Implementing the Singular Value Decomposition in the Helmholtz Analytics Toolkit HeAT

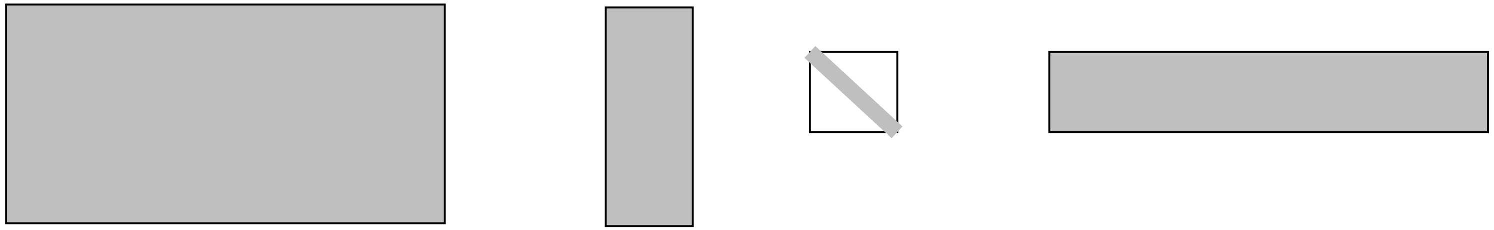
Fabian Hoppe, Philipp Knechtges, Alexander Rüttgers

Institut für Softwaretechnologie – Abteilung High-Performance Computing – Köln

What is the Singular Value Decomposition (SVD)?

Singular Value Decomposition: Given $A \in \mathbb{R}^{m \times n}$ there are uniquely determined orthonormal $U \in \mathbb{R}^{m \times r}$, $V \in \mathbb{R}^{n \times r}$ and $\Sigma = \mathrm{diag}(\sigma_1, \dots, \sigma_r) \in \mathbb{R}^{r \times r}$, $\sigma_1 \geq \cdots \geq \sigma_r > 0$ (“singular values”), such that

