ARGONNE ATPESC2025 EXTREME-SCALE COMPUTING

I/O IN AI WORKLOADS

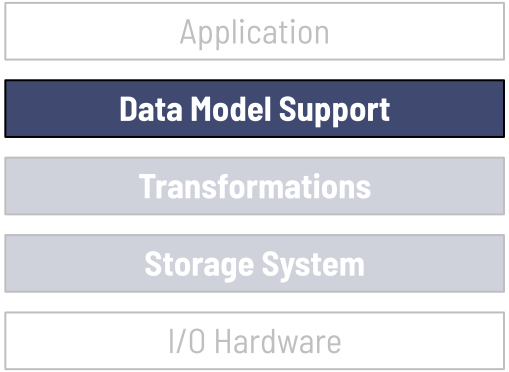
Jean Luca Bez
jlbez@lbl.gov
Lawrence Berkeley National Laboratory

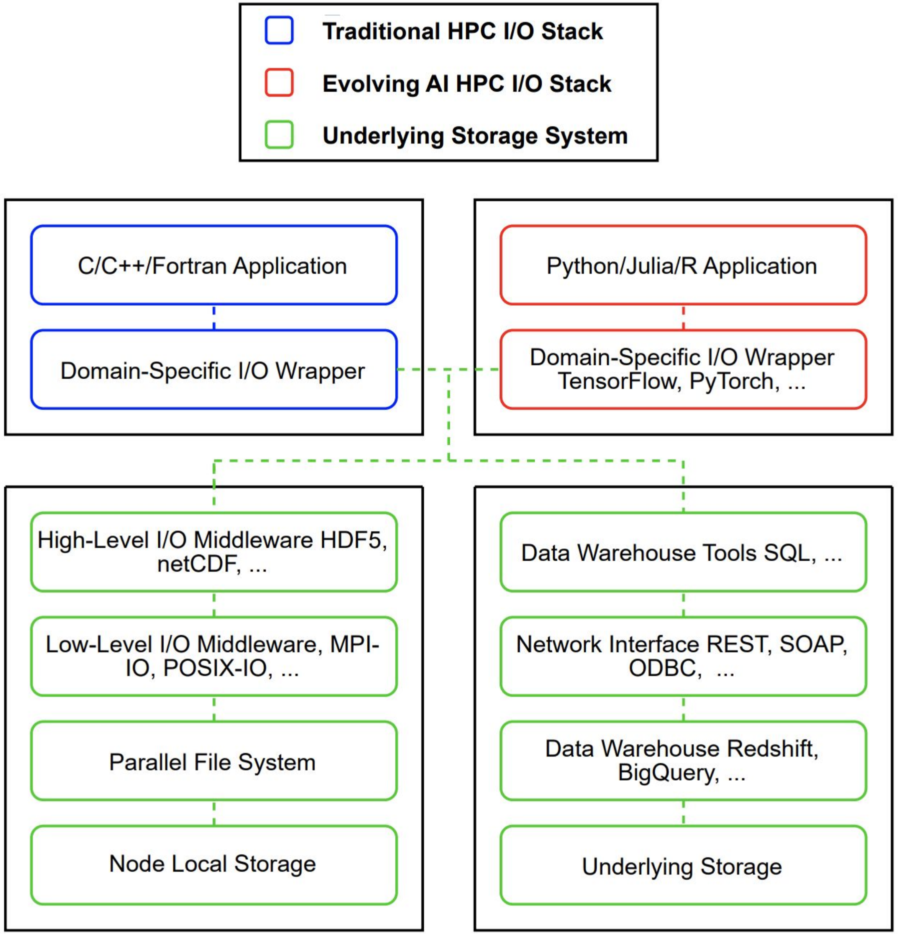
BERKELEY LAB

Scientific Data Division

NEW SCIENTIFIC COMPUTING PARADIGMS

- Understanding and improving I/O behavior in novel HPC applications and compute frameworks is critical to scientific productivity
- Large-scale **MPI** applications are still the **norm** at most HPC centers
- Other **non-MPI** compute frameworks are gaining **traction**:
  - AI/ML (TensorFlow, Keras, PyTorch, etc)
  - Data analytics frameworks (Dask, PySpark)
  - Other non-MPI distributed computing frameworks
- Many of these frameworks define their own **data models**, have their own mechanisms for **managing distributed tasks**, and demonstrate **unique I/O access patterns**




