Image credit: Marguerite Tonjes

Storage for HEP

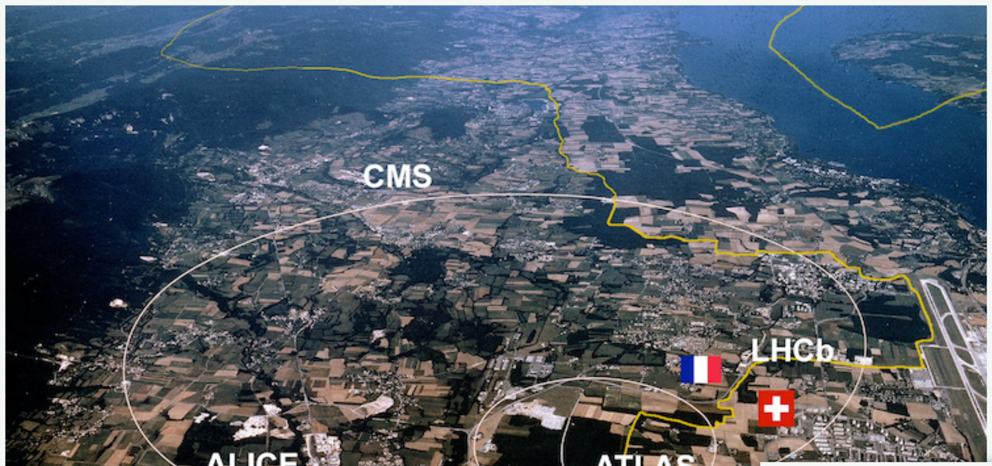
Nick Smith

Computational HEP Traineeship Summer School

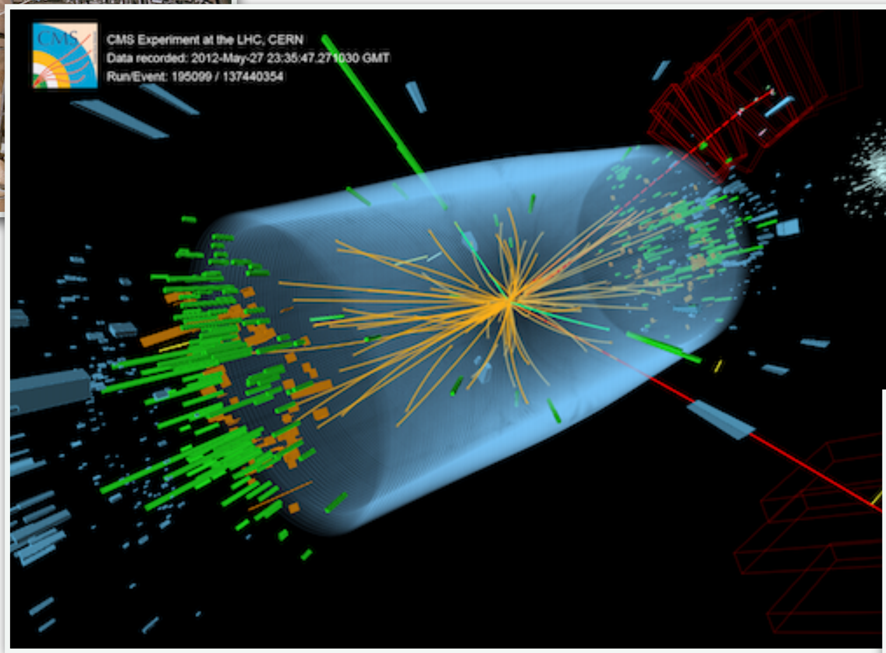
23 May 2024

HEP Experiment: three easy steps

1. Collide particles

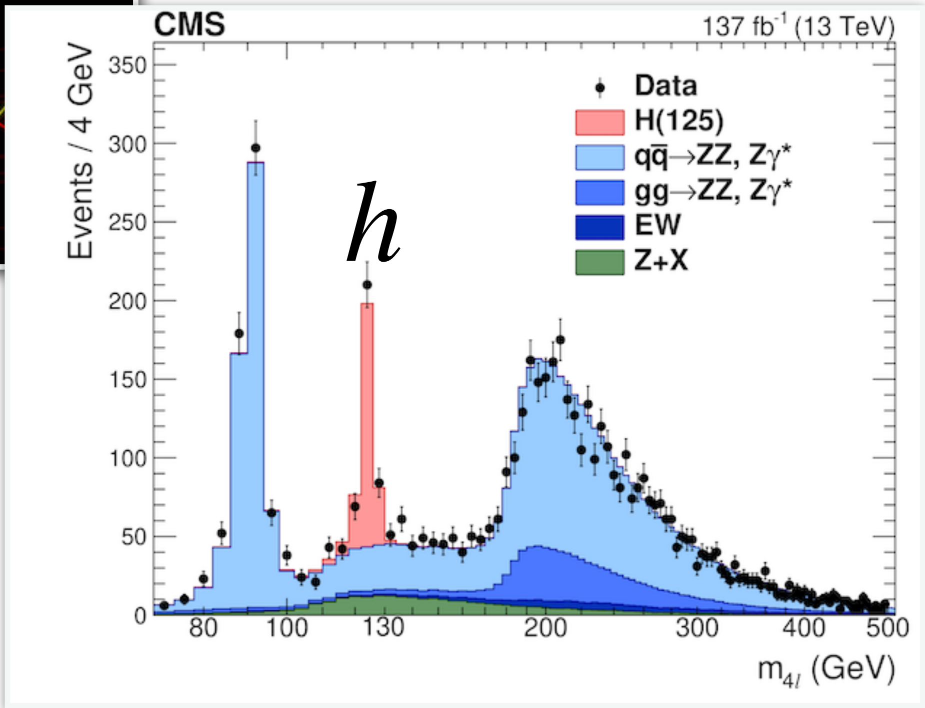


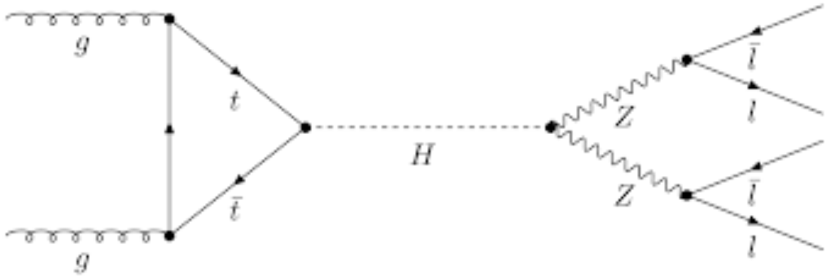
2. Take pictures


CMS Experiment at the LHC, CERN
Data recorded: 2012-May-27 23:35:47.271030 GMT
Run/Event: 195099 / 137440354

