Data-handling and infrastructure

Florian Pinault

Data Handling and infrastructure

Questions

How do I parallelise part … of my workflow?

How do I transfer my data to … ?

Should I use more memory/machines/nodes/GPUs?

Which data should I copy? Where?

Which format should I choose for my data?

How to make “it” fast?

Data Handling and infrastructure

Questions

How do I parallelise part … of my workflow?

How do I transfer my data to … ?

Should I use more memory/machines/nodes/GPUs?

Which data should I copy? Where?

Which format should I choose for my data?

How to make "it" fast?

Know the technology

HPC (High Performance Computing)? Cloud?

S3 buckets vs Lustre filesystems?

"Bring the code close to the data instead of data to the code."

"Cloud-friendly" format?

Know your dataset

Total size on disk?

Total uncompressed size?

How many files?

Any missing data? Nans ?

Dimensions of the data? Full n-dimensional array? Several arrays?

For machine learning :

What is the size of one training sample? Of one batch?

Where is the data from? On which data will I run inference? How?

Know your read/write patterns

Random read (shuffling)

Transpose the data if needed

Universal answer

➔“It depends”

The best solutions will usually depend

- on the size of the data
- on the project requirements
- on the available funding
- on previous experience
- on personal preferences
- and more…..

“When you have a hammer, everything looks like a nail.”

Know your access patterns : transposition

Example data

9 fields (f)

14 dates (p)

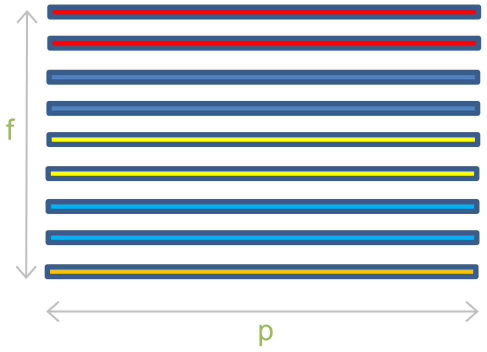
Real data

100 fields

1M dates

+ additional dimensions

Data saved by field

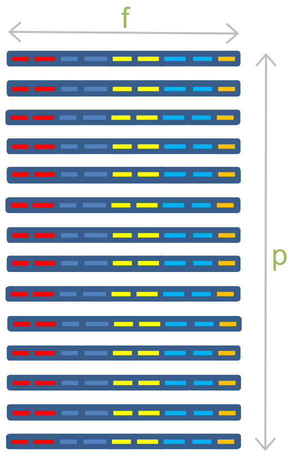


One file contains the timeseries of one given **field**

9 fields

≠

Data saved by date

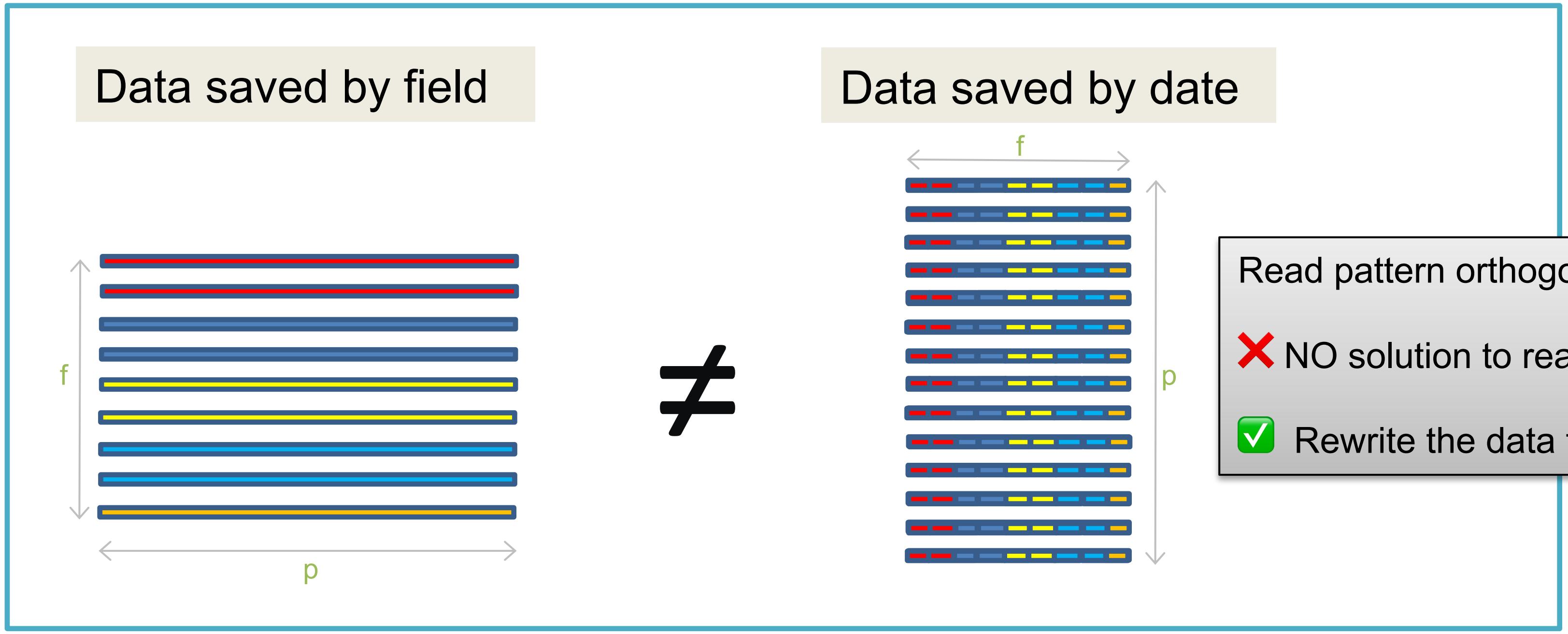


One file contains the set of values for one given **date**

9 dates

More generally:
Each "file" here could be an S3 object, or a part of a file, a record in a database, etc.