Traditional HPC I/O Stack
Evolving AI HPC I/O Stack
Underlying Storage System
C/C++/Fortran Application
Domain-Specific I/O Wrapper
Python/Julia/R Application
Domain-Specific I/O Wrapper TensorFlow, PyTorch, ...
High-Level I/O Middleware HDF5, netCDF, ...
Low-Level I/O Middleware, MPI-IO, POSIX-IO, ...
Parallel File System
Node Local Storage
Data Warehouse Tools SQL, ...
Network Interface REST, SOAP, ODBC, ...
Data Warehouse Redshift, BigQuery, ...
Underlying Storage

IS I/O A PROBLEM IN ML/AI?

- There has been some **debate** within the HPC community...

IS I/O A PROBLEM IN ML/AI?

- There has been some **debate** within the HPC community...
  - *"HPC systems often have enough node-local storage to cache the dataset"*

IS I/O A PROBLEM IN ML/AI?

- There has been some **debate** within the HPC community…
  - *“HPC systems often have enough node-local storage to cache the dataset”*
  - *“Not all HPC systems have large amounts of node local storage”*

IS I/O A PROBLEM IN ML/AI?

- There has been some **debate** within the HPC community…
  - *“HPC systems often have enough node-local storage to cache the dataset”*
  - *“Not all HPC systems have large amounts of node local storage”*
  - *“Datasets used are simply too large to be cached”*

IS I/O A PROBLEM IN ML/AI?

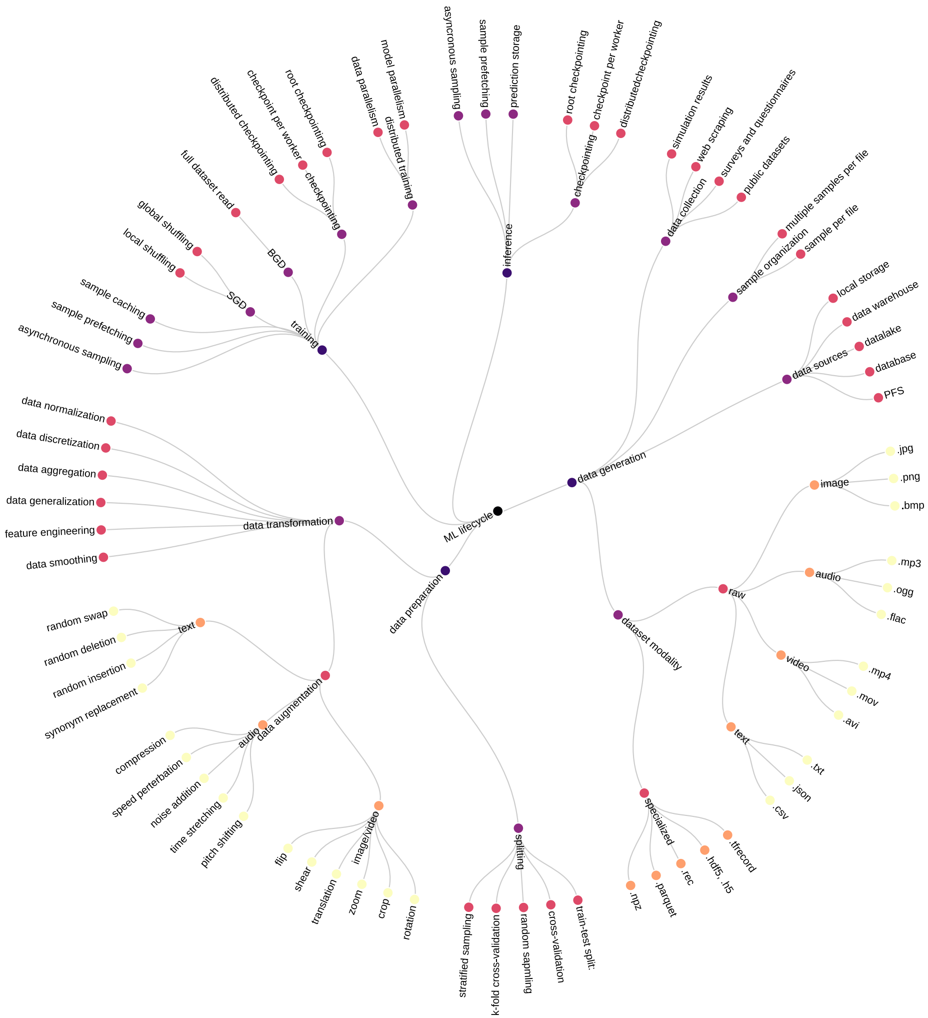
- There has been some **debate** within the HPC community...
    - *"HPC systems often have enough node-local storage to cache the dataset"*
    - *"Not all HPC systems have large amounts of node local storage"*
    - *"Datasets used are simply too large to be cached"*
    - *"Modern PFSs can often become an I/O bottleneck due to random sampling in training"*

IS I/O A PROBLEM IN ML/AI?