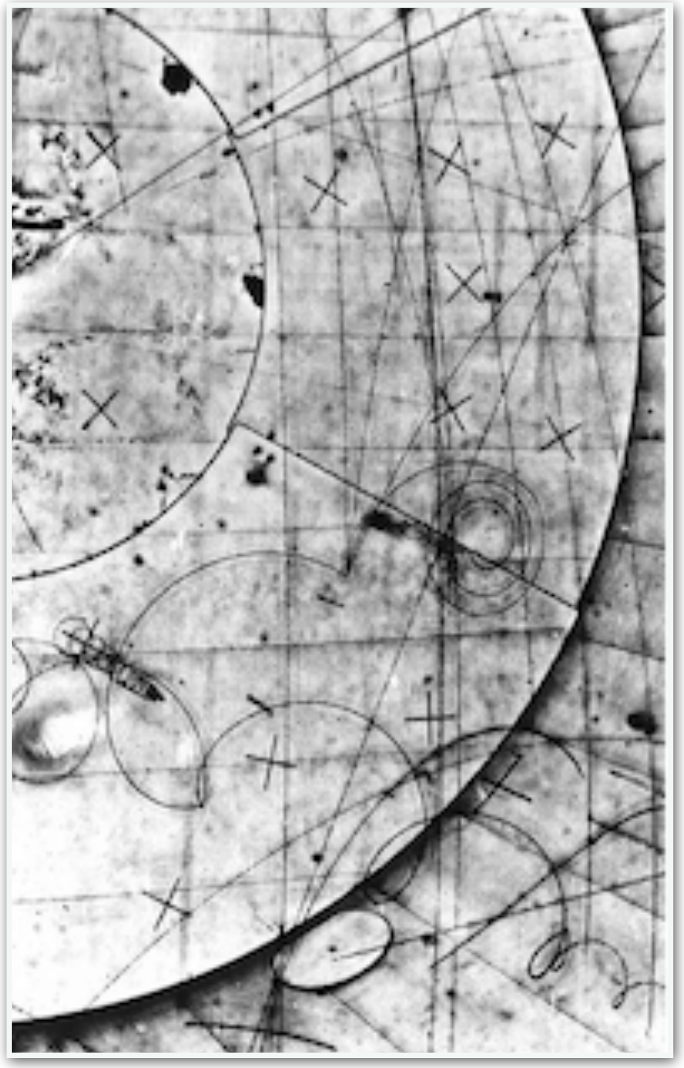
3. Infer parameters

~1MB / event
~100B events





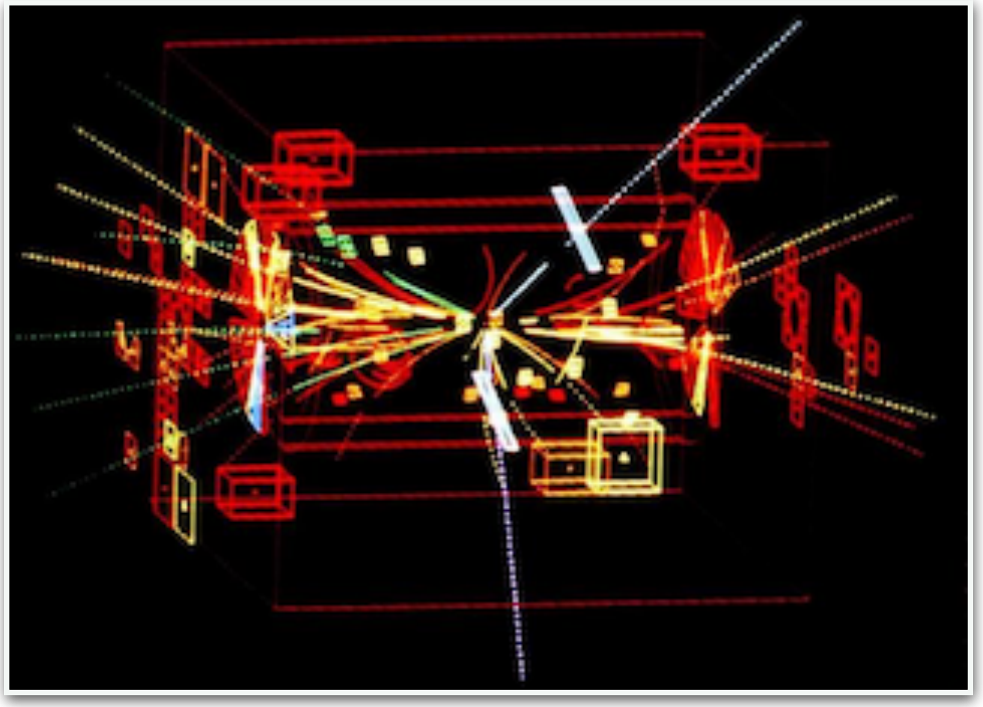
Step 2: take pictures



1 photo / event

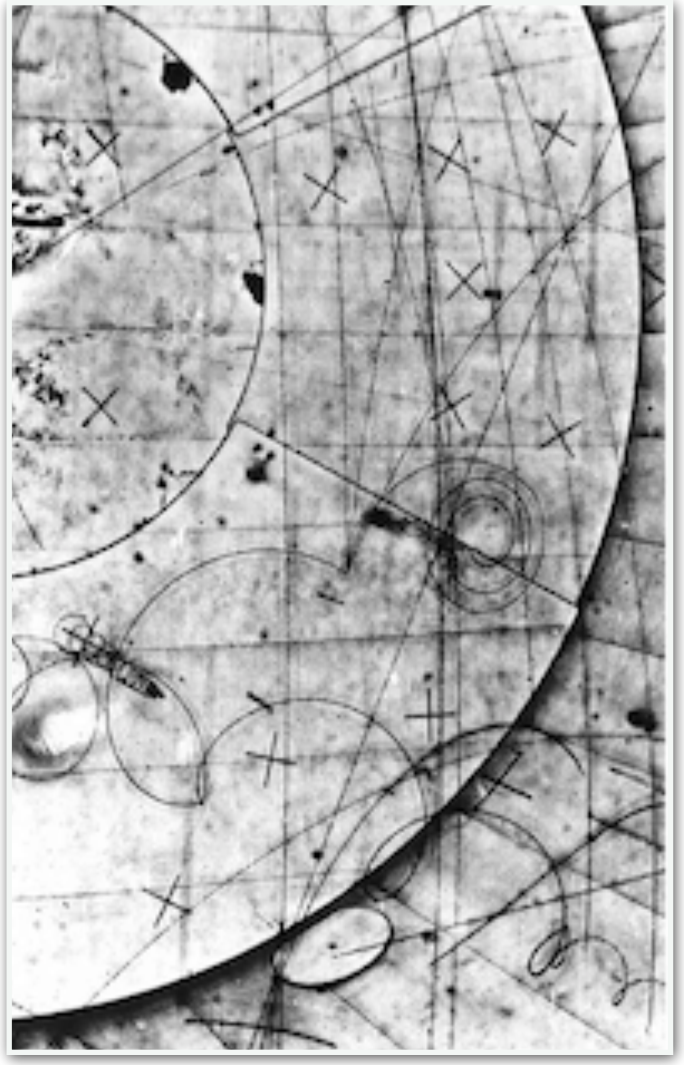
~6M events cds:1733654

Step 2: take pictures



~100kB / event

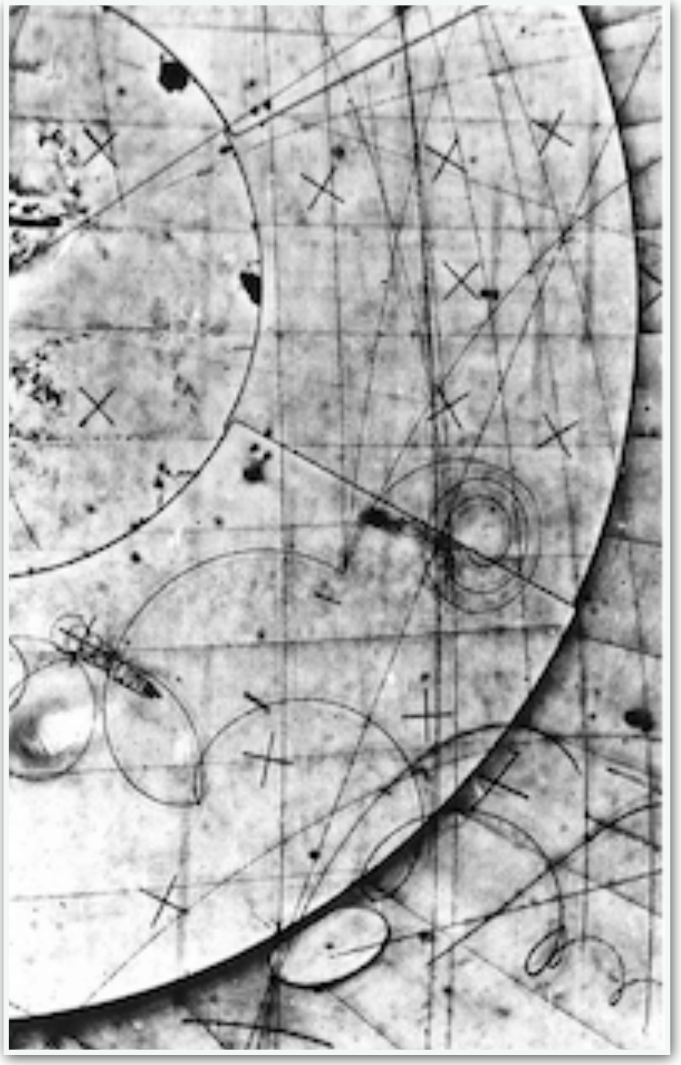
~10M events cds:182190



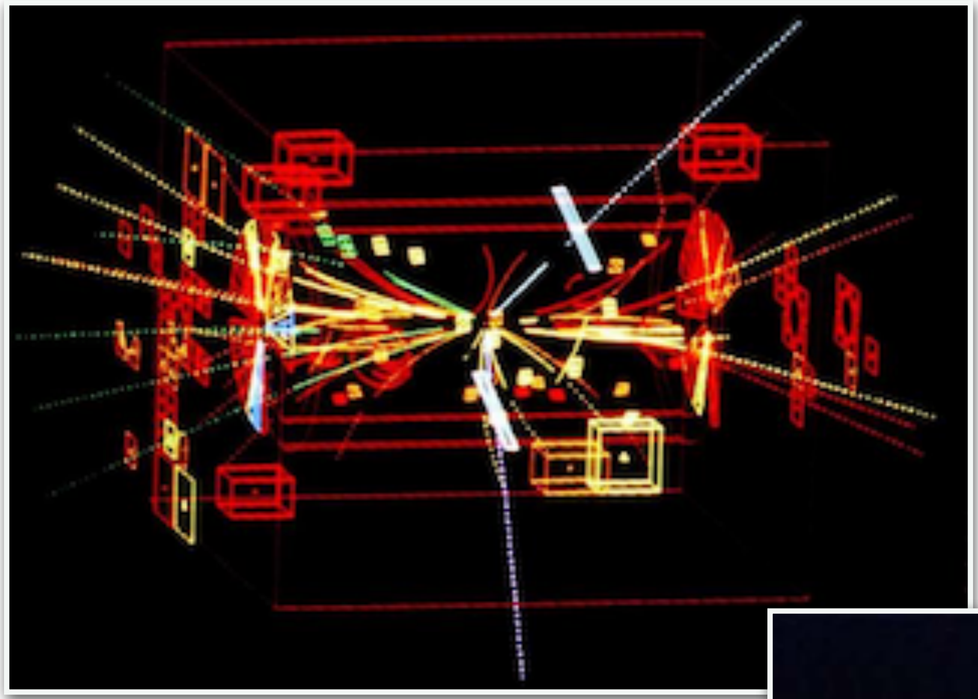
1 photo / event

~6M events cds:1733654

