpers

- `slk`:
  - `0`: successful (includes *file skipped*)
  - `1`: failed
- `slk_helpers`:
  - `0`: successful
  - `1`: special; e.g. specific error or specific file state
    details: https://docs.dkrz.de/doc/datastorage/hsm/slk_helpers.html#exit-codes
  - `2`: failed
  - `3`: timeout error => retry a few minutes later

examples

example 11: exit codes (I)

check if a file exists

```
$ slk_helpers exists /dkrz_test/techtalk/002/file_01.txt
Resource found with following id: 80525280010
$ echo $?
0

$ slk_helpers exists /dkrz_test/techtalk/002/not_exists.txt
Resource not found
$ echo $?
1

$ slk_helpers exists hello world
Only one main parameter allowed but found several: "hello"
and "world"
Please run with '--help exists' for usage information
$ echo $?
2
```

example 12: exit codes (II)

check if a file is cached

```
$ slk_helpers iscached /dkrz_test/cached_file.txt
File is cached
$ echo $?
0

$ slk_helpers iscached /dkrz_test/not_cached_file.txt
File is not cached
$ echo $?
1

$ slk_helpers iscached /dkrz_test/not_exists.txt
Could not handle request. Status: 404. Message: Resource not found
$ echo $?
2
```

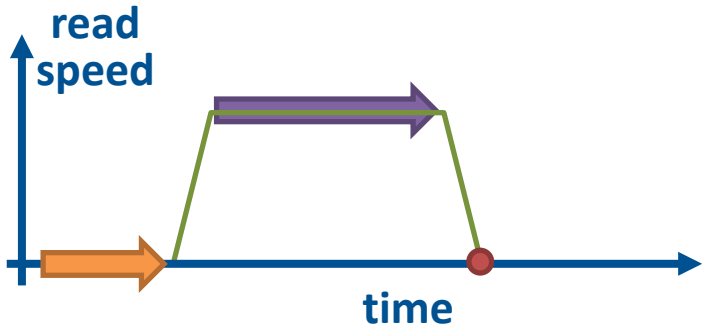
theory

File size rules

- files > 1 TB are possible and have been tested
- recommended file size
  - > 10 GB
  - < 200 GB
  - Why?
    - lower limit: optimize read speed and lifetime of tapes
      => next slides
    - upper limit: read/write rate to/from tape decreases for files larger than ≈250 GB (current HSM system)
- Please do not archive > 3 TB with one `slk archive` call

File size rules: lower limit (I)

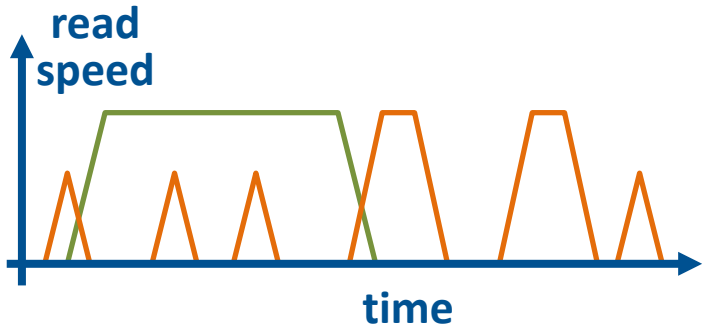
- If a file is read from a tape,
  - the tape drive spools to the location were the file is stored
  - starts reading the file with increasing speed
  - reads at top speed (i.e. 300 MB/s)
  - decreases read speed towards the end of the file
  - stops at the end of the file



- each start / stop of a tape reduces its lifetime

File size rules: lower limit (II)

- 100 GB are stored on one tape
  - as **one 100 GB file**:
    spool, start, long time reading, stop
    - **most time max read speed of 300 MB/s**
    - low impact on tape lifetime
  - as **thousand 100 MB files**:
    repeated: spool, start, short time reading, stop
    - does not reach max read speed
      => **lower read speed + time for spooling**
    - impact on tape lifetime