- There has been some **debate** within the HPC community…
  - *"HPC systems often have enough node-local storage to cache the dataset"*
  - *"Not all HPC systems have large amounts of node local storage"*
  - *"Datasets used are simply too large to be cached"*
  - *"Modern PFSs can often become an I/O bottleneck due to random sampling in training"*
- … in the end, it **depends** on the workload:
  - i.e., dataset size, storage capacity, storage system bandwidth, I/O library implementation and their configurations, and I/O access patterns of applications

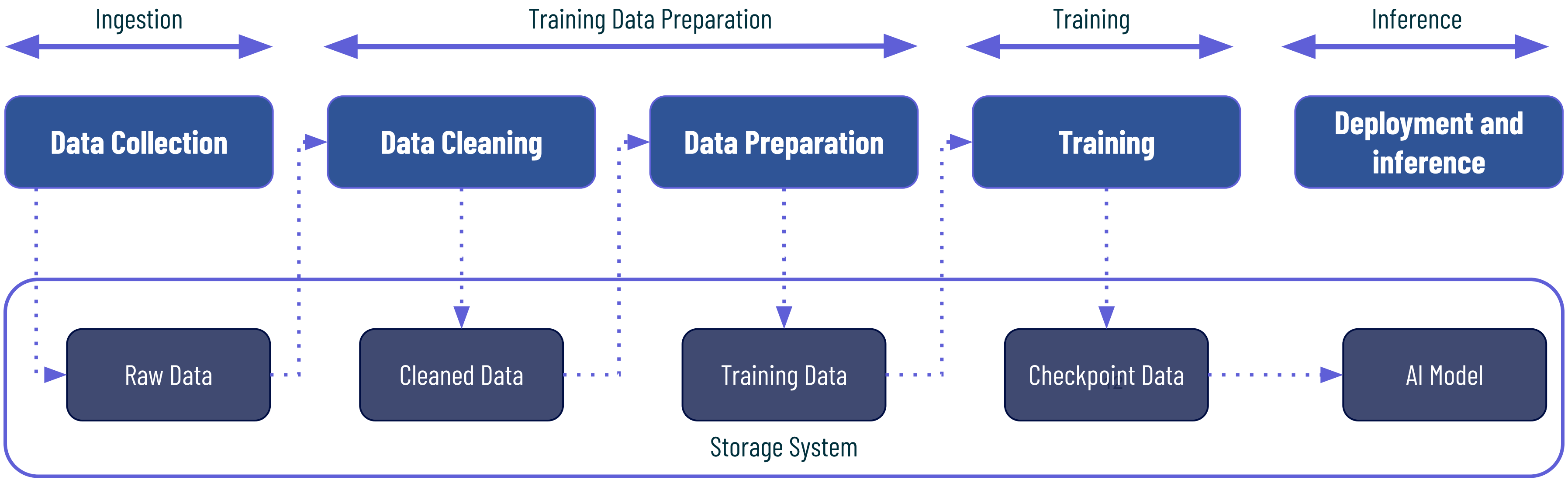
DATA MANAGEMENT DURING THE ML LIFECYCLE

- *"I/O in Machine Learning Applications on HPC Systems: A 360-degree Survey"*
    - https://doi.org/10.1145/3722215
- A taxonomy of data management during the ML lifecycle:
    - **Data Generation**
        - Different ways data is collected and the various data/file models used in application domains
    - **Dataset Preparation**
        - Various operations performed on the data to improve its quality
    - **Training**, and **Inference**
        - Commonly used data access patterns and I/O optimizations