Step 2: take pictures

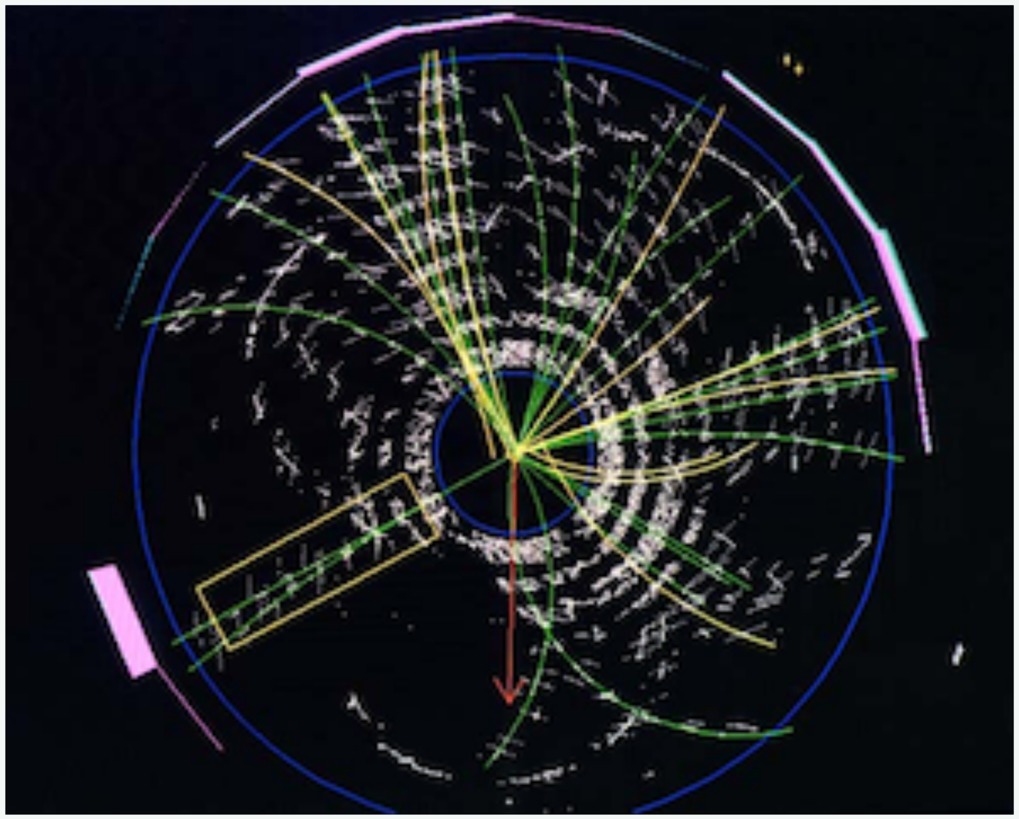


1 photo / event
~6M events cds:1733654



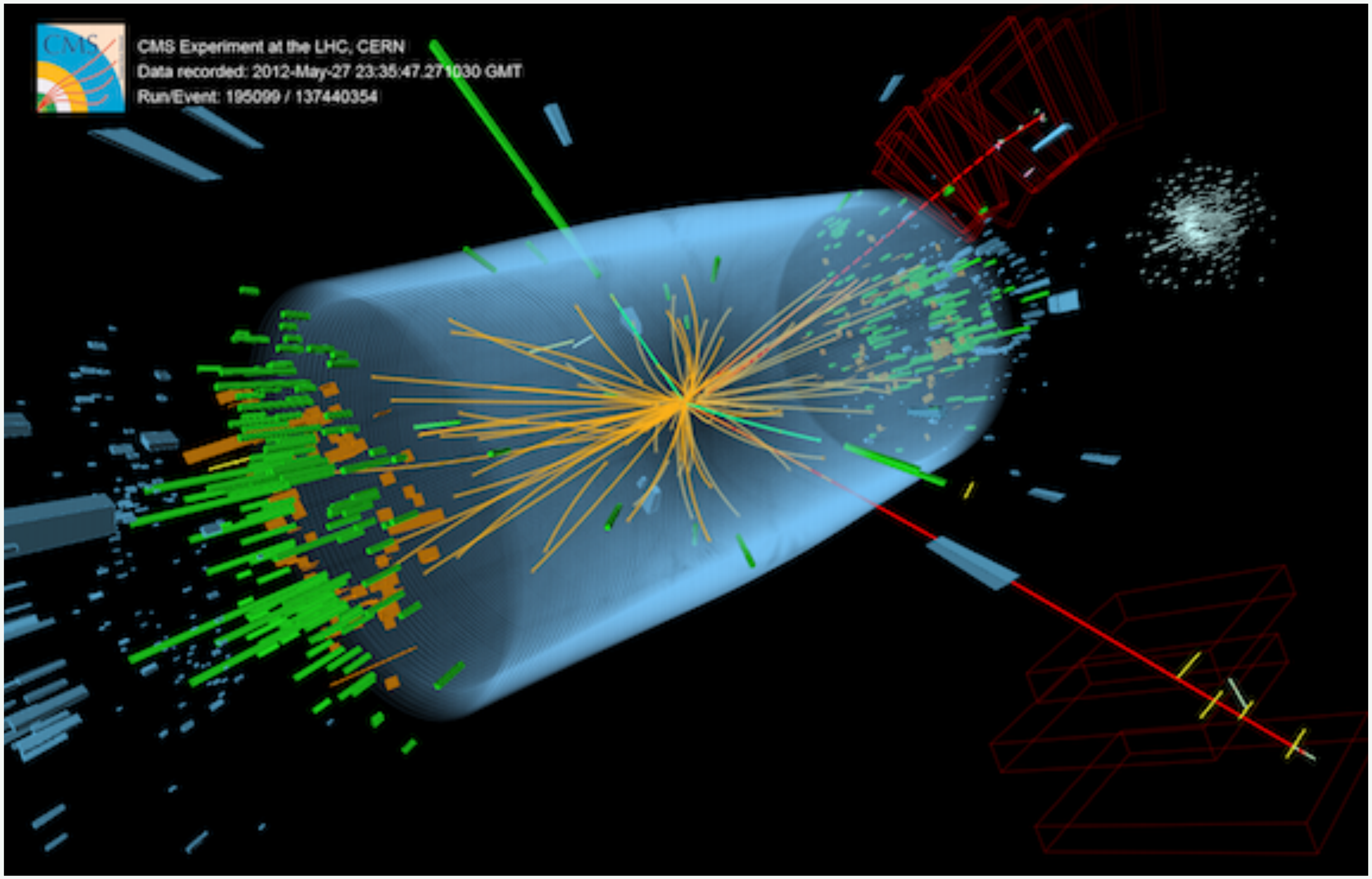
~100kB / event
~10M events cds:182190

~100kB / event
~1B events 10.1016/j.nima.2017.01.043

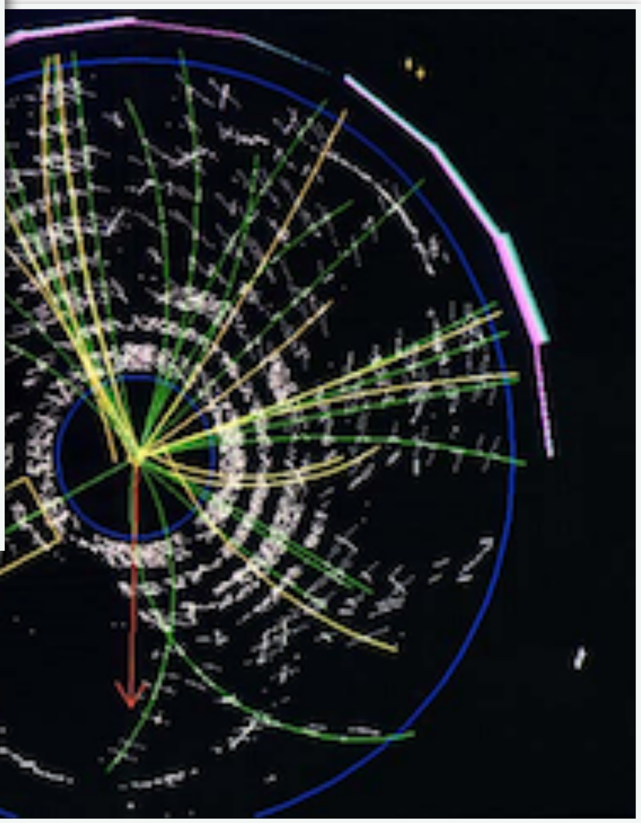


Step 2: take pictures

~1MB / event
~100B events


CMS Experiment at the LHC, CERN
Data recorded: 2012-May-27 23:35:47.271030 GMT
Run/Event: 195099 / 137440354