Know your access patterns : transposition

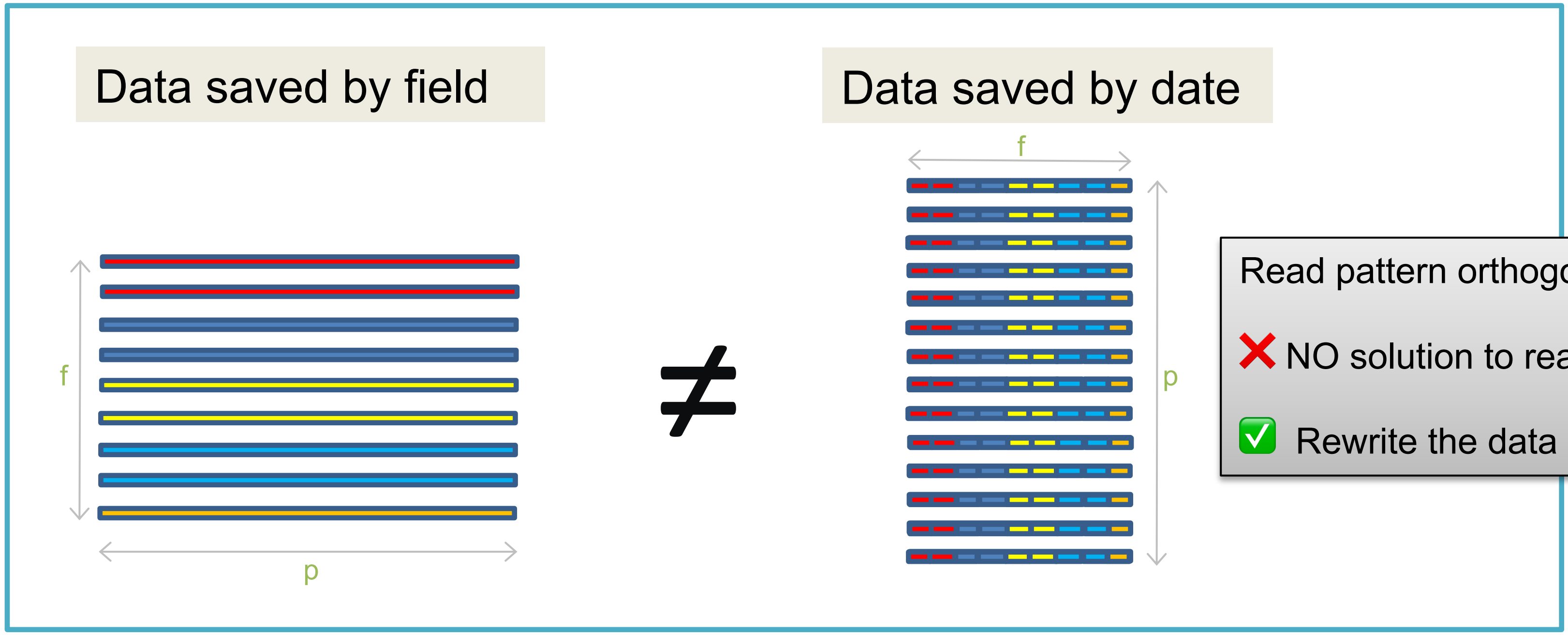


Read pattern orthogonal to the pattern on disk ?

✗ NO solution to read big data efficiently

✓ Rewrite the data to adapt to the read pattern

Know your access patterns : transposition



Read pattern orthogonal to the pattern on disk ?

❌ NO solution to read big data efficiently

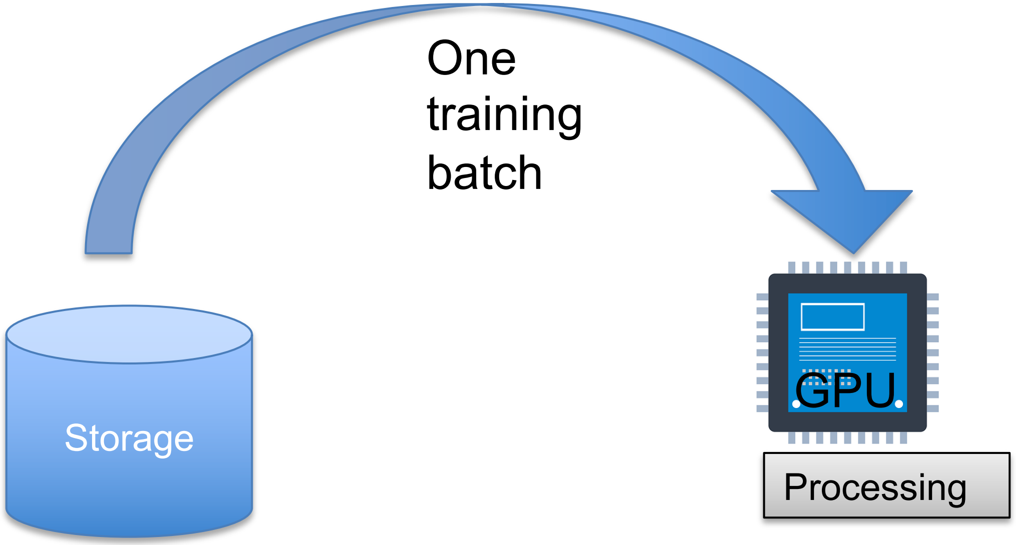
✅ Rewrite the data to adapt to the read pattern

- If you need to read data by timeseries. Store the data by timeseries.
- If you read training samples with 100 parameters for 1 date, do save files containing 100 parameter for 1 dates (but ”it depends”)
- General solution: create a dataset dedicated to your training task (perform the transposition **offline** if required).