File size rules: prevent small files

- please **pack small files** into tar balls
  - manually
  - via `packems`
- please check for **hidden files and directories**
  - examples:
    - `.git`
    - `.svn`
    - `.ipynb_checkpoints`
  - ignore hidden: `slk archive -x ...`

examples

example 13: exclude hidden files

data to archive

```
# go to data to be packed and archived
$ cd /work/bm0146/k204221/material_hsm_workshop/data/
$ ls -la folder_with_hidden_files
-rw-r--r--  ...  .config
drwxr-sr-x  ...  .git
-rwxr-xr--  ...  test_netcdf_a.nc
-rwxr-xr--  ...  test_netcdf_b.nc
```

archival

```
# archive recursively without hidden files
$ slk archive -R -x -vv folder_with_hidden_files /dkrz_test/techtalk/005
.../folder_with_hidden_files/test_netcdf_c.nc SUCCESSFUL
.../folder_with_hidden_files/test_netcdf_d.nc SUCCESSFUL
Recursive Archive completed
```

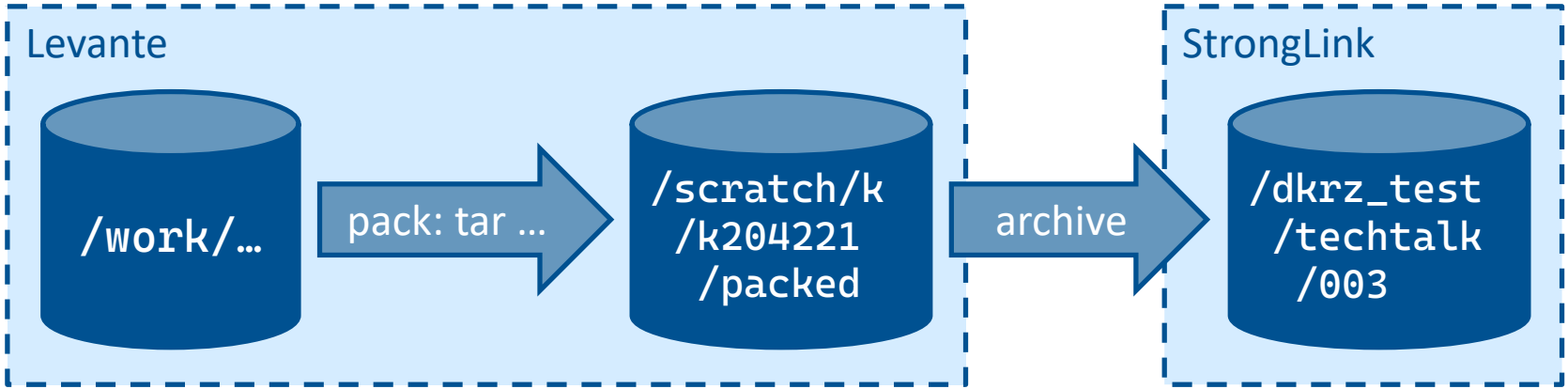
- hidden files are ignored

example 14/15: pack and archive data

data to archive

```
# go to data to be packed and archived
$ cd /work/bm0146/k204221/material_hsm_workshop/data/many_small_files/a/0
$ ls
file_00.txt  file_01.txt  file_02.txt  file_03.txt  file_04.txt
file_05.txt  file_06.txt  file_07.txt  file_08.txt  file_09.txt
```

workflow



example 14: pack and archive data manually

```
# create tmp directory for tar balls and pack
$ tmpDir=/scratch/k/k204221/packed
$ mkdir -p ${tmpDir}
$ tar cfvv ${tmpDir}/data_a_001.tar *.txt > ${tmpDir}/data_a_001.tar.idx

# have a look at the files
$ ls ${tmpDir}
data_a_001.tar   data_a_001.tar.idx

# archive the tar ball
$ slk archive ${tmpDir}/data_a_001.tar /dkrz_test/techtalk/003
Non-recursive Archive completed
```

