NHR FAU

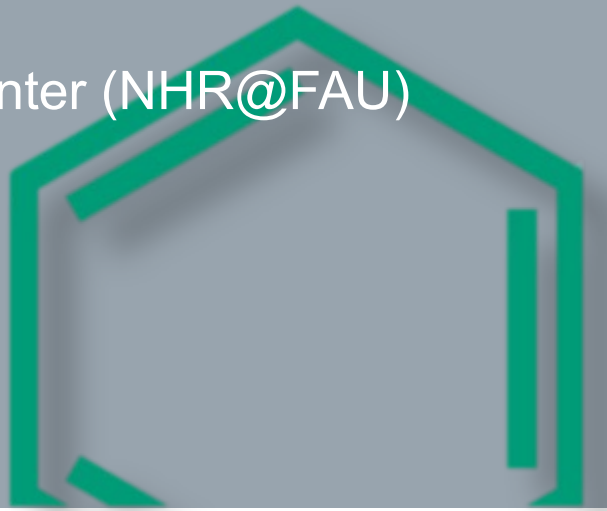
FAU Friedrich-Alexander-Universität Erlangen-Nürnberg

Handling many small files and managing AI data sets

Dr. Anna Kahler

Erlangen National High Performance Computing Center (NHR@FAU)

HPC Café, February 11, 2025



Success Story from the Support

- Background: a user observed severely fluctuating job runtimes leading to performance decreases and job cancellations
- Initial Performance: **28 hours** on NVIDIA V100
- Analysis: 120 GB data in form of 346,133 files
- Underlying reason for bad performance: multi-user system
- Solution: data staging
  - combining the single files into an archive with `tar` (here: 10 GB)
  - extracting the data directly to the node-local storage:

    `tar xf "$SRC/data.tar" -C "$TMPDIR"` (here: 13 minutes)

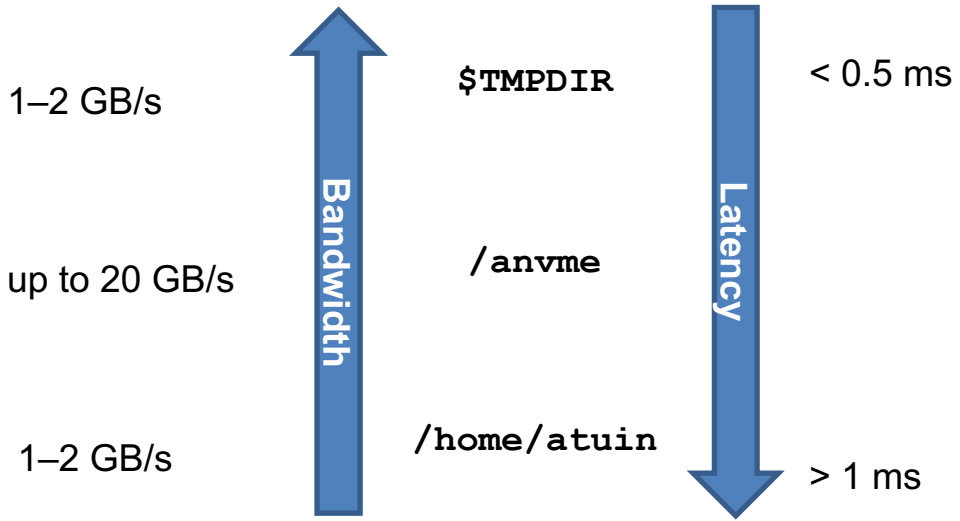
- Final Performance: **12 hours** on NVIDIA V100

Data Sets used for Benchmarking

Data Set	1—medium	2—small	3—big
Number of files	486,995	87,830	139,780
Size of Data Set	452 GB	3.31 GB	1.5 TB
Size of archive	472 GB	1.5 GB	1.45 TB
Number of archive files	21	1	21
Size of gzip archive	198 GB	236 MB	536 GB
Size of zstd archive	220 GB	215 MB	516 GB
Size of lz4 archive	260 GB	393 MB	848 GB

File Systems used for Benchmarking

- **`$TMPDIR`**
  - Node-local directory
  - SSD
- **`/anvme`**
  - High performance IOPS
  - Lustre parallel file system via InfiniBand
- **`/home/atuin`**
  - Short- and mid-term storage
  - General purpose file system
  - Central NFS server



each benchmark run: a single job on Alex with **`#SBATCH --gres=gpu:a40:1`**

Programs/Compression Algorithms used for Benchmarking

Data preparation

- copy archive to **$TMPDIR**:

```
cp $SRC/data.tar.gz $TMPDIR
```

- un-compress data in working directory:

```
gunzip –c $SRC/data.tar.gz > $SRC/data.tar
```