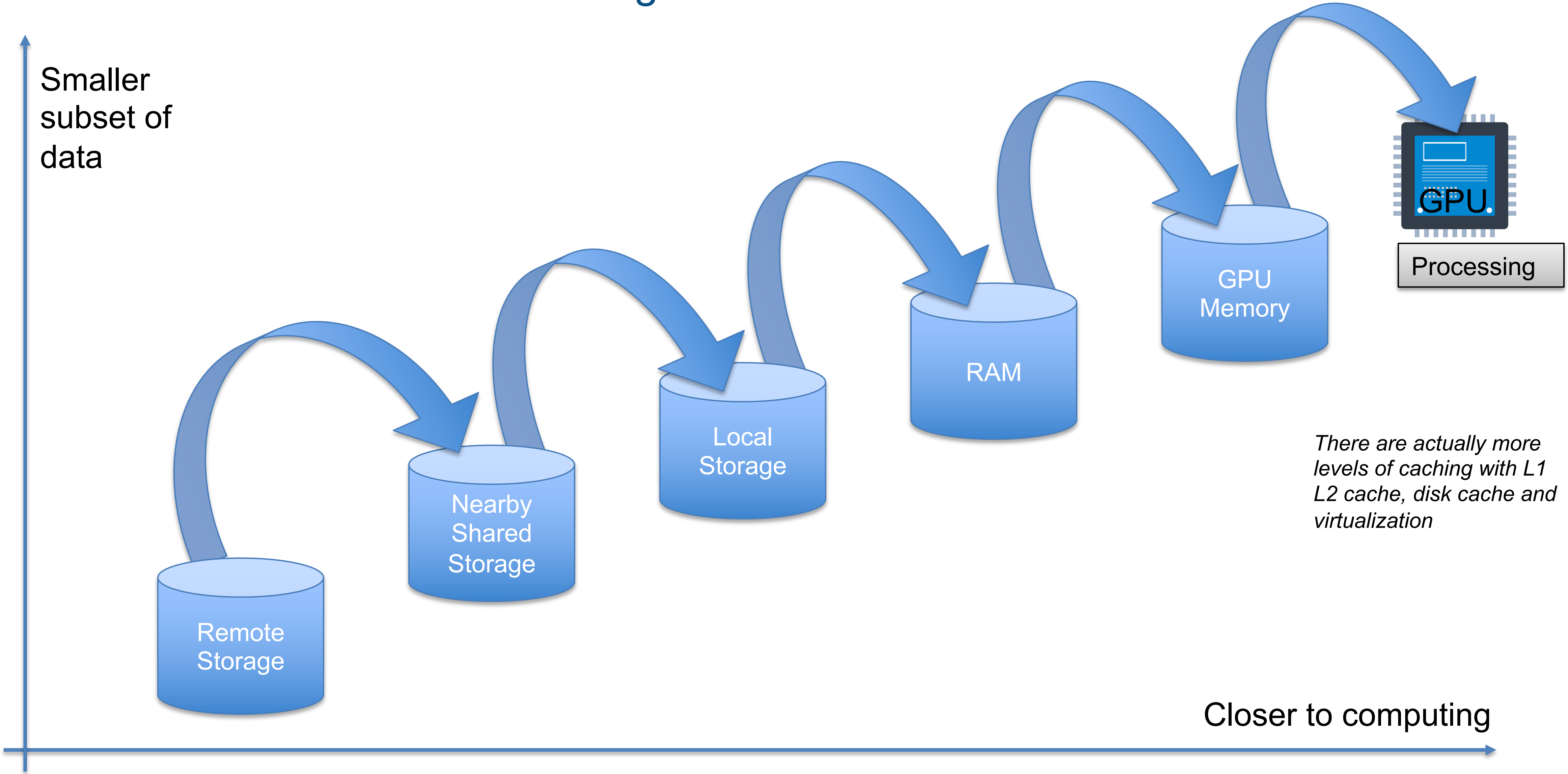
Data flow: Feeding the GPU with data

- Data is on disk
- We need to move it to the GPU
- One main requirement :