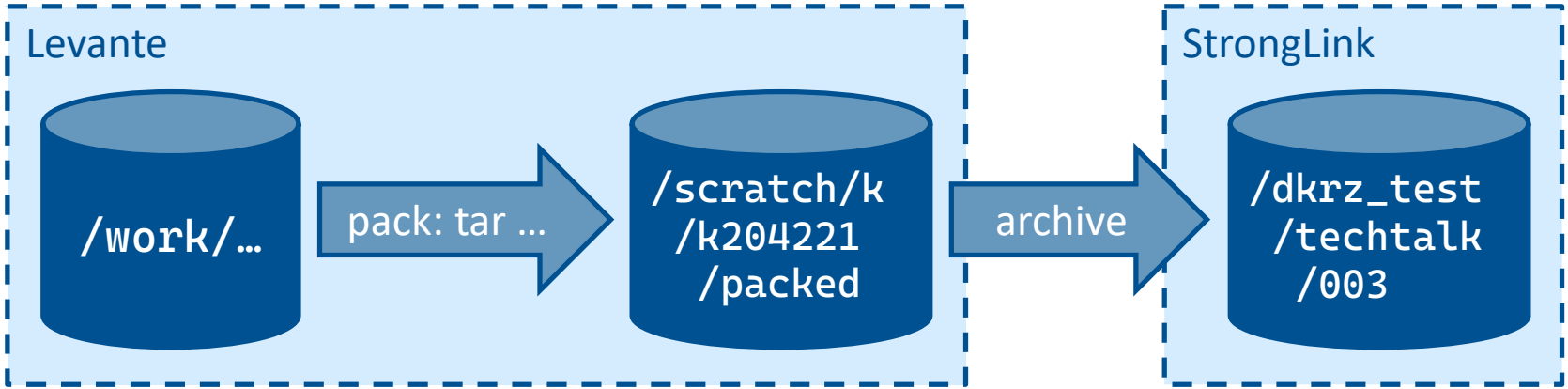
- comply with size recommendations
- *.idx contains list of packed files => keep *somewhere*

example 14/15: pack and archive data

data to archive

```
# go to data to be packed and archived
$ cd /work/bm0146/k204221/material_hsm_workshop/data/many_small_files/a/0
$ ls
file_00.txt  file_01.txt  file_02.txt  file_03.txt  file_04.txt
file_05.txt  file_06.txt  file_07.txt  file_08.txt  file_09.txt
```

workflow



example 15: pack and archive data with packems (I)

```
# pack data with packems
# -d ... => local tmp destination of the tar balls
# -S ... => destination of the tar balls in the tape archive
# -o data_a => prefix of the tar ball names
# --no-archive => only pack and no archival yet
# /work/b.../a/0 => folder to pack recursively
$ packems \
    -d /scratch/k/k204221/packems \
    -S /dkrz_test/techtalk/003 \
    -o data_a \
    --no-archive \
    /work/bm0146/k204221/material_hsm_workshop/data/many_small_files/a/0

# have a look at the files
$ ls /scratch/k/k204221/packems
data_a_001.tar  data_a_001.tar.idx  data_a_001.tar.lst  data_a.mk
```

- automatically pack tar balls of up to **100 GB per ball**

example 15: pack and archive data with packems (II)

```
# archive the tar ball and create index file
# -d ... => local tmp destination of the tar balls
# -S ... => destination of the tar balls in the tape archive
# -o data_a => prefix of the tar ball names
# --archive-only => do only archive data
# /work/b.../a/0 => folder to pack recursively
$ packems \
    -d /scratch/k/k204221/packed \
    -S /dkrz_test/techtalk/003 \
    -o data_a \
    --archive-only \
    /work/bm0146/k204221/material_hsm_workshop/data/many_small_files/a/0
```

- archive tar ball
- more stable to split packing and archival

example 15: pack data with packems

```
$ slk list /dkrz_test/techtalk/003
-rw-r--r--- k204221  bm0146     1.1K   14 Jun 2024 12:11 INDEX.txt
-rw-r--r--- k204221  bm0146     1.0M   14 Jun 2024 12:04 data_a_001.tar
Files: 2
```

What we get in the end:

- Numbered tar balls containing the data
- INDEX.txt
  - list of all packed files
  - mapping: which file is in which tar ball
  - one INDEX.txt per folder <= this will probably change soon!