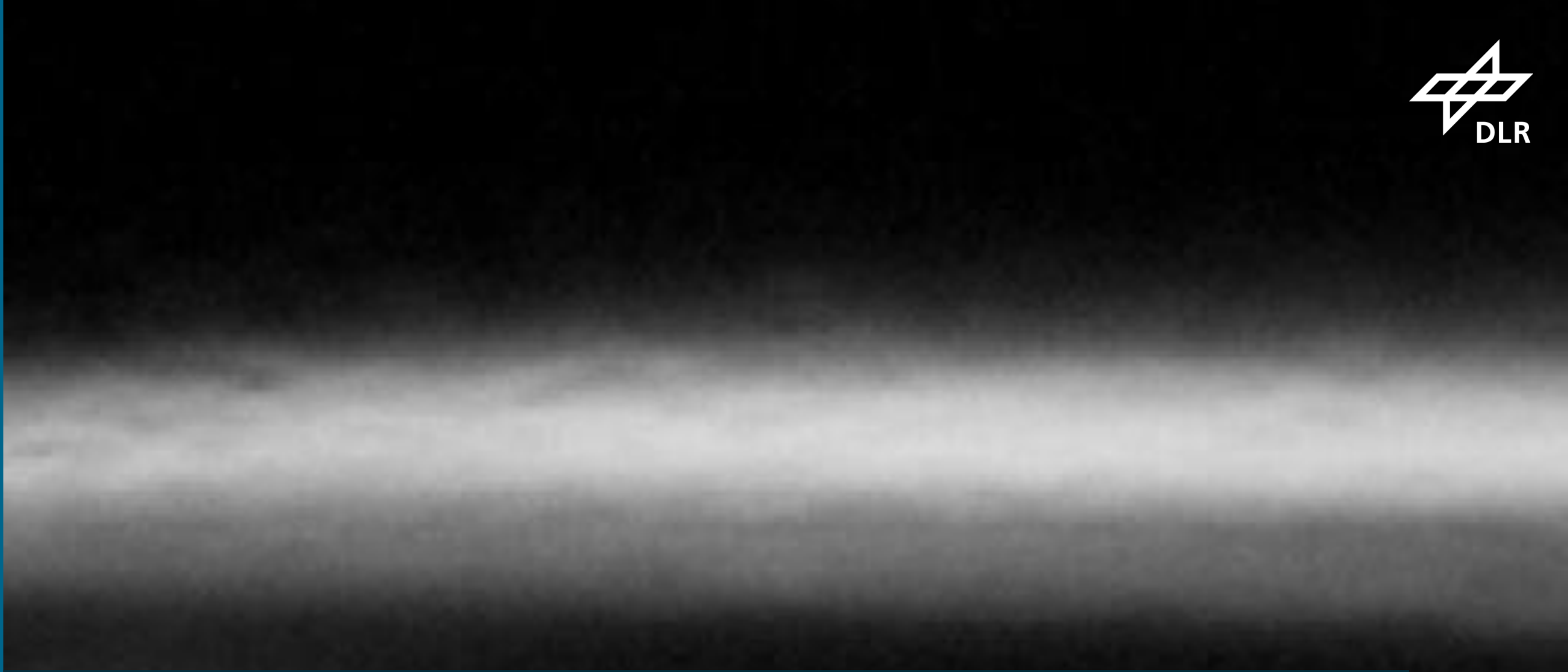
$$A = U \cdot \Sigma \cdot V^T$$



…and why is it important for Machine Learning/Data Science?

SVD (also known as **PCA** – “Principal Component Analysis”) often serves as an **important pre-processing step**

- Determines “*optimal linear coordinate system*” for a given data set
- *Compression/Reduction to the main features* due to low-rank best-approximation property of truncated SVD



The Helmholtz Analytics Toolkit HeAT

HEAT
Helmholtz Analytics Toolkit

HELMHOLTZ
Analytics Framework

- Developed by
  - **KIT** (Markus Götz, Daniel Coquelin, Charlotte Debus)
  - **Jülich Supercomputing Center (JSC)** (Claudia Comito, Kai Krajsek, Michael Tarnawa, Björn Hagemeier)
  - **DLR SC-HPC** (Philipp Knechtges, Martin Siggel, Fabrice von der Lehr, Alexander Rüttgers)
  - and (as usual) plenty of student workers.
- MIT-licensed and available on GitHub: https://github.com/helmholtz-analytics/heat
- Originated from the Helmholtz Analytics Framework (HAF) project.
- Currently available in version 1.2.

➢ **Scope:** High-level library with a numpy-like interface that allows averaged numpy/pytorch-experienced ML users to port their code to HPC systems.

Reference: Götz, Debus, Coquelin, Krajsek, Comito, Knechtges, Hagemeier, Tarnawa, Hanselmann, Siggel, Basermann, Streit. *HeAT - a Distributed and GPU-accelerated Tensor Framework for Data Analytics.* In 2020 IEEE International Conference on Big Data (Big Data) (pp. 276-287).

Data Distribution in HeAT

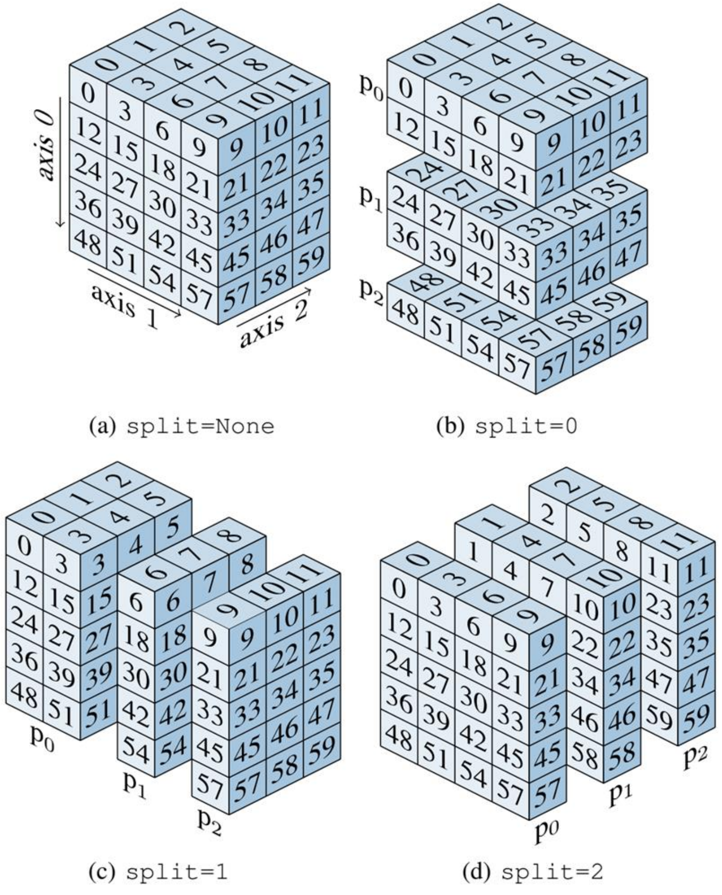
- A distributed HeAT tensor („**`DNDarray`**“) is composed of multiple local **`torch.Tensor`**‘s
- The tensor can be split along an arbitrary axis
- Example:

```
import heat as ht
# construct a range tensor
>>> range_data = ht.arange(6, split=1)
```