  Faster than GPU processing

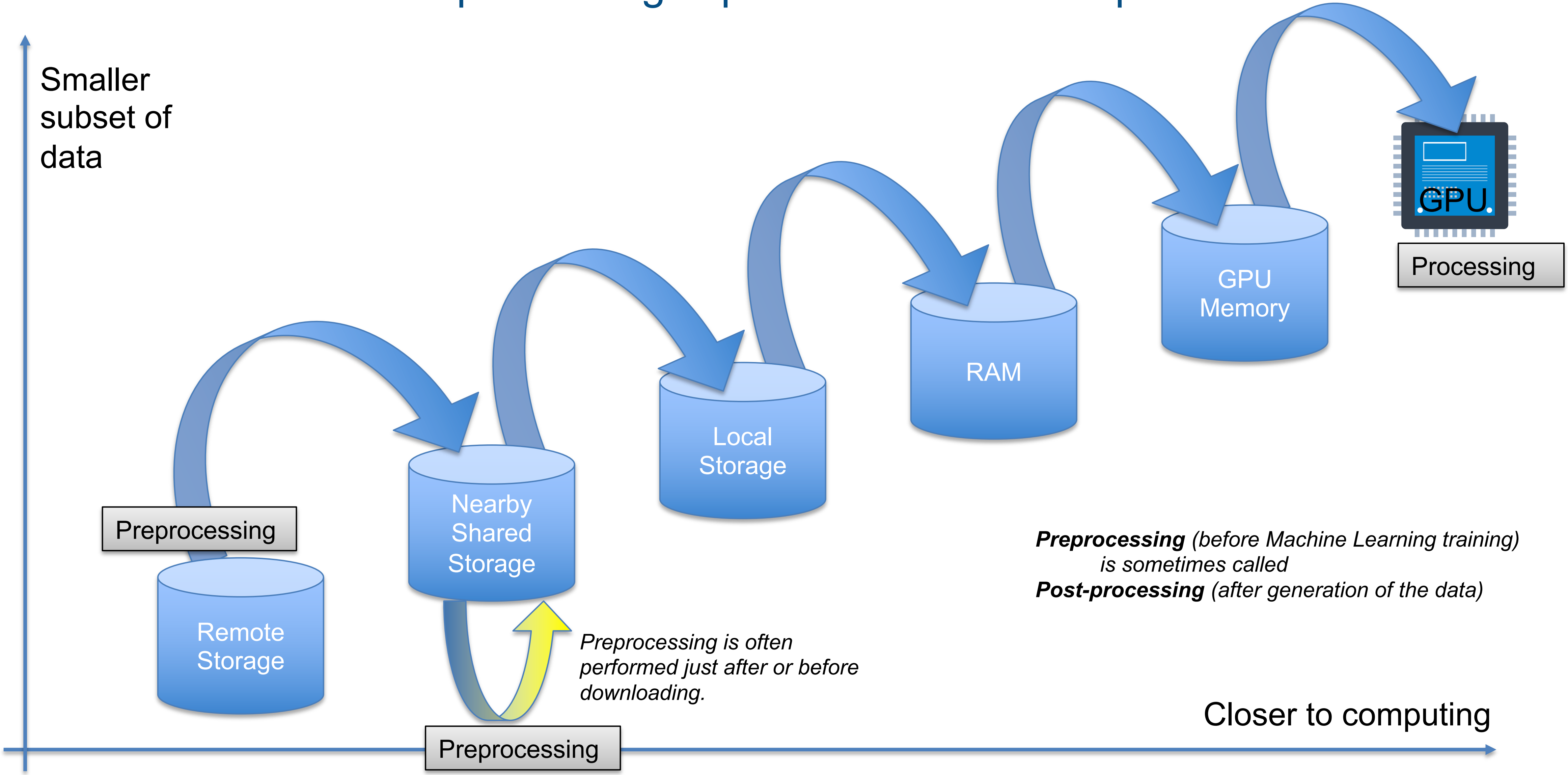


Data cascade of caching



There are actually more levels of caching with L1 L2 cache, disk cache and virtualization

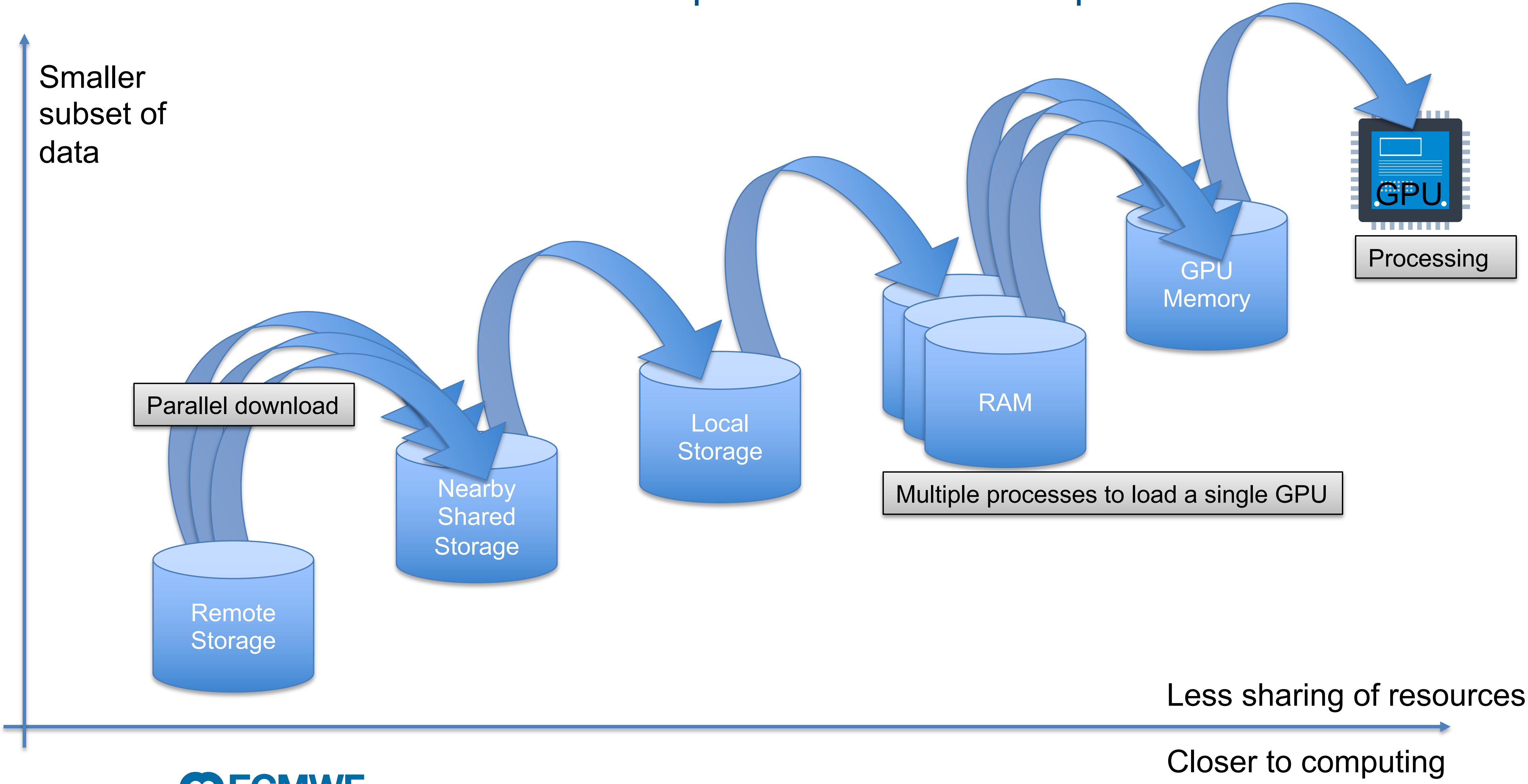
Data flow: Preprocessing is possible on each step