example 16: real-world packems use case

```
$ slk list /arch/bm0146/k204221/iow
-rwxr-xr-x- k204221     bm0146          1.2M   10 Jun 2020 08:25 INDEX.txt
-rw-r--r--t k204221     bm0146         19.5G   05 Jun 2020 17:36 iow_data2_001.tar
-rw-r--r--t k204221     bm0146         19.0G   05 Jun 2020 17:38 iow_data2_002.tar
-rw-r--r--t 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         19.1G   05 Jun 2020 17:40 iow_data2_005.tar
-rw-r--r--t k204221     bm0146          7.8G   05 Jun 2020 17:41 iow_data2_006.tar
-rw-r--r--t k204221     bm0146        186.9G   05 Jun 2020 19:37 iow_data3_001.tar
-rw-r--r--t k204221     bm0146         24.6G   05 Jun 2020 19:14 iow_data3_002.tar
-rw-r--r--- k204221     bm0146          4.0M   05 Jun 2020 19:43 iow_data4_001.tar
-rw-r--r--t k204221     bm0146         10.5G   05 Jun 2020 19:46 iow_data4_002.tar
-rw-r--r--t 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--t 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
```

theory

problem: incompletely archived files

sources files

```
$ ls -l
... size    ... filename
... 1.1G    ... file_001gb_a.nc
... ...     ... ...
... 1.1G    ... file_001gb_j.nc
```

archived files (archival was killed)

```
$ slk archive *.nc /dkrz_test/techtalk/021
... killed ...
```

Why might slk archive be killed?

- OOM kill => out of memory
  too less memory allocated (`--mem=6GB`)
- SLURM job timeout
  archival took longer than allocated run time
- Connection timeout
  StrongLink overloaded and/or
  too large amount of data (>3 TB) archived at once
- manually killed slk archive

problem: incompletely archived files

archived files (archival was killed)

```
$ slk list /dkrz_test/techtalk/021
...    1.1G  ...  file_001gb_a.nc (Partial File)
...       0  ...  file_001gb_b.nc (Partial File)
...    1.1G  ...  file_001gb_c.nc
...    1.1G  ...  file_001gb_d.nc
...    1.1G  ...  file_001gb_e.nc (Partial File)
...    1.1G  ...  file_001gb_f.nc (Partial File)
...       0  ...  file_001gb_g.nc (Partial File)
...    1.1G  ...  file_001gb_h.nc (Partial File)
... 144.4M  ...  file_001gb_i.nc (Partial File)
...       0  ...  file_001gb_j.nc (Partial File)
Files: 10
```

- green => OK!
- orange => OK?
- red => incomplete => re-archive

problem: incompletely archived files

archived files (archival was killed)