- lz4: compress archive

```
tar --use-compress-program lz4 -cf $SRC/data.tar.lz4 <data to add>
```

- zstd: compress archive

```
tar --use-compress-program zstd -cf $SCR/data.tar.zst <data to add>
```

Programs/Compression Algorithms used for Benchmarking

- copy unpacked data to **$TMPDIR**

  cp -r $SRC/data $TMPDIR or **rsync -avPh $SRC/data $TMPDIR**

- un-pack archive

  cd $TMPDIR; tar xf data.tar

Data usage

- un-pack multiple files in parallel

  cd $TMPDIR; ls -1 $SRC/*.tar | xargs -P 8 -I{} tar xf {}

- un-pack compressed gzip archive

  cd $TMPDIR; tar xzf $SRC/data.tar.gz

- un-pack compressed lz4 archive

  cd $TMPDIR; tar --use-compress-program lz4 -xf $SRC/data.tar.lz4

- un-pack compressed zstd archive

  cd $TMPDIR; tar --use-compress-program unzstd -xf $SRC/data.tar.zst

Copy un-packed data

`cp -r $SRC/data $TMPDIR` or `rsync -avPh $SRC/data $TMPDIR`

Data Set 1: 486,995 files, 452 GB

from **/home/atuin**

- **cp** > 4 h
- **rsync** > 4 h

from **/anvme**

- **cp** 41 m 12 s
- **rsync** 1h 1 m 3 s

Data Set 2: 87,830 files, 3.31 GB

from **/home/atuin**

- **cp** 8 m 48 s
- **rsync** 3 m 54 s

from **/anvme**

- **cp** 1 m 56 s
- **rsync** 1 m 28 s

Copy compressed archive, then un-pack

first copy archive: `cp $SRC/data.tar.gz $TMPDIR`
then un-pack on `$TMPDIR`: `cd $TMPDIR; tar xzf data.tar.gz`

Data Set 1: 486,995 files, 452 GB

from **/home/atuin**

- 1 h 38 m 12 s
  + 59 m 38 s

from **/anvme**

- 2 m 44 s ± 0.2 s
  + 55 m 4 s ± 7 m 38 s

Data Set 2: 87,830 files, 3.31 GB

from **/home/atuin**

- 0.3 s
  + 7 s

from **/anvme**

- 0.2 s
  + 7 s

Copy compressed archive, then un-pack

first copy archive: `cp $SRC/data.tar.gz $TMPDIR`
then un-pack on `$TMPDIR`: `cd $TMPDIR; tar xzf data.tar.gz`

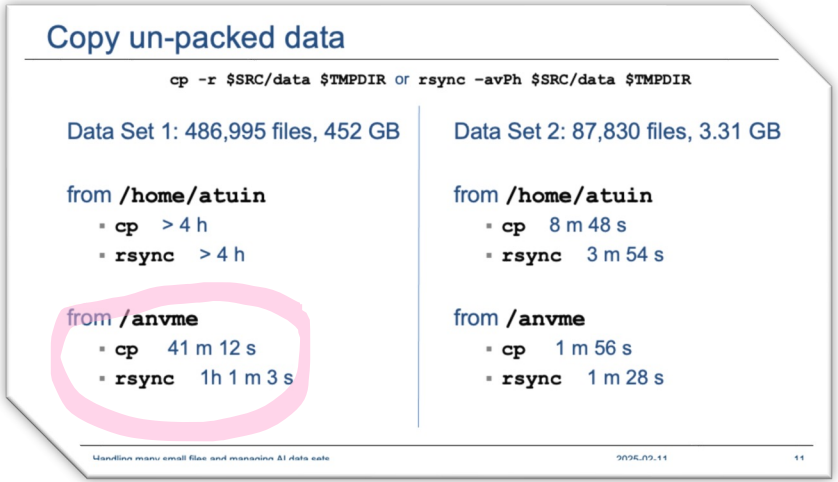
Data Set 1: 486,995 files, 452 GB

from **`/home/atuin`**

- 1 h 38 m 12 s
  + 59 m 38 s

from **`/anvme`**

- 2 m 44 s ± 0.2 s
  + 55 m 4 s ± 7 m 38 s



Un-compress archive, then un-pack

first un-compress archive in working directory: `gunzip -c $SRC/data.tar.gz > $SRC/data.tar`
then un-pack on `$TMPDIR`: `cd $TMPDIR; tar xf $SRC/data.tar`