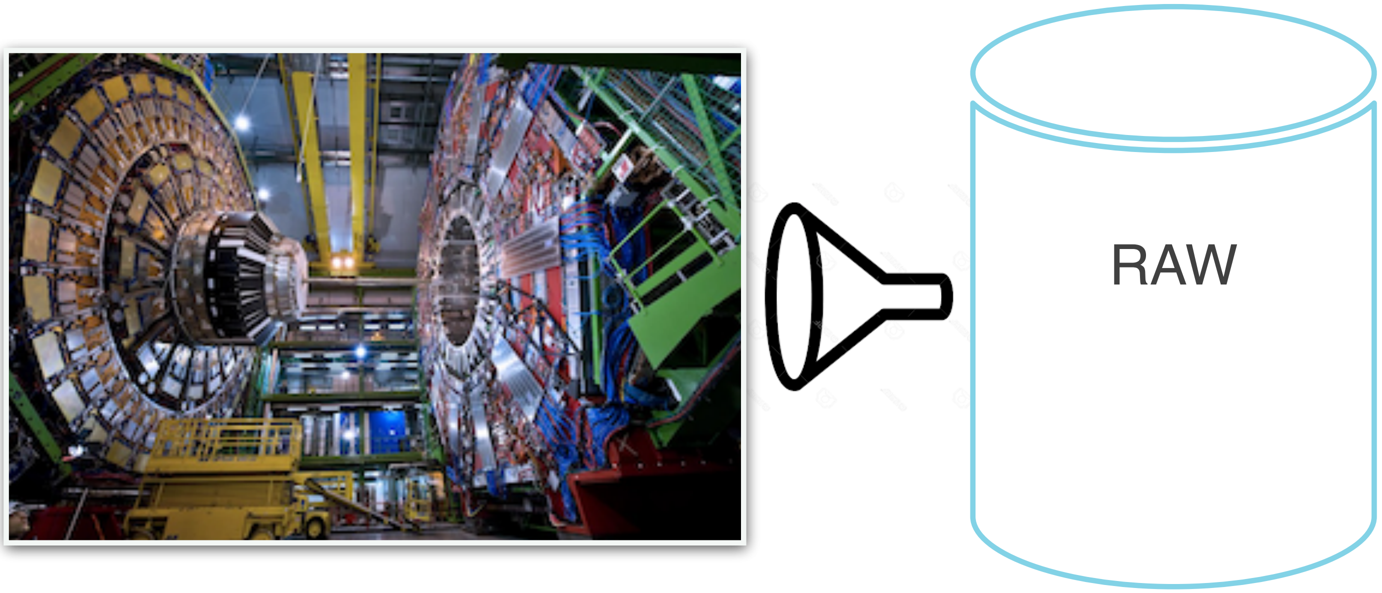
0kB / event
events 10.1016/j.nima.2017.01.043