Server#1	Server#2	Server#3
[0, 1]	[2, 3]	[4, 5]

```
>>> range_data.mean()
2.5
>>> range_data.argmax()
5
```

- Parallelization utilizing **`MPI`** (“Message Passing Interface”) and **`mpi4py`**, GPU-support inherited from **`torch`**



(a) split=None (b) split=0

(c) split=1 (d) split=2

What is available?
(Excerpt)

All relevant options are available on **both CPU and GPU**

- *Global functions:*
  - Global and local setters and getters
  - mean, std, var
  - max, argmax, min, argmin
  - transpose
  - sort
- *Elementwise functions:*
  - cov
  - where
  - nonzero
- *Parallel IO using HDF5*
- *Linear Algebra:*
  - Matrix-matrix multiplication
  - Matrix-vector multiplication (dot)
  - QR
- *Analysis / Regression:*
  - K-means clustering
  - Spectral clustering
  - Lasso regression
  - DASO (data-parallel NN)
- *Basic building blocks:*
  - concatenate
  - diag
  - arange
  - Distributed random numbers
  - …

Currently being developed

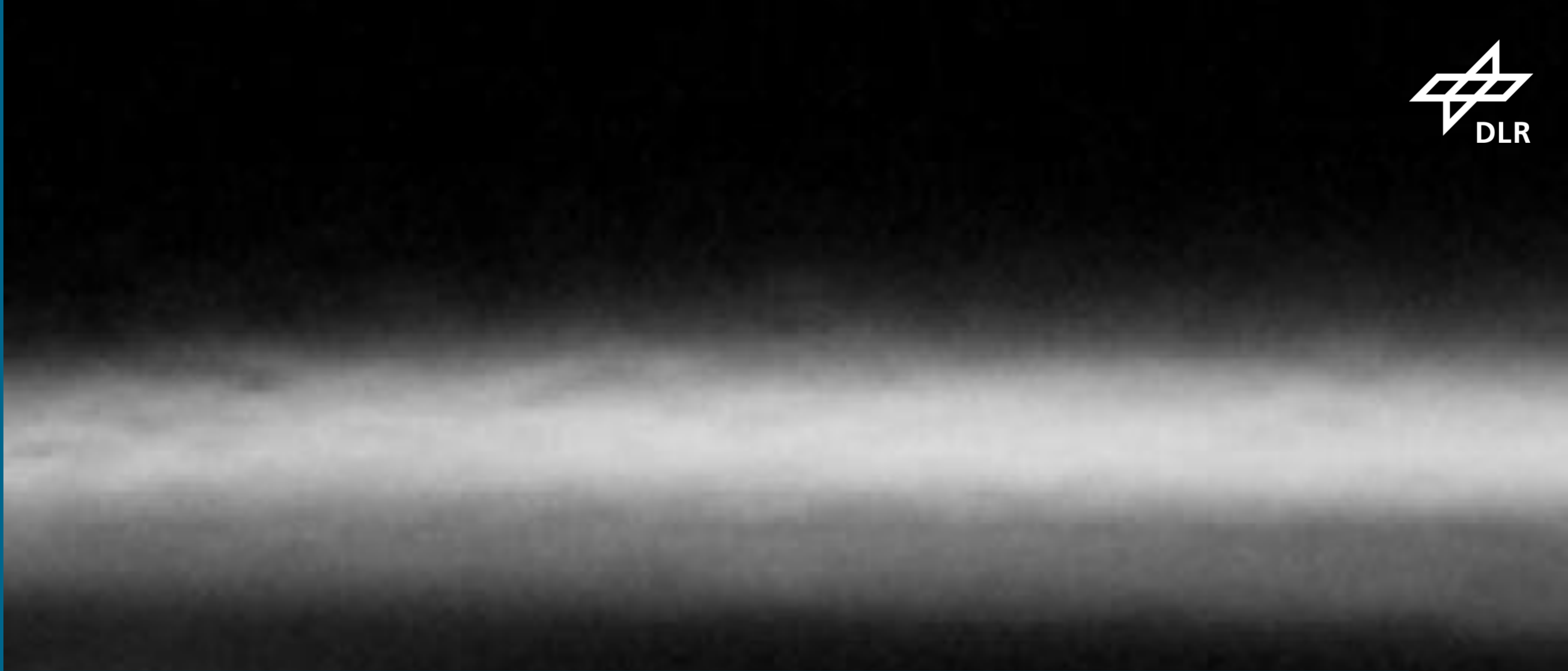
- **SVD (PCA)**
  - **work in progress: distributed hierarchical SVD** (hSVD, available as branch on github)
  - Future work: Zolotarev-SVD*
- Autograd / Automatic Differentiation