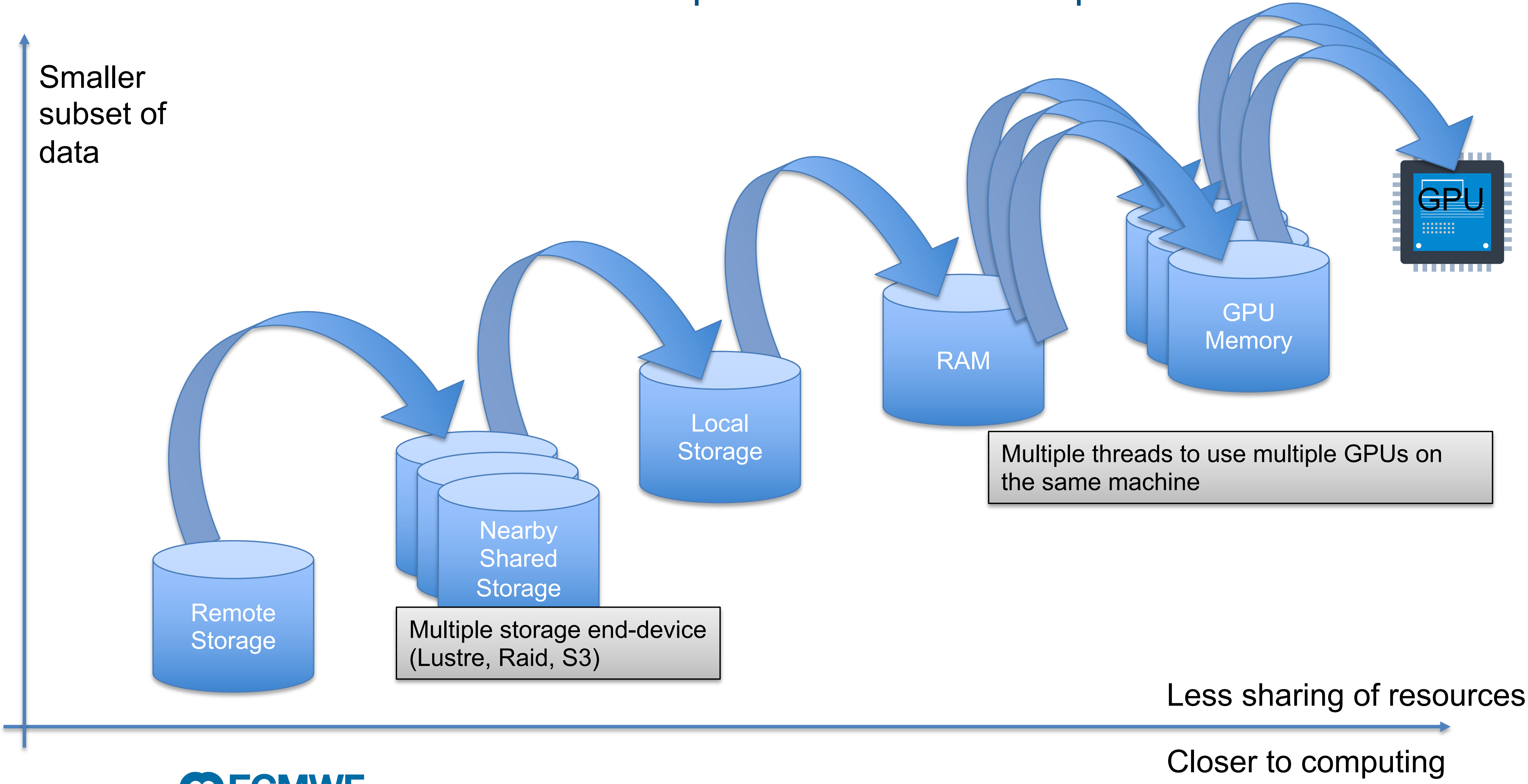
Preprocessing is often performed just after or before downloading.