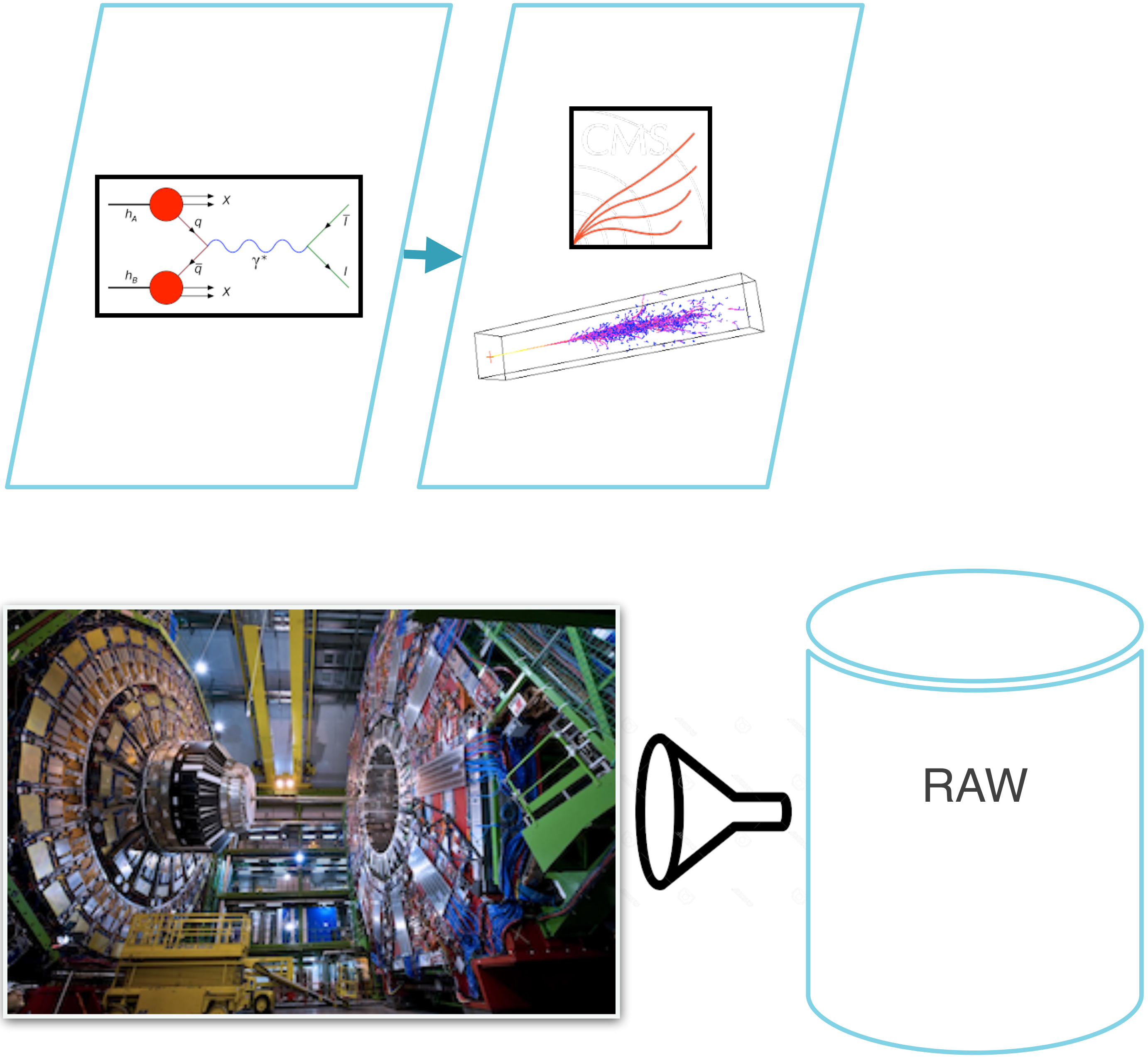
1 photo / event
~6M events cds:1733654

Our inference pipeline

Our inference pipeline

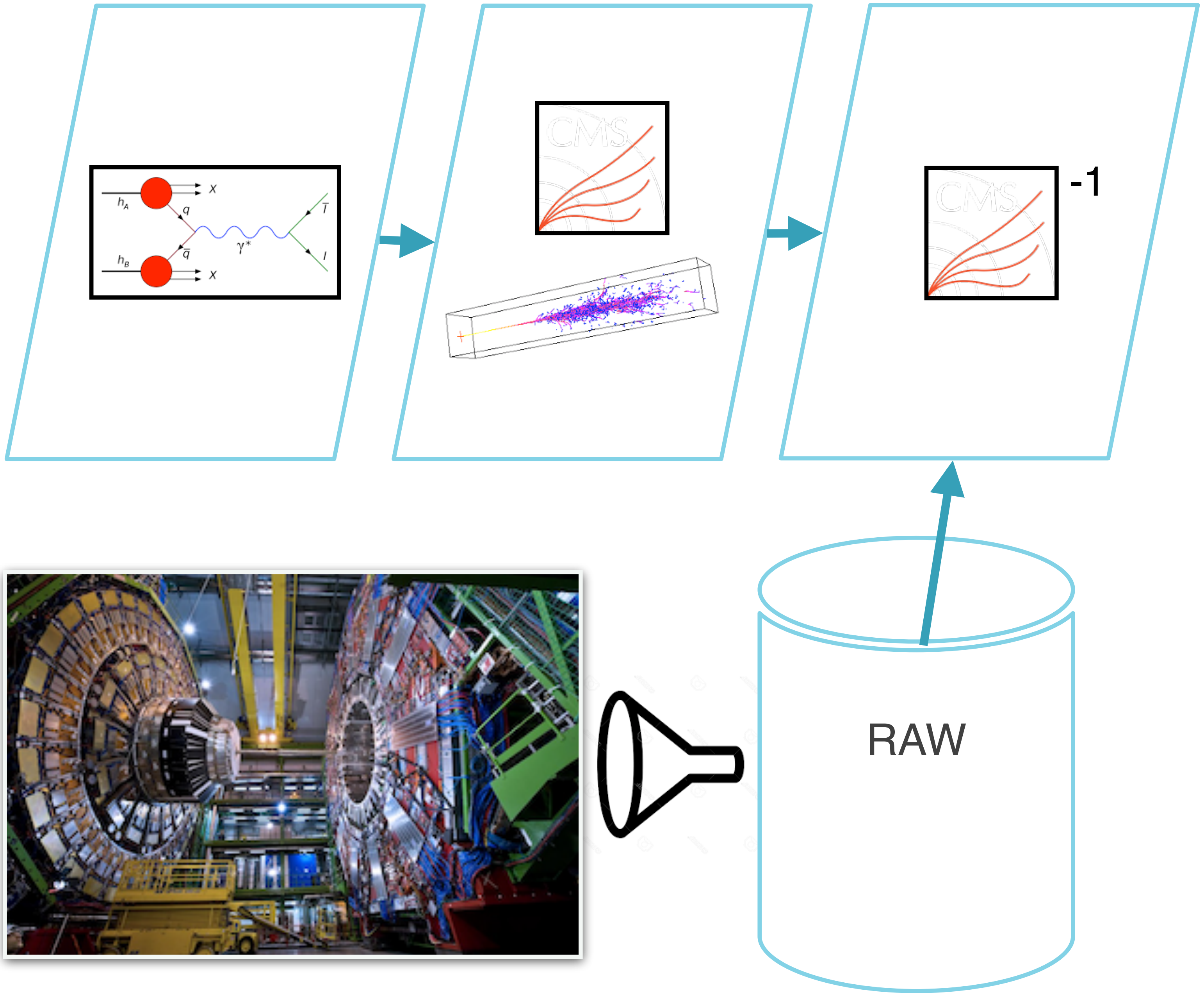


Our inference pipeline

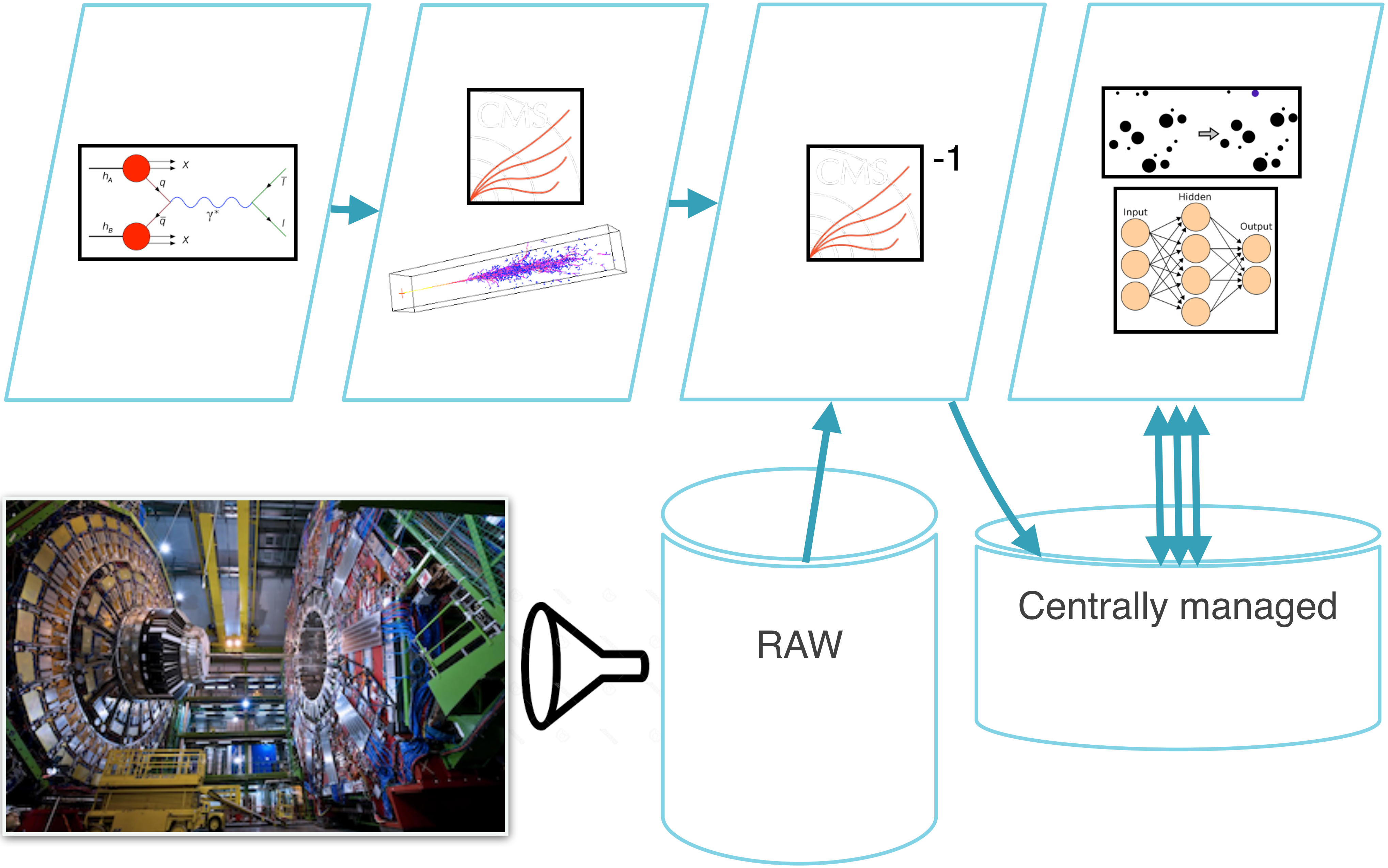


RAW

Our inference pipeline



Our inference pipeline

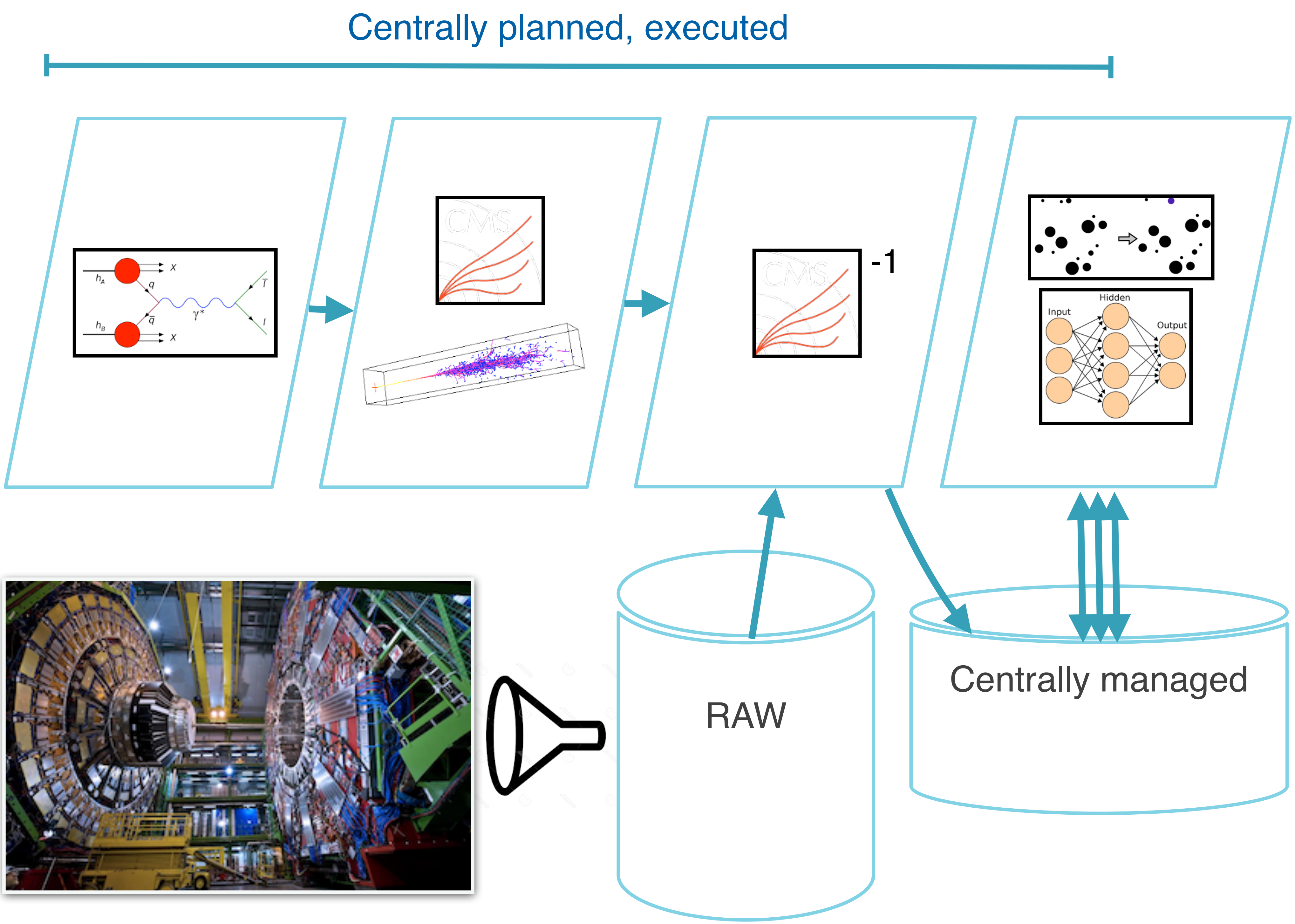


-1

RAW

Centrally managed