- K-nearest neighbours
- Logistic Regression
- Neural Networks
- Random Decision Trees
- SVM

- DBSCAN

- GMM
- ICA

*Nakatsukasa, Freund. *Computing fundamental matrix decompositions accurately via the matrix sign function in two iterations: The power of Zolotarev's functions.* SIAM Rev. 58, 3 (2016), 461--493.



Distributed hierarchical SVD

Distributed hierarchical SVD

- **„Big Data is Low Rank"*** – in many applications, it suffices to compute the first (i.e. largest) „few" singular values and vectors
- Exploit the specific structure of **`DNDarray`**'s: apply algorithms that can make use of **`torch`**-routines on the local arrays *in MPI-parallel* and somehow „merge" the results

The „distributed hierarchical" idea in a nutshell – Example: Computation of the mean value on 2 processes

Process 1 | **Process 2**

numbers $\underbrace{x_1, x_2, \dots, x_{m-1}, x_m}, \underbrace{x_{m+1}, x_{m+2}, \dots, x_{m+n-1}, x_{m+n}}$

Process 1: Compute „local" mean $\bar{x}^{(1)}$

Process 2: Compute „local" mean $\bar{x}^{(2)}$

Process 2: Send

Process 1: Receive „local" mean $\bar{x}^{(2)}$

„Merge":

Compute global mean by $\bar{x} = \frac{m}{m+n}\bar{x}^{(1)} + \frac{n}{m+n}\bar{x}^{(2)}$

* see, e.g.: Udell, Townsend. *Why are big data matrices approximately low rank?* SIAM J. Math. Data Sci., 1 (2019).

Distributed hierarchical SVD

DLR

Example: Compute *truncated* SVD on 2 processes

Process 1 Matrix $[A_1 | A_2]$ (split along columns) **Process 2**

Compute „local" SVD $A_1 = U_1 \Sigma_1 V_1^T$

Truncate to rank r and form $U_1[:, :r]\Sigma_1[:r, :r]$

Receive $U_2[:, :r]\Sigma_2[:r, :r]$

„Merge":