***Preprocessing** (before Machine Learning training) is sometimes called **Post-processing** (after generation of the data)*

Data flow: Parallelisation is possible on each step



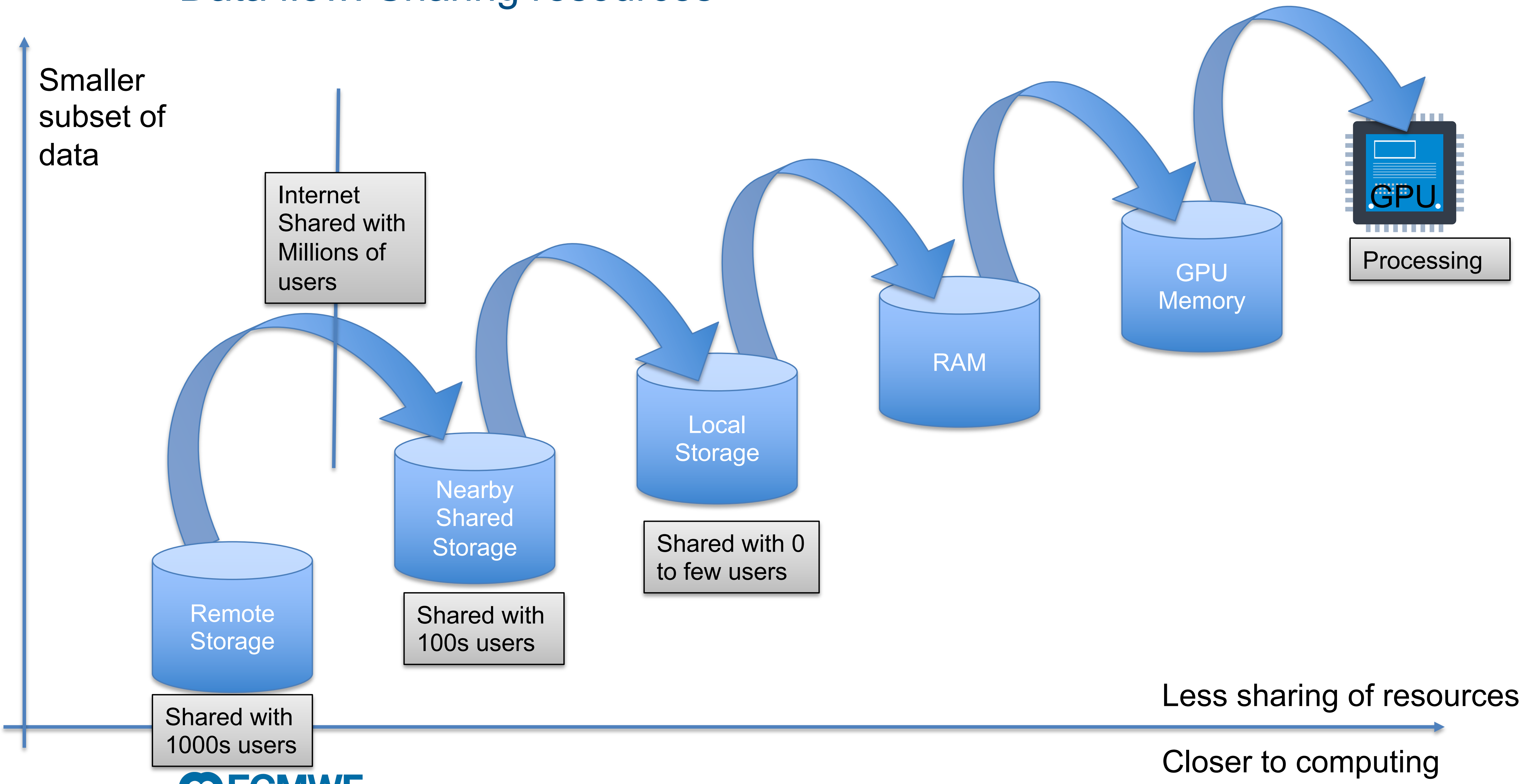
Data flow: Parallelisation is possible on each step



Multiple storage end-device (Lustre, Raid, S3)

Multiple threads to use multiple GPUs on the same machine

Data flow: Sharing resources



Clouds and HPC

Clouds and HPC

A **cloud** is a set of shared computers

An **HPC** is a set of shared computers

Clouds and HPC : Examples of clouds

"Public" clouds, from private companies : Microsoft (Azure), Amazon (AWS), Google, others

- Each cloud provider promotes their own ML platform
- Some have nice Graphical User Interface (GUI) and great automation tools
- Work well on toy problems, try them with free credit
- Sometimes difficult to cap the expense

Publicly funded clouds, related to ECMWF

- ECMWF EWC (European Weather Cloud), federated with EUMETSAT EWC.
- CCI (Common Cloud Infastructure)
- CDS (Copernicus Data Store) not really a cloud, but it has a toolbox

Clouds and HPC: Environments and virtualisation

Various solutions to set up a [python] environment:

- `pip install [--user]` (✗ bad practice? But “it depends”)
- `pip install` on a python virtual env (`python -m venv /path/to/new/env`)
- `pip install` on a conda virtual env
- `conda/mamba install` on a conda virtual env
- Use a Jupyter server as an additional layer (collab, deepnote, binder)
- Docker (cloud) / Apptainer (HPC) container
- Virtual machine (cloud only)
- System install (cloud only, but it depends)

You can stack them all !

➔ Use what you know.