ML lifecycle
data generation
data collection
simulation results
web scraping
surveys and questionnaires
public datasets
sample organization
multiple samples per file
sample per file
data sources
local storage
data warehouse
datalake
database
PFS
dataset modality
raw
image
.jpg
.png
.bmp
audio
.mp3
.ogg
.flac
video
.mp4
.mov
.avi
text
.txt
.json
.csv
specialized
.tfrecord
.hdf5, .h5
.rec
.parquet
.npz
data preparation
splitting
stratified sampling
k-fold cross-validation
random sapmling
cross-validation
train-test split:
data augmentation
image/video
flip
shear
translation
zoom
crop
rotation
audio
compression
speed perterbation
noise addition
time stretching
pitch shifting
text
random swap
random deletion
random insertion
synonym replacement
data transformation
data normalization
data discretization
data aggregation
data generalization
feature engineering
data smoothing
training
asynchronous sampling
sample prefetching
sample caching
SGD
local shuffling
global shuffling
BGD
full dataset read
checkpointing
distributed checkpointing
checkpoint per worker
root checkpointing
distributed training
data parallelism
model parallelism
inference
asyncronous sampling
sample prefetching
prediction storage
checkpointing
root checkpointing
checkpoint per worker
distributedcheckpointing

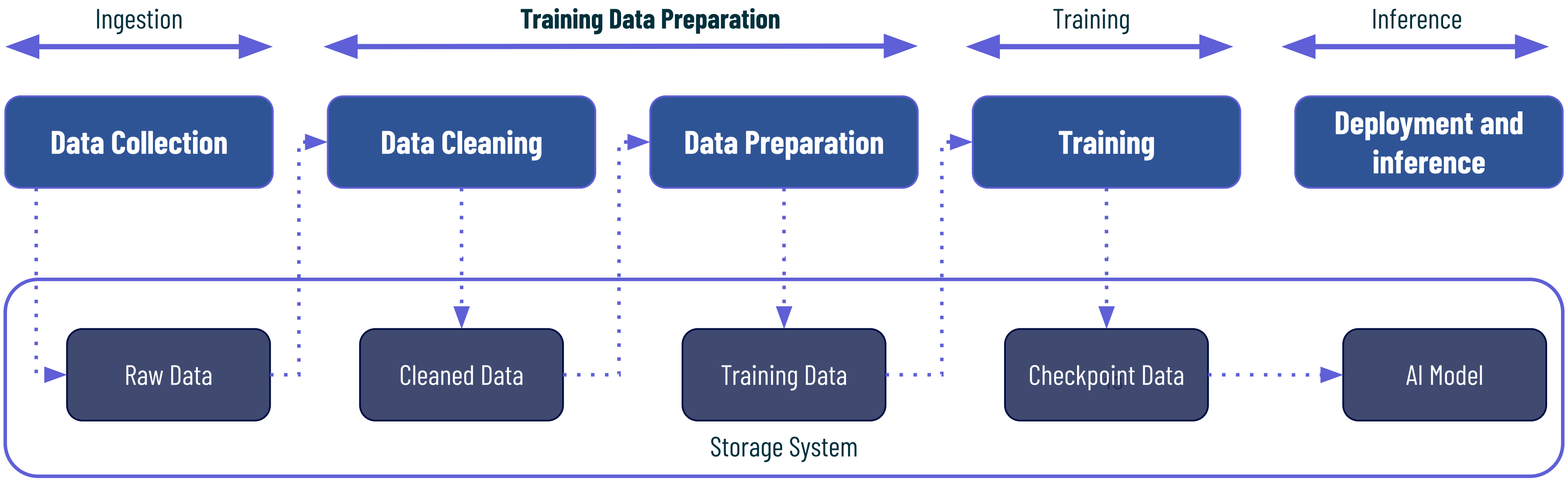
AI DATA / STORAGE PIPELINE



Based on the image adapted by Suren Byna from Jason Vallery's talk at SNIA Developer Conference