Compute the SVD of the concatenation

$$[U_1[:, :r]\Sigma_1[:r, :r] \mid U_2[:, :r]\Sigma_2[:r, :r]]$$

and *truncate* to rank r

Compute „local" SVD $A_2 = U_2 \Sigma_2 V_2^T$

Truncate to rank r and form $U_2[:, :r]\Sigma_2[:r, :r]$

Send

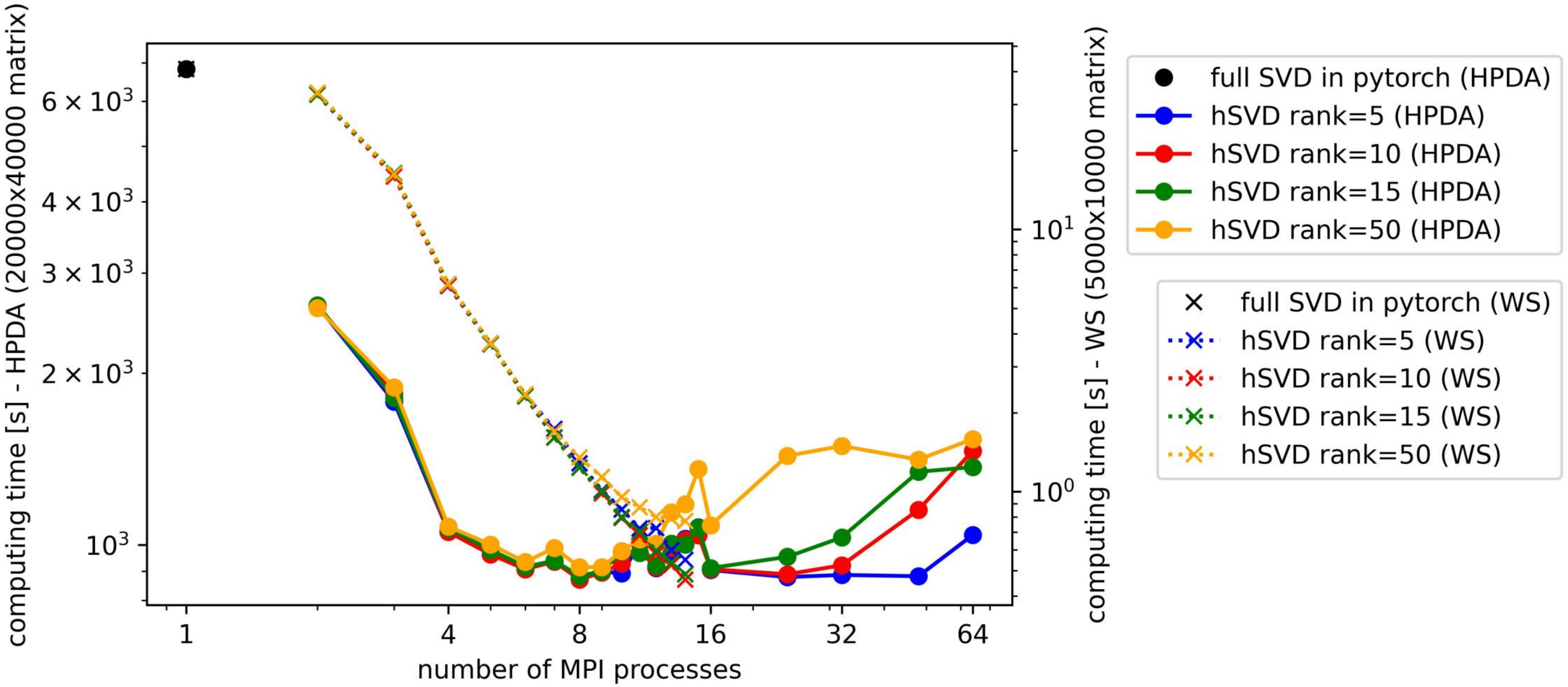
- Generalization towards „merging" along more complicated tree-structures is straightforward
- Theoretical analysis including error estimation/control is available

References:

Iwen, Ong. *A distributed and incremental SVD algorithm for agglomerative data analysis on large networks.* SIAM J. Matrix Anal. Appl., 37(4), 2016.
Himpe, Leibner, Rave. *Hierarchical approximate proper orthogonal decomposition.* SIAM J. Sci. Comput., 40 (5), 2018.