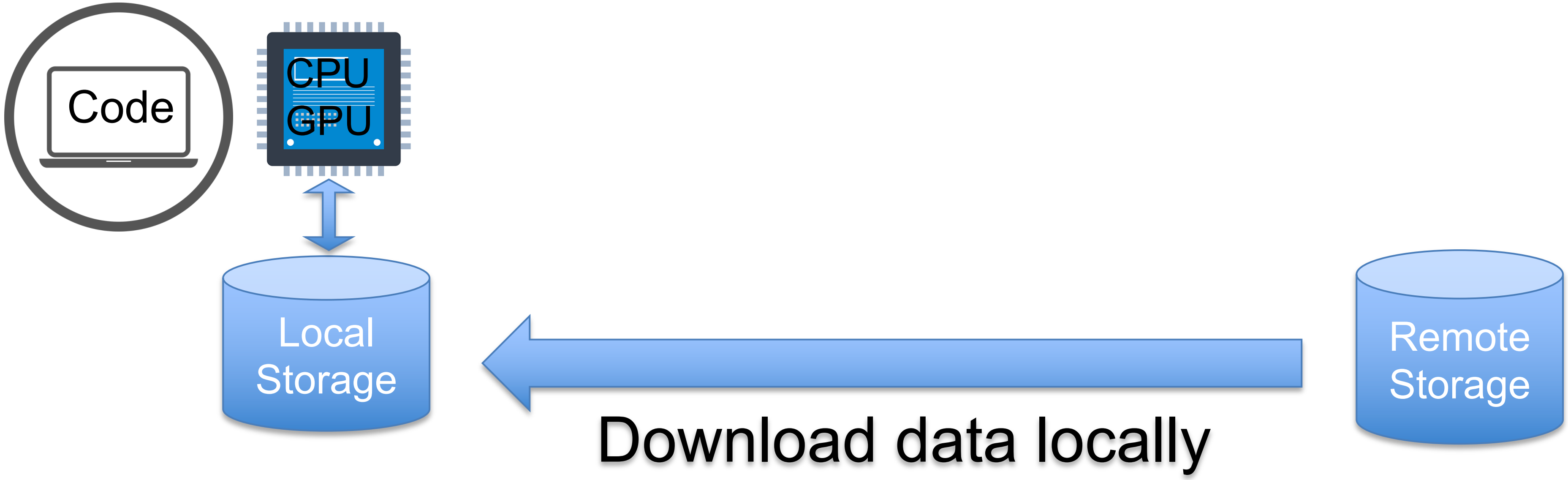
➔ Stack as few as possible.

General software ideas

- Use what you know. But use python, and pytorch. And virtual env.
- Use VS-Code. With SSH-remote plugin.
- Use git.
  And github
  And github actions, with black, and isort
- Make your code open-source

Clouds and HPC

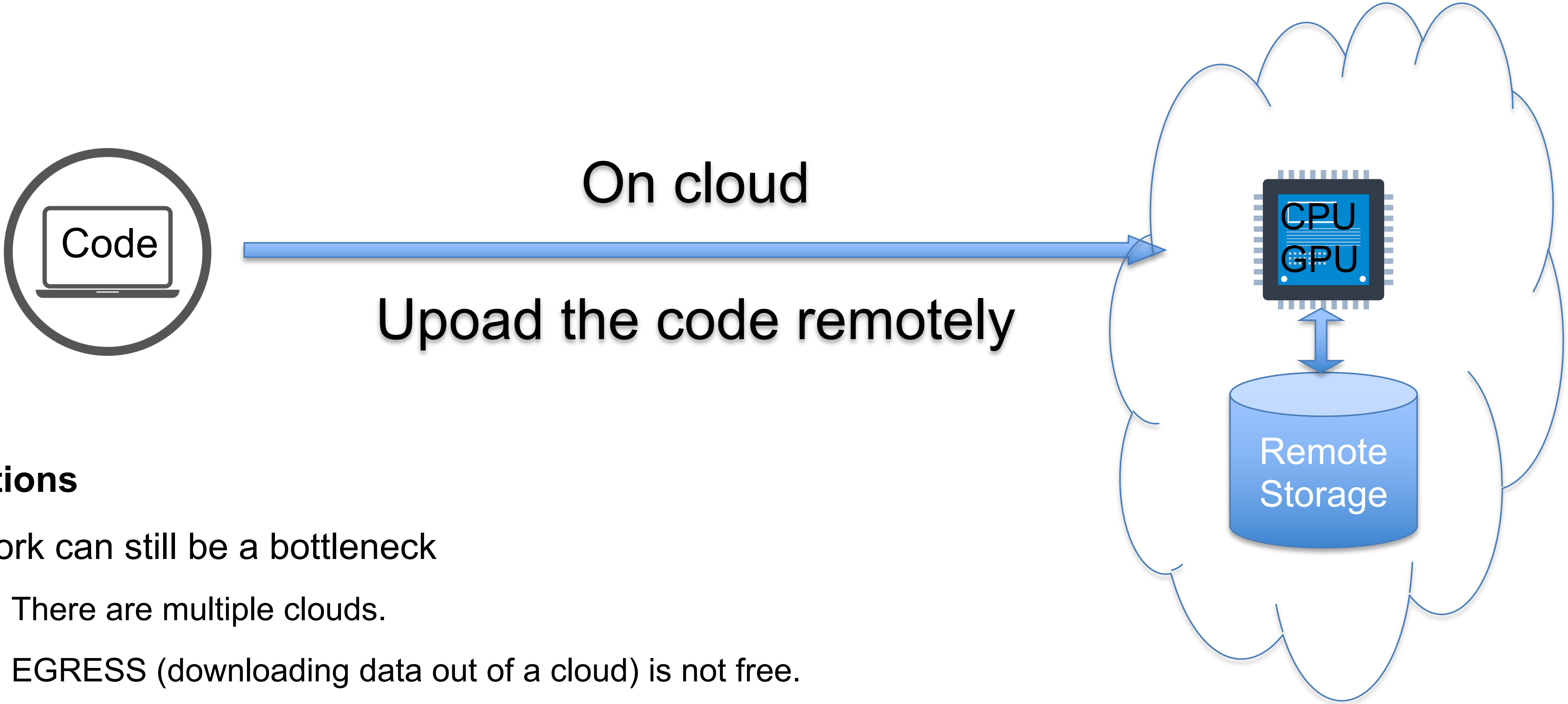
“Bring the code close to the data instead of data to the code.”



Download data locally

Clouds and HPC

“Bring the code close to the data instead of data to the code.”