Data Set 1: 486,995 files, 452 GB

from **`/home/atuin`**

- 1 h 39 m 11 s
  + 24 m 5 s

from **`/anvme`**

- 53 m 47 s ± 33 s
  + 8 m 36 s ± 12 s

Data Set 2: 87,830 files, 3.31 GB

from **`/home/atuin`**

- 8 s
  + 2 s

from **`/anvme`**

- 8 s ± 0.6 s
  + 2 s

Un-compress archive, then un-pack

first un-compress archive in working directory: `gunzip -c $SRC/data.tar.gz > $SRC/data.tar`
then un-pack on `$TMPDIR`: `cd $TMPDIR; tar xf $SRC/data.tar`

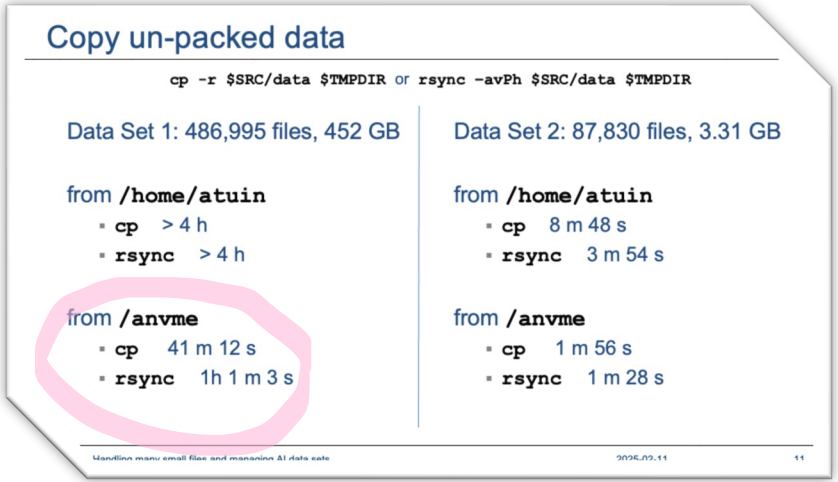
Data Set 1: 486,995 files, 452 GB

from **/home/atuin**

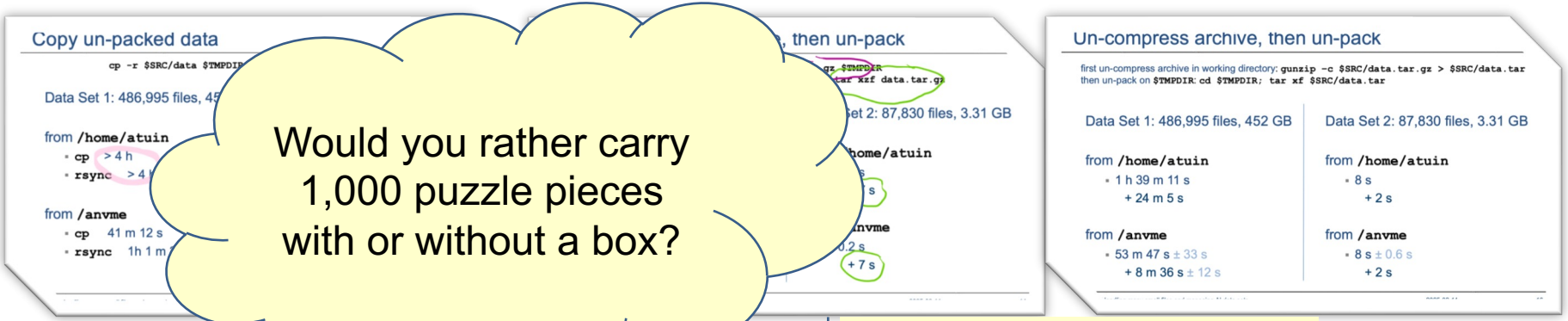
- 1 h 39 m 11 s
  + 24 m 5 s

from **/anvme**

- 53 m 47 s ± 33 s
  + 8 m 36 s ± 12 s



Quick Summary so far…



Would you rather carry 1,000 puzzle pieces with or without a box?