AI DATA / STORAGE PIPELINE

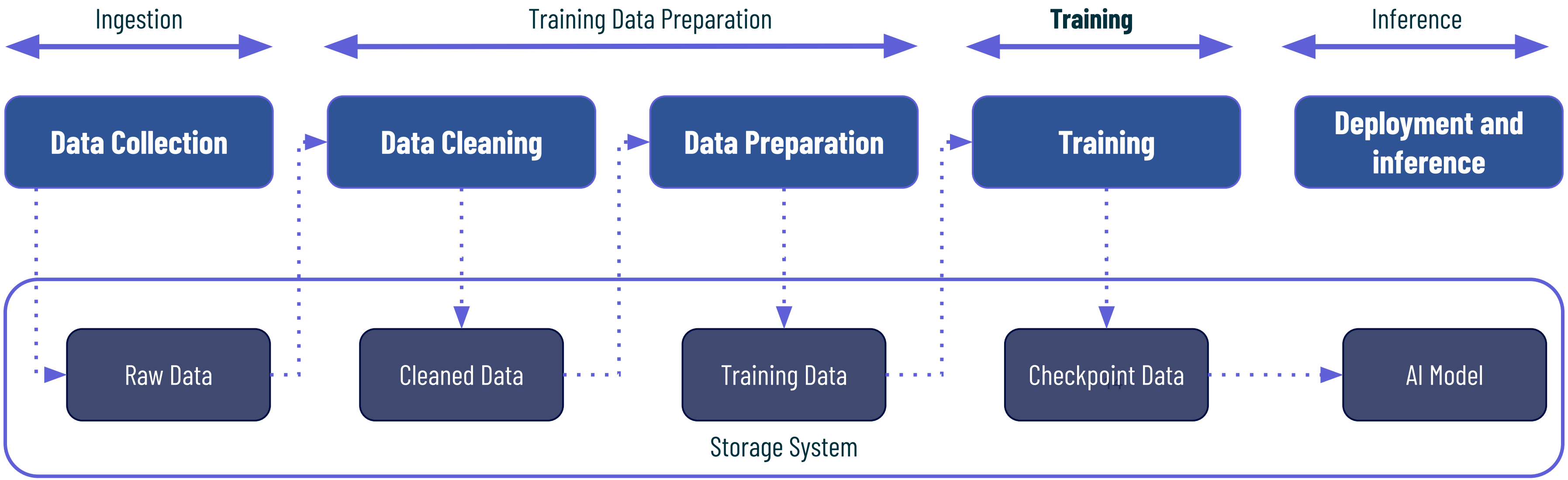
often **sequentially reads** a **lot** of data, and
writes back a **fewer** amount of data



Based on the image adapted by Suren Byna from Jason Vallery's talk at SNIA Developer Conference

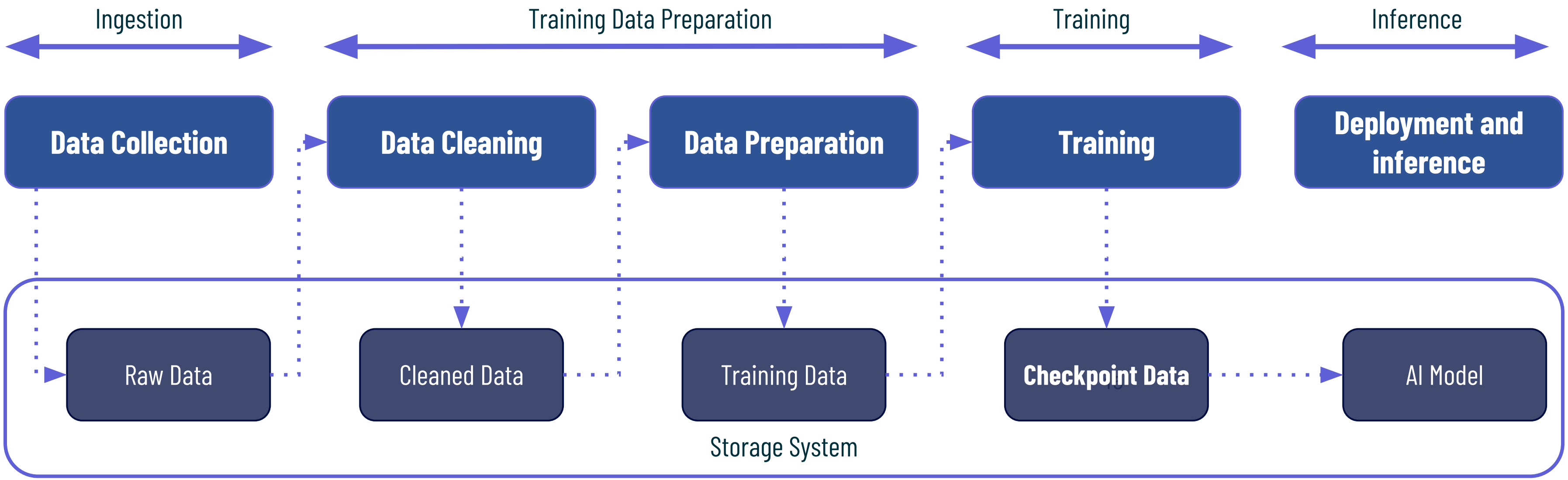
AI DATA / STORAGE PIPELINE

often **read-intensive** (and **random**)