Our inference pipeline

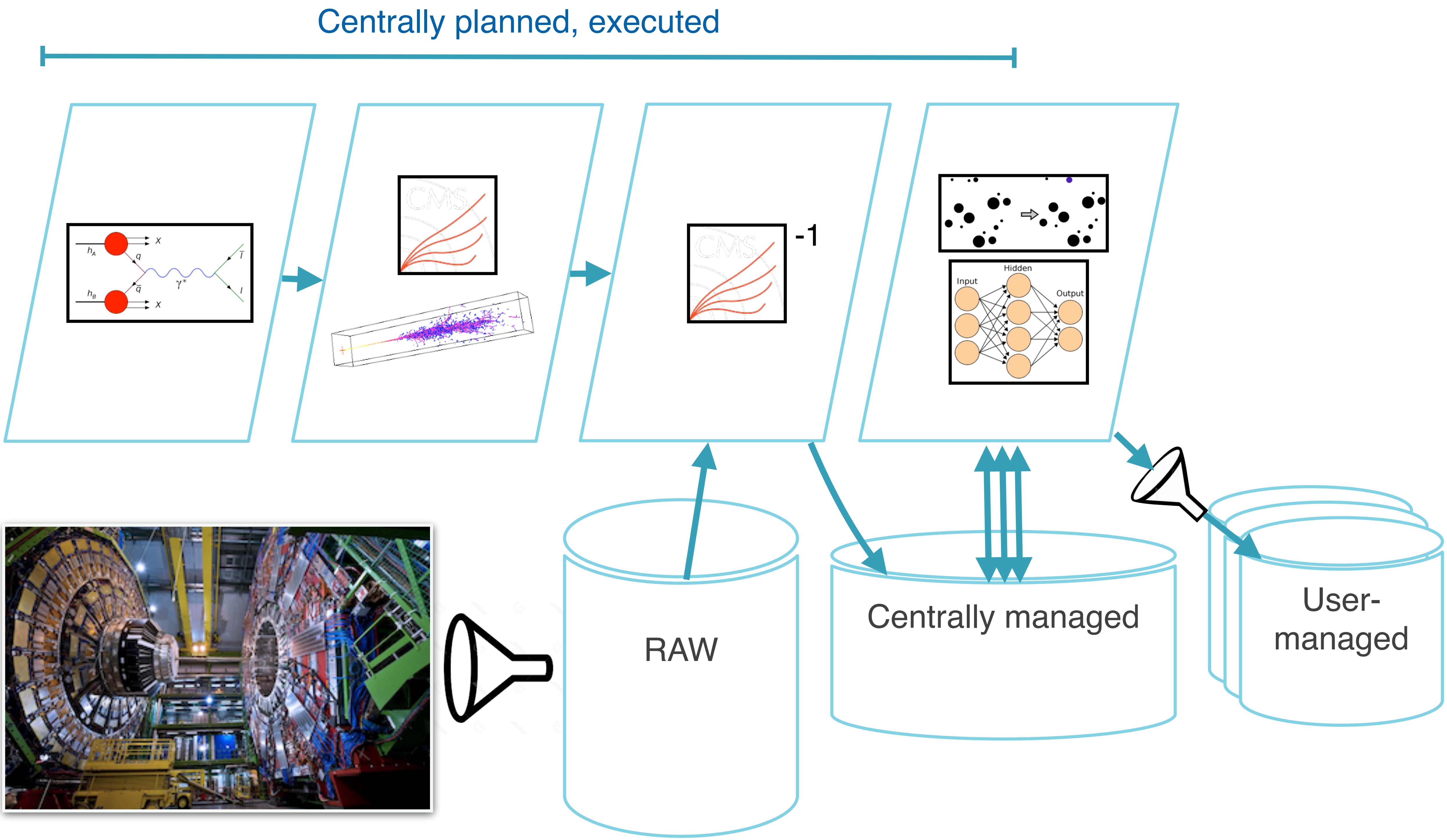


Centrally planned, executed

RAW

Centrally managed

Our inference pipeline



Centrally planned, executed

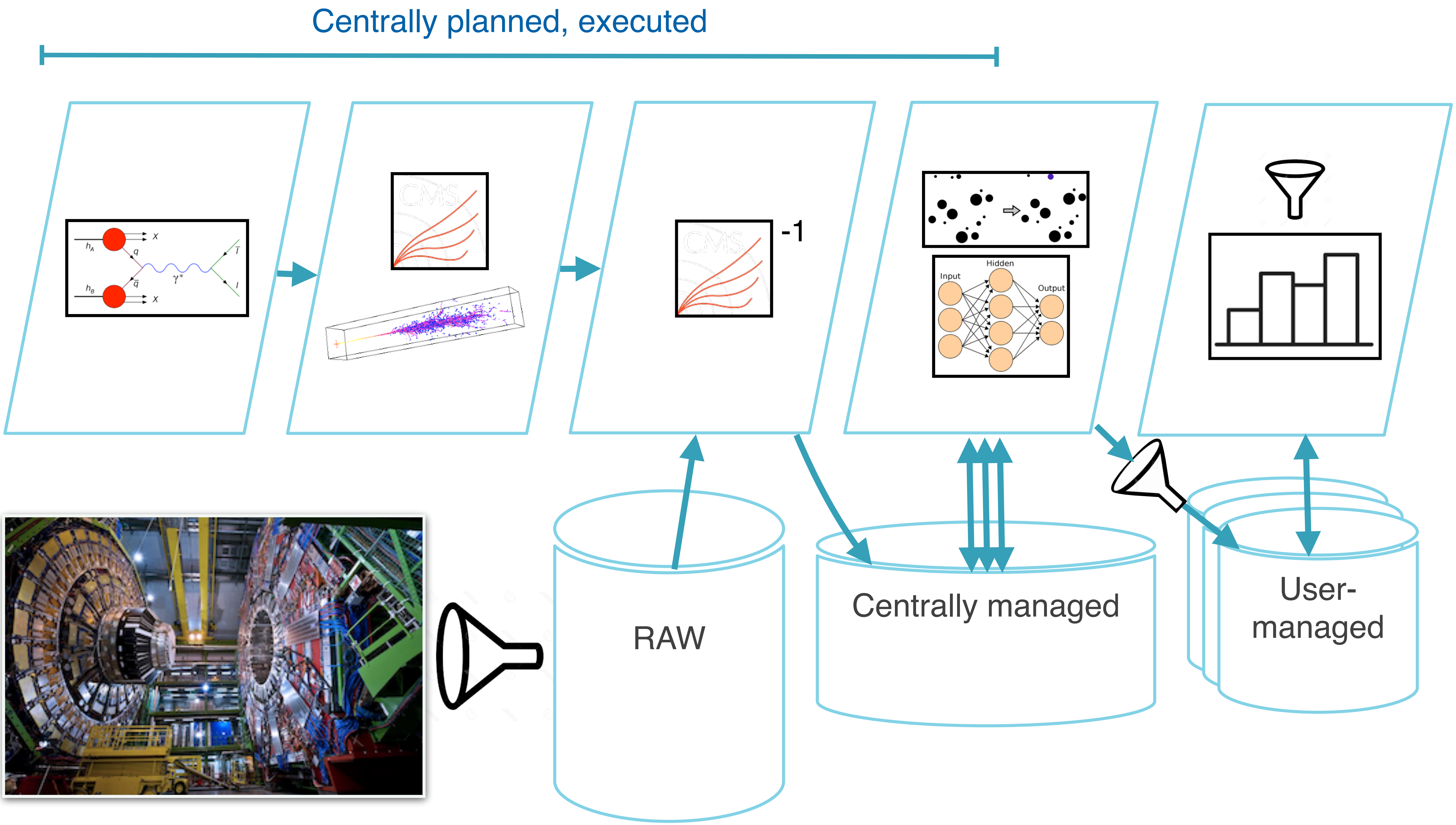
-1

RAW

Centrally managed

User-managed

Our inference pipeline



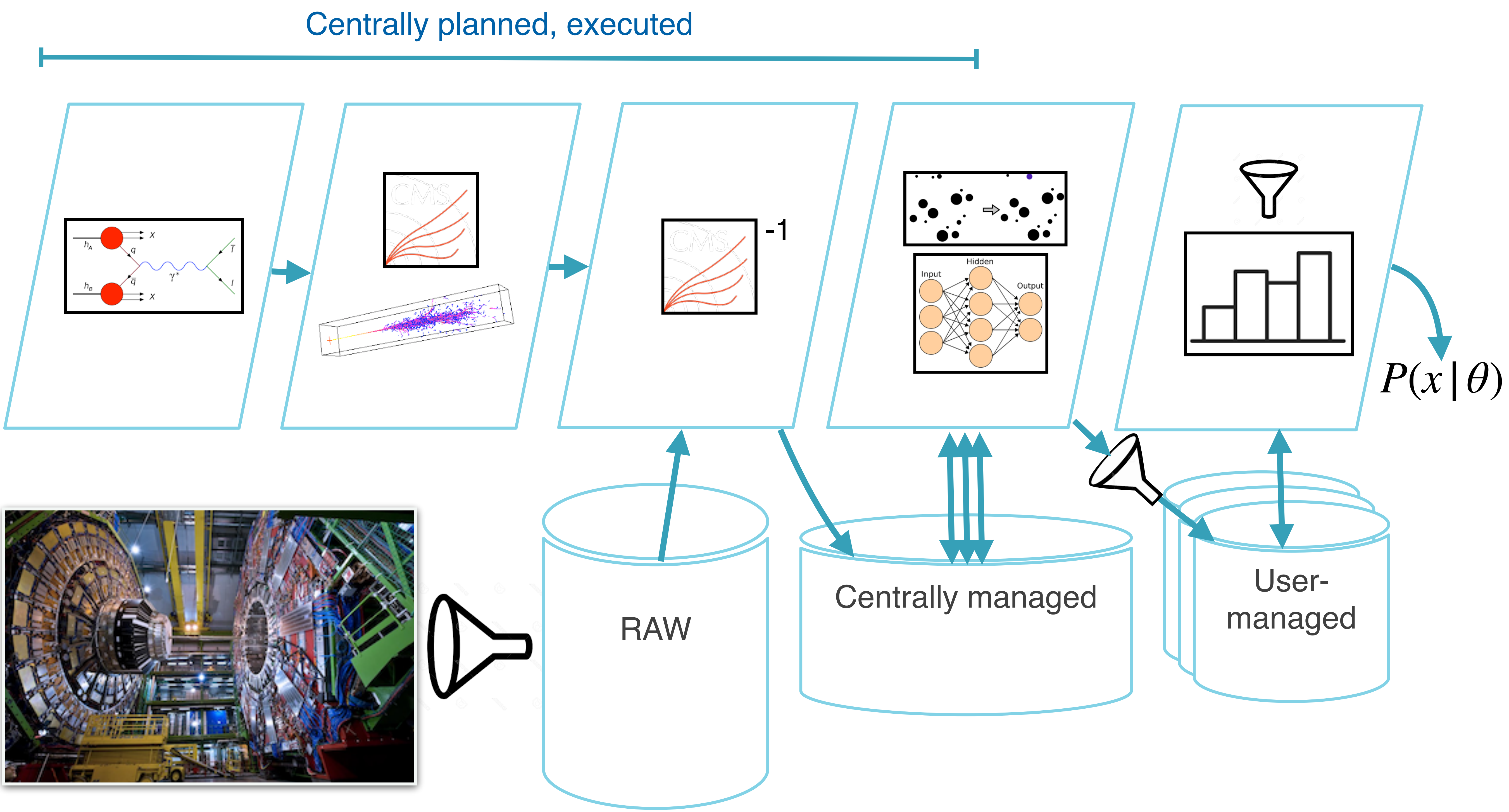
Centrally planned, executed

RAW

Centrally managed