DON'Ts

- Don't copy un-archived data.
- Don't copy and un-pack.
- Don't un-compress and un-pack.

`$TMPDIR` doesn't need a copy of the archive.

`tar` can do both at the same time.

Different compression algorithms

Un-compress archive, then un-pack

first un-compress archive in working directory: `gunzip -c $SRC/data.tar.gz > $SRC/data.tar`
then un-pack on `$TMPDIR`: `cd $TMPDIR; tar xf $SRC/data.tar`

Data Set 1: 486,995 files, 452 GB

from **/home/atuin**

- 1 h 39 m 11 s
  + 24 m 5 s

from **/anvme**

- 53 m 47 s ± 33 s
  + 8 m 36 s ± 12 s

Copy un-packed data

`cp -r $SRC/data $TMPDIR` or `rsync -avPh $SRC/data $TMPDIR`

Data Set 1: 486,995 files, 452 GB	Data Set 2: 87,830 files, 3.31 GB
from **/home/atuin**	from **/home/atuin**
▪ **cp** > 4 h	▪ **cp** 8 m 48 s
▪ **rsync** > 4 h	▪ **rsync** 3 m 54 s
from **/anvme**	from **/anvme**
▪ **cp** 41 m 12 s	▪ **cp** 1 m 56 s
▪ **rsync** 1h 1 m 3 s	▪ **rsync** 1 m 28 s

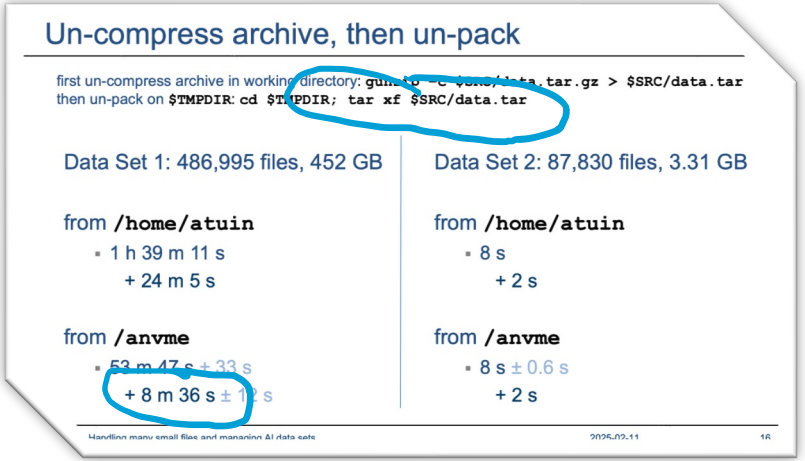
+ 2 s

Different compression algorithms

```
cd $TMPDIR; tar --use-compress-program <lz4|unzstd> -xf $SRC/data.tar.<lz4|zst>
```

lz4

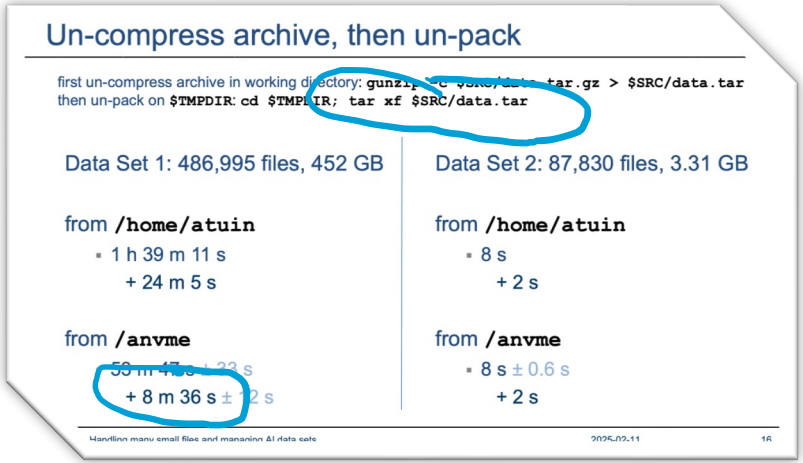
- time for re-packing *.tar on **`/anvme`**
  - Data Set 1: 22 m 52 s
  - Data Set 2: 4s
  - Data Set 3: 17 m 26 s (parallel)
- un-pack compressed archive
  - Data Set 1: 13 m 21 s ± 3 m 21 s
  - Data Set 2: 2 s
  - Data Set 3: 38 m 20 s ± 8 s