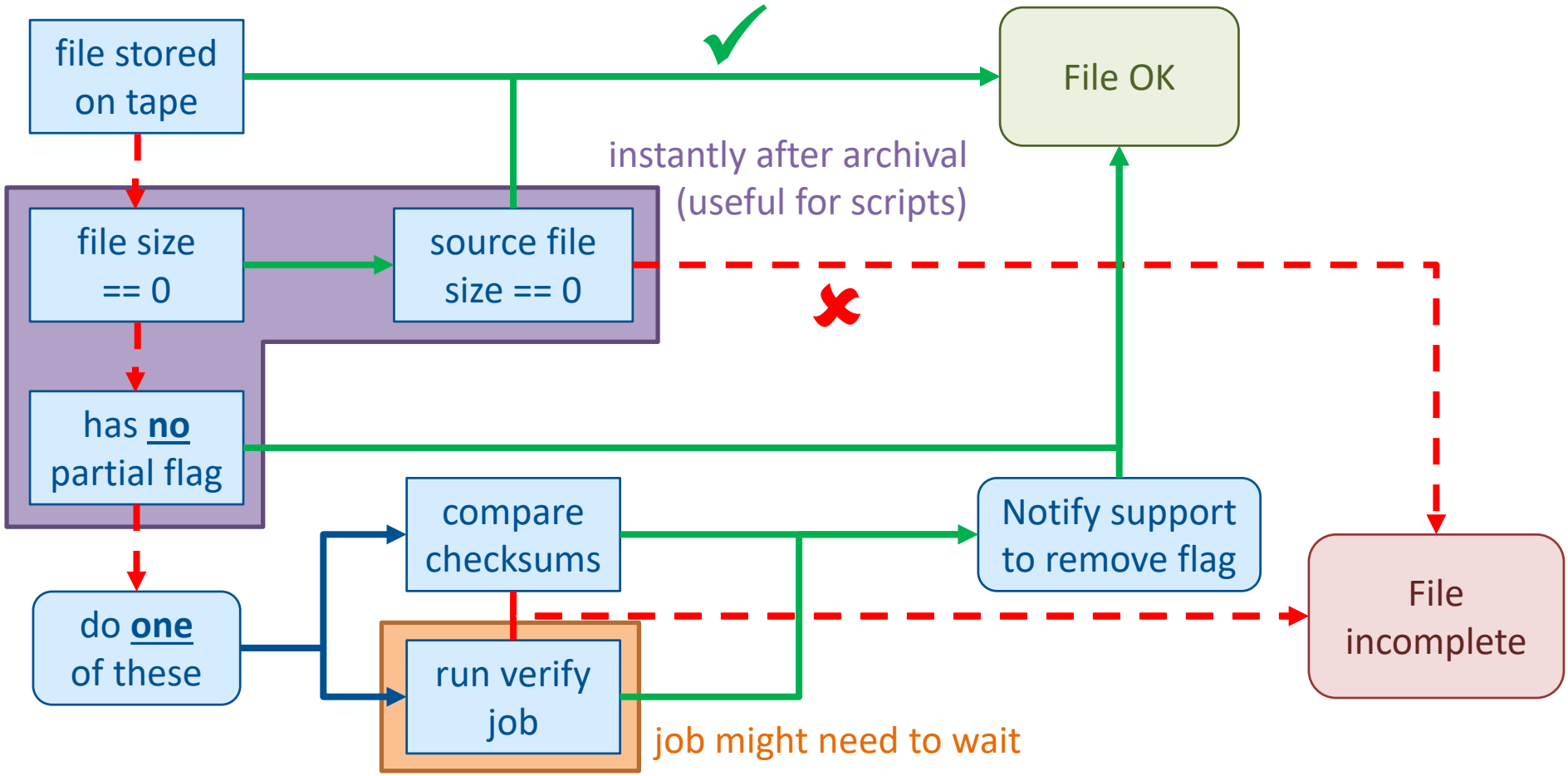
```
$ slk chmod +r /dkrz_test/techtalk/021/file_001gb_i.nc

$ slk list /dkrz_test/techtalk/021
...   1.1G  ...  file_001gb_a.nc (Partial File)
...      0  ...  file_001gb_b.nc (Partial File)
...   1.1G  ...  file_001gb_c.nc
...   1.1G  ...  file_001gb_d.nc
...   1.1G  ...  file_001gb_e.nc (Partial File)
...   1.1G  ...  file_001gb_f.nc (Partial File)
...      0  ...  file_001gb_g.nc (Partial File)
...   1.1G  ...  file_001gb_h.nc (Partial File)
... 144.4M  ...  file_001gb_i.nc
...      0  ...  file_001gb_j.nc (Partial File)
Files: 10
```

- `partial file` of `*_i.nc` disappeared

important to remember

- if `slk list` lists a file, it is not necessarily complete
- the flag `partial file` is not always printed by `slk list`
  => instead: `slk_helpers has_no_flag_partial <file(s)>`
- files flagged as `partial file` cannot be retrieved

verify archived files



verification of archived files

What I do:

- **run `slk archive`**
- if `slk archive` fails, repeat until it succeeds
- **submit verify job** for newly archived files
- check **results of verify job**

Disadvantage:

- verify job might be queued => waiting time
- verify jobs only check the file size

examples

example 17a: run verify job

```
# run verify job
$ slk_helpers submit_verify_job /dkrz_test/techtalk/021 -R
Submitting up to 1 verify job(s) based on results of search id 732325:
search results: pages 1 to 1 of 1; visible search results: 10;
submitted verify job: 247340
Number of submitted verify jobs: 1

# wait for job to be finished
$ slk_helpers job_status 247340
QUEUED (28)
$ slk_helpers job_status 247340
COMPLETED

# collect results
$ slk_helpers result_verify_job 247340
# ... next slide ...
```

example 17b: results verify job

```
# collect results
$ $ slk_helpers result_verify_job 247340
Errors:
Resource content size does not match record: /dkrz_test/techtalk/021/file_001gb_f.nc
Resource content size does not match record: /dkrz_test/techtalk/021/file_001gb_j.nc
Resource content size does not match record: /dkrz_test/techtalk/021/file_001gb_b.nc
Resource content size does not match record: /dkrz_test/techtalk/021/file_001gb_i.nc
Resource content size does not match record: /dkrz_test/techtalk/021/file_001gb_e.nc
Erroneous files: 5
```

- common error when archival was killed:

  `Resource content size does not match record`

  => re-archive!
- all files which are not listed, are Ok

```
$ slk archive *.nc /dkrz_test/techtalk/021
...
```

Example 17c: re-archive data

```
$ slk archive *.nc /dkrz_test/techtalk/021 -vv
file_001gb_a.nc SKIPPED                                  1.1G  *_a.nc (Part...)
file_001gb_b.nc SUCCESSFUL                                  0  *_b.nc (Part...)
file_001gb_c.nc SKIPPED                                  1.1G  *_c.nc
file_001gb_d.nc SKIPPED                                  1.1G  *_d.nc
file_001gb_e.nc SUCCESSFUL                               1.1G  *_e.nc (Part...)
file_001gb_f.nc SKIPPED                                  1.1G  *_f.nc (Part...)
file_001gb_g.nc SUCCESSFUL                                  0  *_g.nc (Part...)
file_001gb_h.nc SKIPPED                                  1.1G  *_h.nc (Part...)
file_001gb_i.nc SUCCESSFUL                             144.4M  *_i.nc (Part...)
file_001gb_j.nc SUCCESSFUL                                  0  *_j.nc (Part...)
Non-recursive Archive completed
```

- just looking into slk list output might be misleading (*_e.nc)
- all files with size mismatch were archived successful
- re-archival complete

Example 17c: re-archive data

```
$ slk archive *.nc /dkrz_test/techtalk/021 -vv
file_001gb_a.nc SKIPPED                                  1.1G  *_a.nc (Part...)
...                                                      ...
file_001gb_h.nc SKIPPED                                  1.1G  *_h.nc (Part...)
...                                                      ...
```

partial file flag not removed when skipped

```
$ slk list /dkrz_test/techtalk/021
... 1.1G  ...  file_001gb_a.nc
... 1.1G  ...  file_001gb_b.nc
... 1.1G  ...  file_001gb_c.nc
... 1.1G  ...  file_001gb_d.nc (Partial File)
... 1.1G  ...  file_001gb_e.nc
... 1.1G  ...  file_001gb_f.nc
... 1.1G  ...  file_001gb_g.nc
... 1.1G  ...  file_001gb_h.nc (Partial File)
... 1.1G  ...  file_001gb_i.nc
... 1.1G  ...  file_001gb_j.nc
```

Example 18: result of another verify job

```
# collect results
$ slk_helpers result_verify_job 247338
Warning: Missing key in the JSON input: attributes.best_store; resource id:
         80527401010; resource path: /dkrz_test/techtalk/020/file_001gb_a.nc
Warning: Resource has an unclear caching state; resource id: 80527401010; resource
         path: /dkrz_test/techtalk/020/file_001gb_a.nc
Errors:
Resource content size does not match record: /dkrz_test/techtalk/020/file_001gb_e.nc
Resource content size does not match record: /dkrz_test/techtalk/020/file_001gb_f.nc
File not found: /dkrz_test/techtalk/020/file_001gb_a.nc
Erroneous files: 3 (some errors have to be solved by the DKRZ support; please
contact support@dkrz.de)
```

- common error when archival was killed:

  `Resource content size does not match record`

  => re-archive!
- file `*_d.nc` has partial file flag but is not listed here => file is OK
- file `*_a.nc` has a rare error => warning + please contact support

summary

- basic usage of slk archive
- capture exit codes
- pack data quickly
- determine storage location of files
- verify archived files