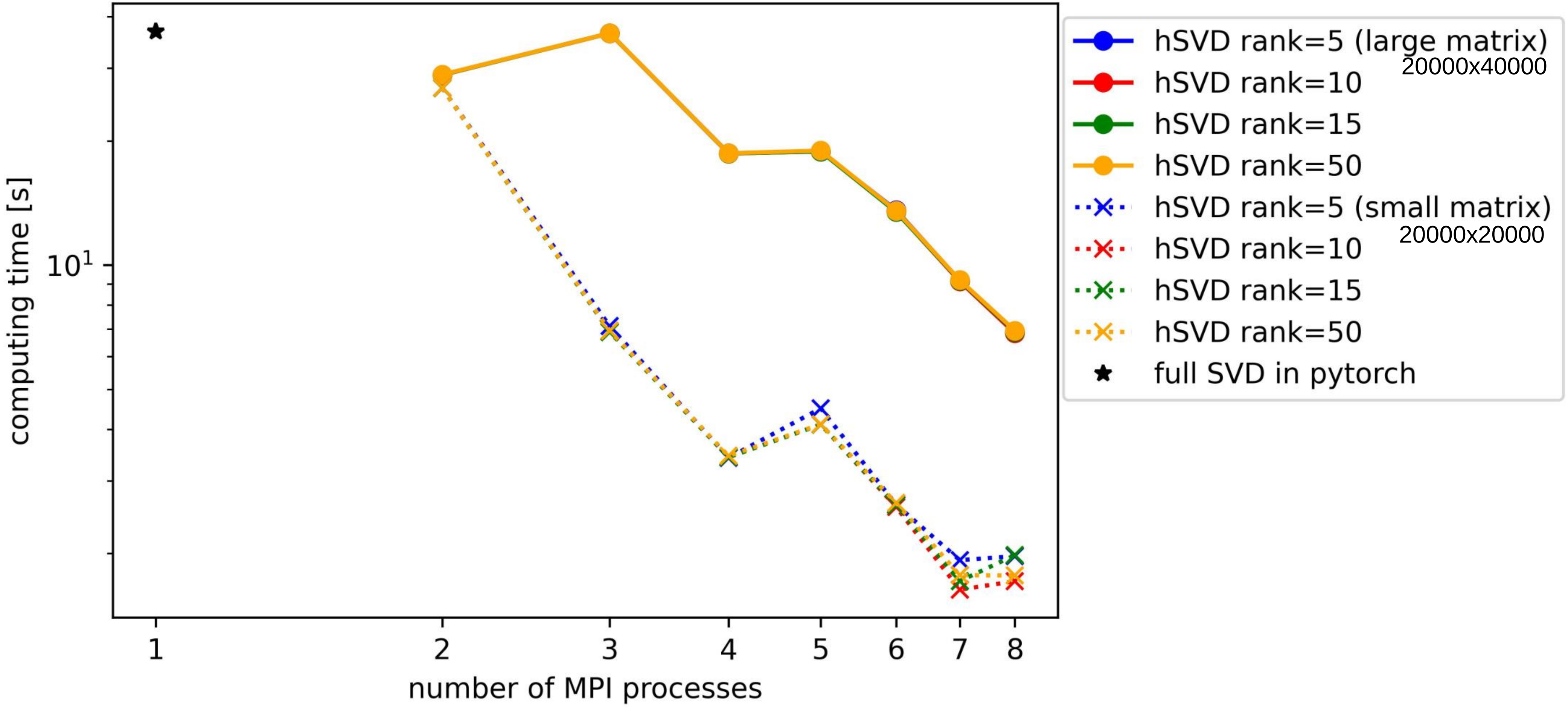
Numerical Experiment: hSVD on CPU

(Strong scaling)



Data: synthetic data set with rapidly decaying singular values [rel. truncation errors (in Frobenius norm): ~10% for rank 5, ~1% for rank 10 and < 0.002‰ for rank 15…]
Hardware: HPDA-Cluster SC-HPC, BS (=HPDA) and Linux Workstation, KP (=WS)

Numerical Experiment: hSVD on GPU
(strong scaling)



Data: synthetic data set with rapidly decaying singular values [rel. truncation errors (in Frobenius norm): ~10% for rank 5, ~1% for rank 10 and < 0.002‰ for rank 15…]
Hardware: HPDA-Cluster SC-HPC, BS

Thank you for your attention!