Different compression algorithms

```
cd $TMPDIR; tar --use-compress-program <lz4|unzstd> -xf $SRC/data.tar.<lz4|zst>
```

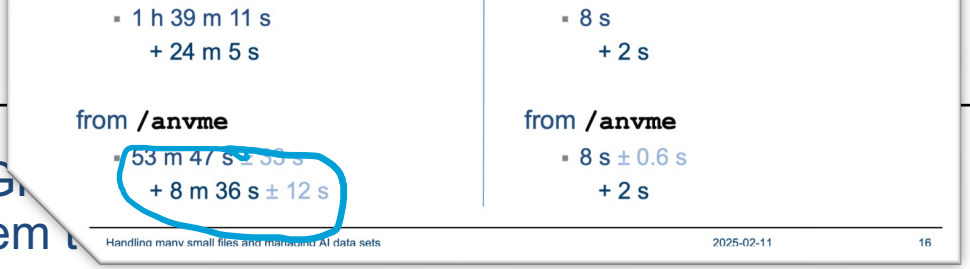
zstd



- time for re-packing *.tar on **`/anvme`**
  - Data Set 1: 43 m 42 s
  - Data Set 2: 6 s
  - Data Set 3: 26 m 56 s (parallel)
- un-pack compressed archive
  - Data Set 1: 14 m 58 s ± 1 m 36 s
  - Data Set 2: 2 s
  - Data Set 3: 53 m 40 s ± 7 m 46 s

Run `tar` in

- Allocating one G
  Why not use them




```
cd $TMPDIR; ls -1 $SRC/*.tar.zst | xargs -P 8 -I{} tar --use-compress-program unzstd -xf {}
```

lz4

- **`xargs -P 4`**
  - Data Set 1: 8 m 35 s ± 4 m 48 s
  - Data Set 3: 19 m 6 s ± 53 s
- **`xargs -P 8`**
  - Data Set 1: 10 m 7 s ± 24 s
  - Data Set 3: 17 m 52 s ± 4 m
- **`xargs -P 16`**
  - Data Set 1: 9 m 49 s ± 14 s
  - Data Set 3: 12 m 39 s ± 1 m 12 s

zstd

- **`xargs -P 4`**
  - Data Set 1: 5 m 53 s ± 9 s
  - Data Set 3: 25 m 54 s ± 1 m 41 s
- **`xargs -P 8`**
  - Data Set 1: 4 m 38 s ± 12 s
  - Data Set 3: 26 m 30 s ± 5 m 4 s
- **`xargs -P 16`**
  - Data Set 1: 4 m 43 s ± 47 s
  - Data Set 3: 14 m 5 s ± 48 s

Handling many small files: Best Practices

- It is possible that the file system is over-loaded. Be patient and still follow our suggestions.
- Pack your data in an archive!
- Use either lz4 or zstd as compression algorithm.


```
tar --use-compress-program=lz4 -cf $SRC/data.tar.lz4 <data to add>
tar --use-compress-program zstd -cf $SRC/data.tar.zst <data to add>
```

- Run **`tar`** in parallel.

```
cd $TMPDIR; ls -1 $SRC/*.tar.zst | xargs -P 8 -I{} tar --use-compress-program unzstd -xf {}
```

Data Pool concept by DKRZ/GWDG

- Training data sets for machine learning applications
- Open data sets of any HPC users
- Project data for other projects to use
- Semi-public project data that should only be shared upon application

https://docs.hpc.gwdg.de/services/datapool/index.html

https://docs.dkrz.de/doc/dataservices/finding_and_accessing_data/pool-data/index.html

- Requirements: name, version, data files, metadata files (public), and README.md (public)
- (Project) PI can request access including a rough estimate on disk space and files/directories
- Successful review process by PIs and “Data Pool Approval Team”

- Publishing generates a unique pool ID which is reference-able

- Editing pools or adding new versions is possible but needs to undergo review process

- Periodical review to retain, delete or archive the data

Data Set formats

- Directories with a million files are not allowed because anyone using the pool would harm the performance of the filesystem for everyone
- Example: ImageNet 1k ILSVRC2012 at CSCS

https://user.cscs.ch/computing/data_science/pytorch/#imagenet-1k-ilsvrc2012

- 1,281,166 training + 50,000 validation images repacked as TFRecord files
- Greatly speeds up data loading during training
- Use Nvidia DALI dataloader to read the ImageNet TF records

- Example script available from CSCS

ML Data Sets of Interest?