User-managed

Our inference pipeline



Centrally planned, executed

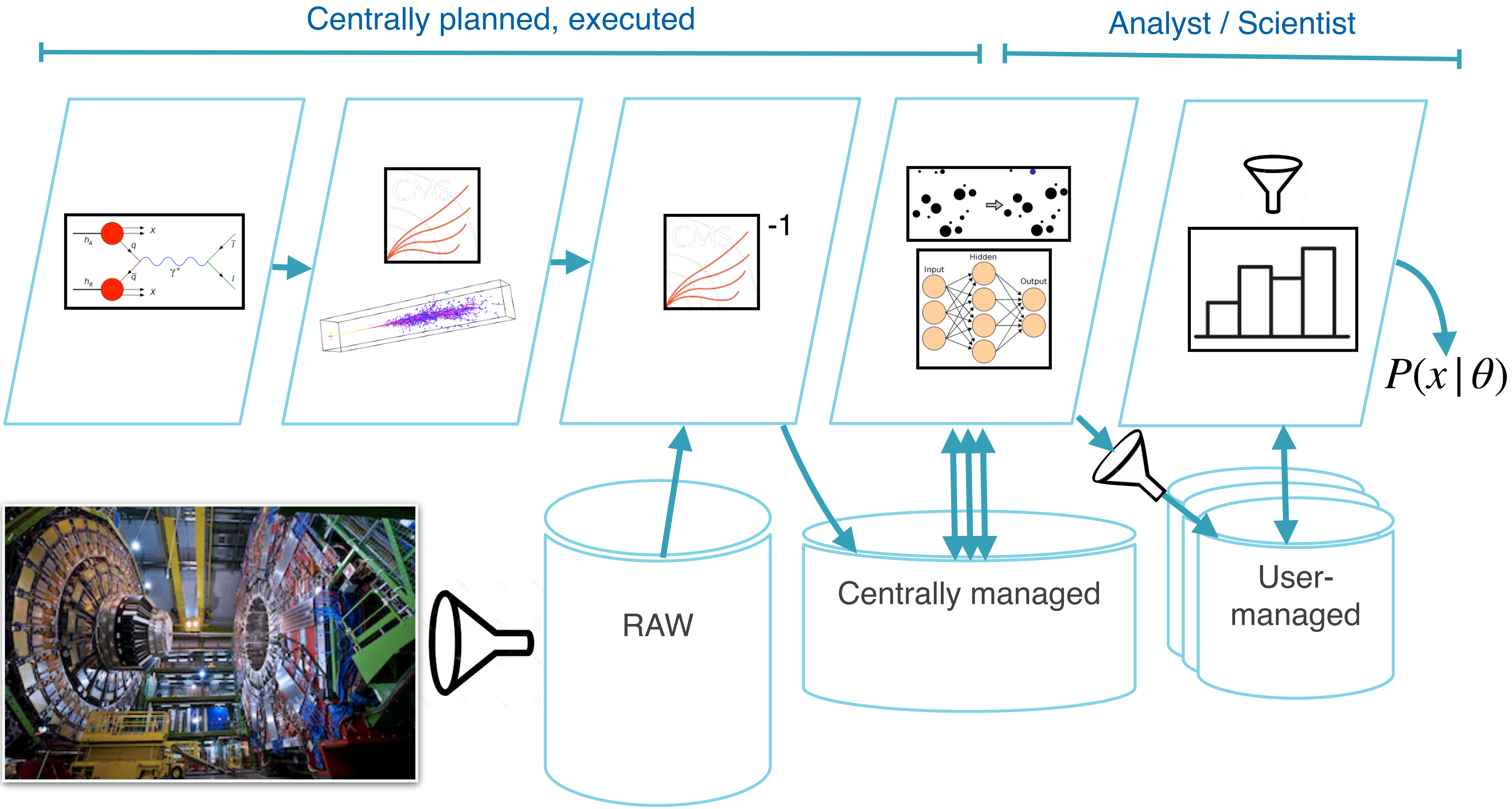
$P(x\,|\,\theta)$

RAW

Centrally managed

User-managed

Our inference pipeline



Centrally planned, executed

Analyst / Scientist

-1

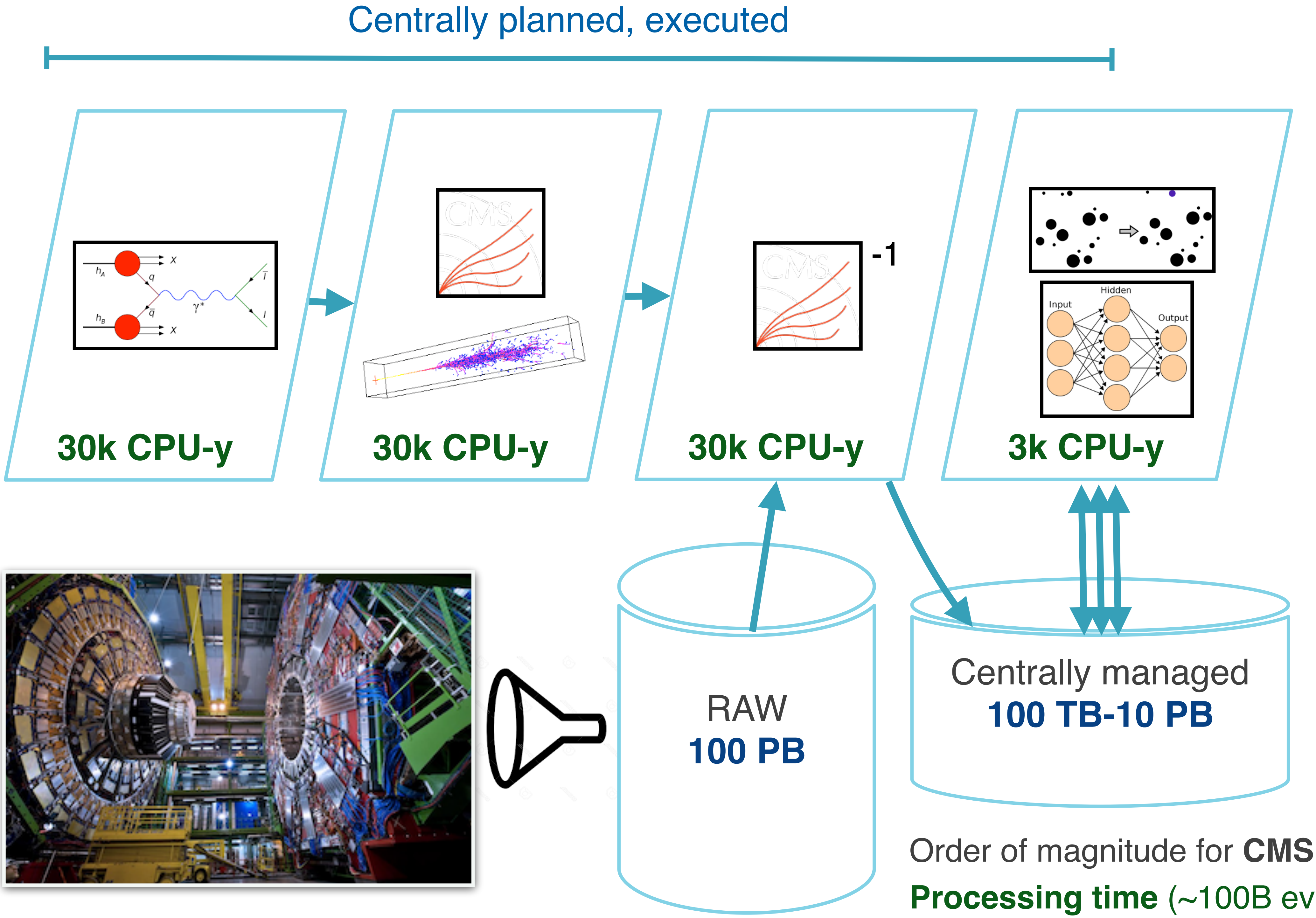
$P(x\,|\,\theta)$

RAW

Centrally managed

User-managed

Our inference pipeline