Based on the image adapted by Suren Byna from Jason Vallery's talk at SNIA Developer Conference

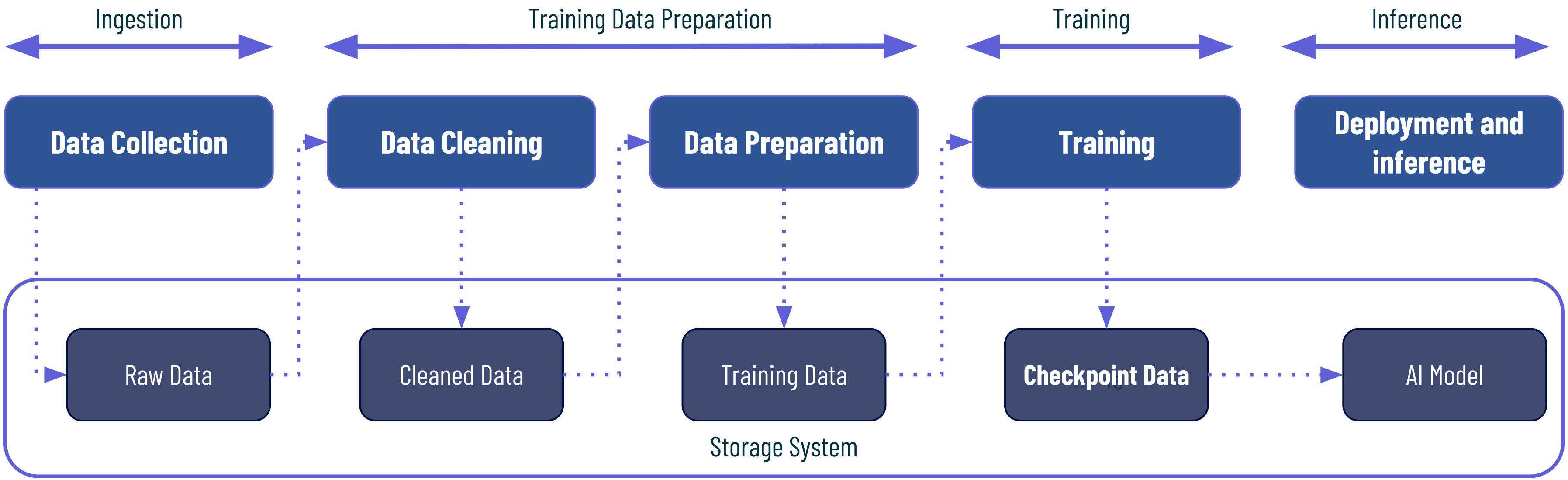
AI DATA / STORAGE PIPELINE



write frequently, as fast as possible

Based on the image adapted by Suren Byna from Jason Vallery's talk at SNIA Developer Conference