Name	Size	Train/Val/Test Samples
ImageNet	~ 150 GB	~ 1.2 M / 50 K / 100 K
LAION-5B	~ 240 TB	~ 5B
LAION-400M	~ 10 TB (if resized to 256x256)	~ 400 M
MNIST	~ 11MB	60 K / - / 10 K
CIFAR-10	~ 163 MB	50 K / - / 10 K
CIFAR-100	~ 163 MB	50 K / - / 10 K
CelebA	~ 1.3 GB	~ 202 K
SVHN	~ 600 MB	~ 73 K / - / 26 K
CUB-200-2011	~ 1.3 GB	~ 6 K / - / 6 K
MS COCO	~ 43 GB	118 K / 5 K / 41 K
Cityscapes	~ 11 GB	~ 3 K / 500 / 1.5 K
KITTI	~ 13 GB	~ 7.5 K / - / 7.5 K
ShapeNet	~ 50 GB	varies
EuroSAT	~ 600 MB	27 K / - / 5 K
nuScenes	~ 400 GB	~ 28 K / 6 K / 6 K
Waymo Open Dataset	~ 1.2 TB	1 K / 150 / 750
Objects365	~ 600 GB	~ 2 M
Wikipedia Corpus	~ 16 GB	varies
GLUE	~ 1 GB	varies

Name	Size	Train/Val/Test Samples
SST-2	~ 80 MB	~ 215 K
SQuAD	~ 48 MB	~ 87 K / 10 K / 10 K
MultiNLI	~ 420 MB	~ 392 K / 20 K / 20 K
Visual Question Answering	~ 25 GB	~ 443 K / 214 K / 447 K
IMDb Movie Reviews	~ 80 MB	25 K / - / 25 K
MMLU	~ 50 MB	- / 1.5 K / 14 K
BraTS	~ 17 GB	369
LibriSpeech	~ 60 GB	~ 960 h / 10 h / 10 h
VoxCeleb1	~ 150 MB	~ 149 K
VoxCeleb2	~ 1.5 GB	~ 1.1 M
IEMOCAP	~ 12 GB	~ 7.4 K
AudioSet	~ 2.1 TB	~ 2.1 M
PubMed	~ 90 GB	N/A
Open Graph Benchmark	varies	varies
Reddit	~ 20 GB	~ 3.8 M posts
UCF101	~ 7.2 GB	~ 9.5 K / - / 3.8 K
Kinetics	~ 450 – 710 GB	~ 250 K - 650 K
HMDB51	~ 2 GB	~ 3.5 K / 1.5 K / 1.5 K
NeRF	~ 5 GB	User-defined

Central Availability of Data Sets at NHR@FAU

- Contact person from chair can talk to us. We can store your data sets.
- Either NFS-based on **`/home/janus`** or **`/anvme`** (very expensive).
  - Chair of Machine Learning and Data Analytics
  - Chair of Multimedia Communications and Signal Processing
  - Chair of Computer Science 9: Computer Graphics
  - Chair of Computer Science 5: Pattern Recognition
  - Chair of Computational Imaging
  - Professorship Artificial Intelligence in Medical Imaging
  - AudioLabs

`tar` options

- -X, --exclude-from=FILE
  Exclude files matching patterns listed in FILE.

- -T, --files-from=FILE
  Get names to extract or create from FILE.

  Unless specified otherwise, the FILE must contain a list of names separated by ASCII LF (i.e. one name per line). The names read are handled the same way as command line arguments. They undergo quote removal and word splitting, and any string that starts with a - is handled as tar command line option.

  If this behavior is undesirable, it can be turned off using the --verbatim-files-from option.

  The --null option instructs tar that the names in FILE are separated by ASCII NUL character, instead of LF. It is useful if the list is generated by find(1) -print0 predicate.

- Please check the **`tar`** manual to find the appropriate flags for your case.

Central Availability of Data Sets at NHR@FAU

- Contact person from chair can talk to us. We can store your data sets.
- Either NFS-based on **`/home/janus`** or **`/anvme`** (very expensive).
- Work with us regarding licensed data sets. We can limit access.
- Work with us regarding appropriate data formats. We will not store millions of single files!

`hpc-support@fau.de`

NHR FAU

PRACE

FAU Friedrich-Alexander-Universität Erlangen-Nürnberg

Thank you