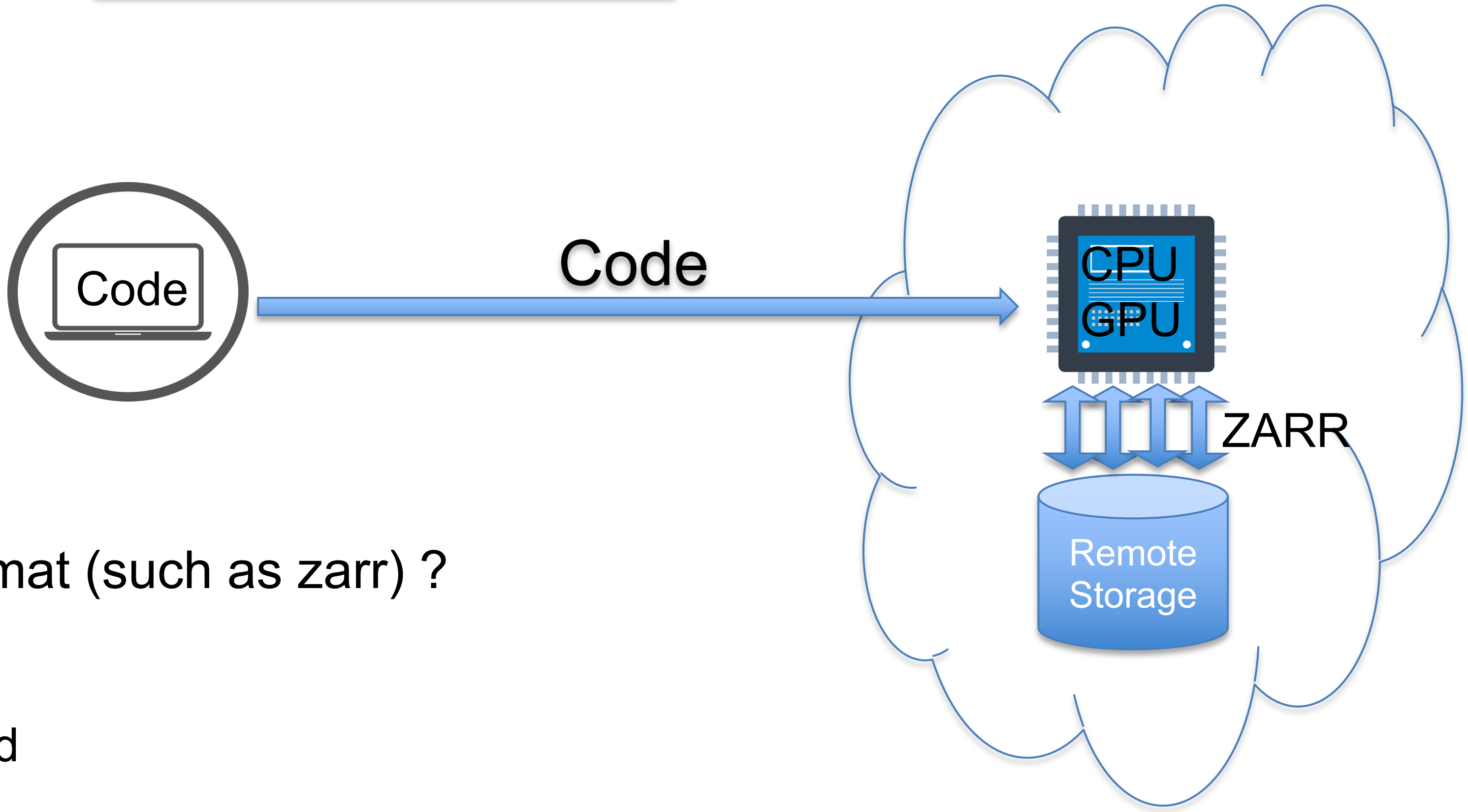


Implications

- Network can still be a bottleneck
  - There are multiple clouds.
  - EGRESS (downloading data out of a cloud) is not free.
- Data is processed by the GPU/CPUs located in the cloud instead of locally

Clouds and HPC

“Cloud-friendly” format ?



What is a “cloud-friendly” format (such as zarr) ?

- Combined with parallel backend
  - Such as S3
- Zarr targets massively parallel I/O

ECMWF

Clouds and HPC : S3 (clouds) vs Lustre (HPC)

- What is similar between S3 and Lustre ?

  - Is a set of shared computers

    Actual data is written on actual hard disks

    Data communication on actual network cables

  - Concurrency amongst users.

    Higher performance is accessible with more redundancy

  - Network bandwidth is a limiting factor

  - Metadata server(s), with load balancing

  - One file/object can be split into parts

    Allowing parallel read and write

  - Many layers of caching

  - Sharing data saves resources

Clouds and HPC : S3 (clouds) vs Lustre (HPC)

- Selected differences that matter

  - S3 uses HTTP protocol,

    No firewall issues.

  - Different permissions management

    Credentials vs linux users.

  - Limited Random-access on S3

    Accessing part of a file.

  - No lock mechanism on S3.

  - Different configurations are implemented by different human beings with different design decisions.

    Lustre sometimes optimized for write patterns,

    S3 sometimes optimized for read patterns.

Recent and future evolutions

- Ai-models
  - Python package to run Machine Learning models for meteorology
  - Relies on plugins to run inference for various models (AIFS, GraphCast, FourcastNet, FuXi, PanguWeather…)
  - Runs operationally

- climetlab is evolving into earthkit-data
  - Mostly compatible with climetlab API
  - Targeting operational use
  - climetlab plugins will be supported (some changes will be required for the migration)

- **In preparation: anemoi**
  - Tools to build meteorological datasets, catalogue of datasets, library of training components, tracking experiments
  - Open-source
  - Community-oriented

Thank you !

More comments ?