AI DATA / STORAGE PIPELINE

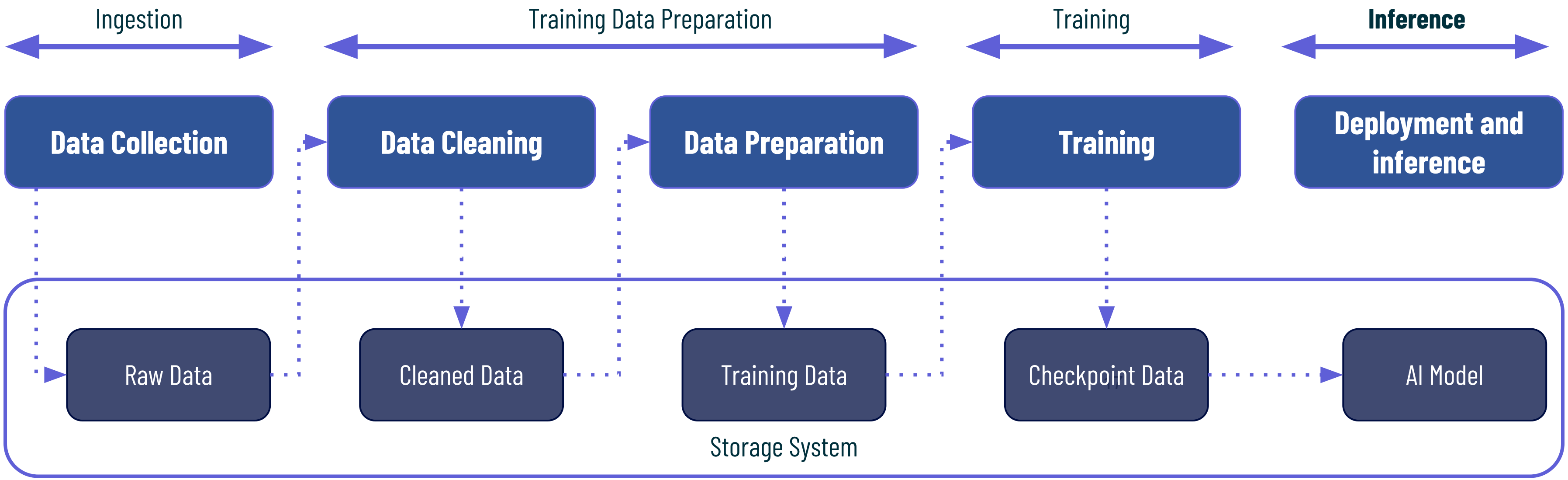


for restoring, **intense sequential reads**
for all nodes/GPUs involved

Based on the image adapted by Suren Byna from Jason Vallery's talk at SNIA Developer Conference

AI DATA / STORAGE PIPELINE

tends to be **I/O bound**, meaning GPU is often waiting on storage to provide data



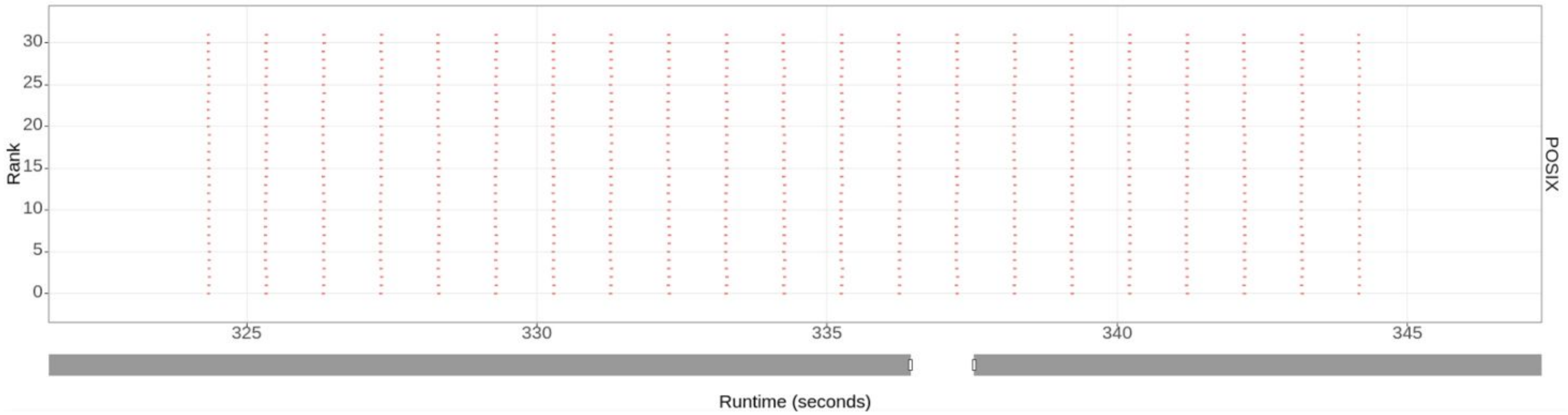
Based on the image adapted by Suren Byna from Jason Vallery's talk at SNIA Developer Conference

TUNING THE STORAGE SYSTEM

- I/O patterns **vary** depending on the **phase**
  - e.g., training uses random sampling
  - e.g. inferences favors contiguous reads
- **Optimizations** in one phase may not benefit others
  - e.g., caching samples speeds up training
  - e.g., caching may have less effect during inference

TUNING THE STORAGE SYSTEM

- Training may involve ranks reading batches of samples followed by rank syncs for model updates
- Training of **BERT LLM** using **DLIO** (https://github.com/argonne-lcf/dlio_benchmark)
  - Each rank reads a batch of samples (red dot), then synchronizes



BENCHMARKING I/O

- **DLIO** is a benchmark designed to simulate I/O access patterns found in Deep Learning (DL) workloads
  - Released as part of the **MLPerf Storage Benchmarks**
    https://github.com/mlcommons/storage
  - Selection of interfaces (HDF5, TFRecord, CSV, NPZ), file access patterns (one file per process versus shared file per process), data access patterns, I/O types, and transfer buffer sizes
- There are others that touch on I/O aspects, take a look at the paper

DARSHAN INSTRUMENTATION BEYOND MPI

- Darshan was re-designed to support **instrumentation** in **non-MPI** contexts as well:
  - Uses GCC-specific library constructor/destructor attributes to initialize/shutdown Darshan
- To enable non-MPI mode, users must **explicitly opt-in** by setting the `DARSHAN_ENABLE_NONMPI` env
  - A unique log will be generated for every **process** that executes
  - Often best to **limit** instrumentation scope to the target executable:

```
LD_PRELOAD=/path/to/libdarshan.so DARSHAN_ENABLE_NONMPI=1 ./exe <args>
```

CAVEATS FOR INSTRUMENTING PYTHON WITH DARSHAN

- Darshan initially enabling comprehensive instrumentation of a growing Python ecosystem in HPC:
    - Support for **non-MPI**, as **Python** often uses other mechanisms for parallelizing/distributing work

```
LD_PRELOAD=/path/to/libdarshan.so DARSHAN_ENABLE_NONMPI=1 python script.py <args>
```

    - Darshan library configuration support for **focusing scope** of Darshan instrumentation:

```
# exclude Python compiled code, shared libraries, etc.
NAME_EXCLUDE \.pyc$, \.so$, *

# pre-allocate 5000 POSIX records (default 1024)
MAX_RECORDS 5000 POSIX

# bump up Darshan's default memory usage to 8 MiB
MODMEM 8
```

- Otherwise, Darshan **exhausts its memory** and only instruments a **portion** of the application I/O workload

CHECKPOINTING

- Model **checkpointing** is a vital part of large model training
  - Number of model parameters continues to scale
  - Checkpointing is an **expensive** process
  - Involves **blocking** training progress in order to save out the latest model weights
  - Checkpointing commonly done by single rank, which can lead to stragglers
  - PyTorch recently added support for distributed checkpointing
- **Asynchronous** checkpointing modularizes the checkpointing process into **two parts**:
  - Copy the data from each GPU/rank from GPU to CPU
  - Asynchronously copy the data from CPU memory to disk to persist the checkpoint
- Once data is copied to CPU in the first phase, the GPU is free to immediately resume training

Reducing Model Checkpointing Times by Over 10x with PyTorch Distributed Asynchronous Checkpointing
By Meta: Lucas Pasqualin, Less Wright, Iris Zhang (PyTorch), Chien-Chin Huang; IBM Research: Swaminathan Sundararaman, Saransh Gupta, Raghu Ganti
https://pytorch.org/blog/reducing-checkpointing-times

- Try the model training checkpoint examples with **PyTorch**
  - https://github.com/raxid-io/hands-on/**ai-checkpoint**
  - Detailed instructions available in the README
- Remember to collect **Darshan logs** and **traces**!
- What should I **look at**?
  - What can you infer about the application I/O **behavior** from Darshan's report?
  - What is the **I/O bandwidth** and **time**?

- For **Aurora**, some **changes** are needed:
    - You will **need** to use `intel_extesion_for_pytorch` and `oneccl_bindings_for_pytorch`

      https://docs.alcf.anl.gov/aurora/data-science/frameworks/pytorch.html#code-changes-to-train-on-multiple-gpus-using-ddp
    - You will also need to set the **proxy** host to download the model

      https://docs.alcf.anl.gov/aurora/getting-started-on-aurora/#proxy
- Notice that DCP requires a newer version of PyTorch (not in Aurora)

ARGONNE ATPESC2025 EXTREME-SCALE COMPUTING

I/O IN AI WORKLOADS

Jean Luca Bez
jlbez@lbl.gov
Lawrence Berkeley National Laboratory

BERKELEY LAB

Scientific Data Division