Centrally planned, executed

30k CPU-y

30k CPU-y

-1

30k CPU-y

3k CPU-y

RAW

100 PB

Centrally managed

100 TB-10 PB

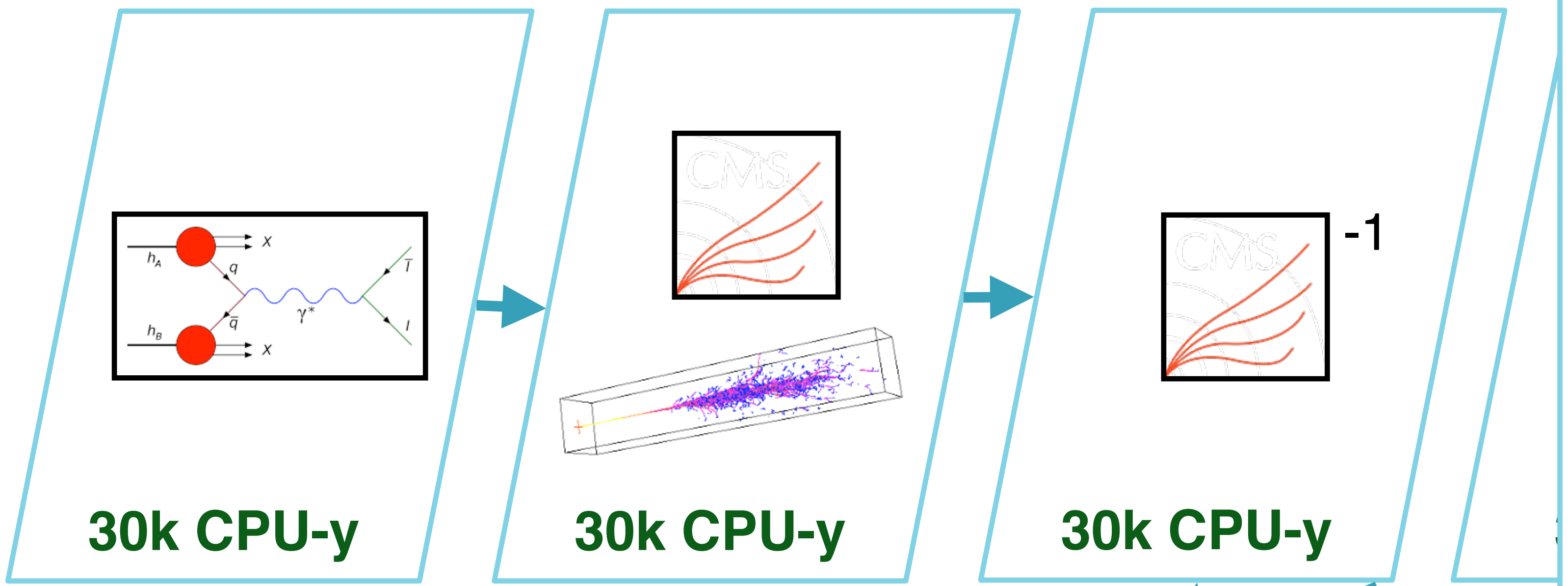
Order of magnitude for **CMS Run-II** (2016-18)

Processing time (~100B events)

Data volume on disk

Our inference pipeline

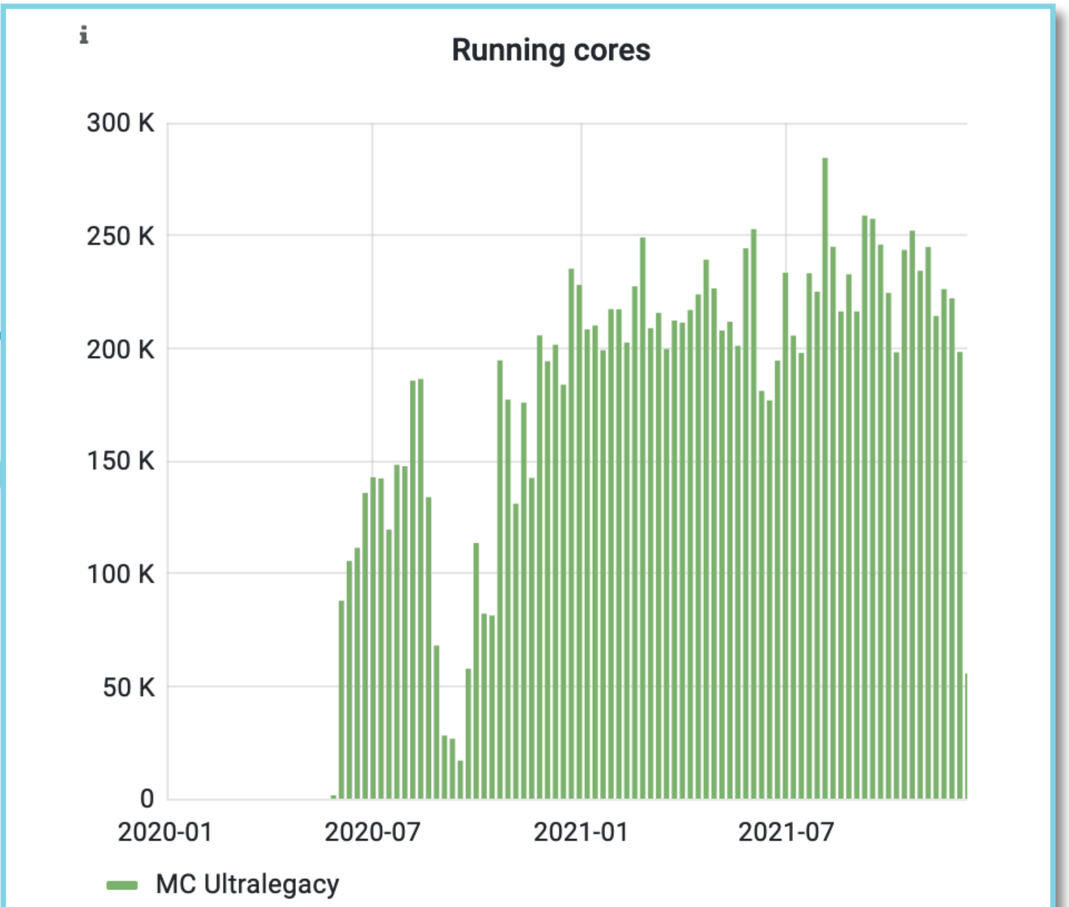
Centrally planned, executed



30k CPU-y

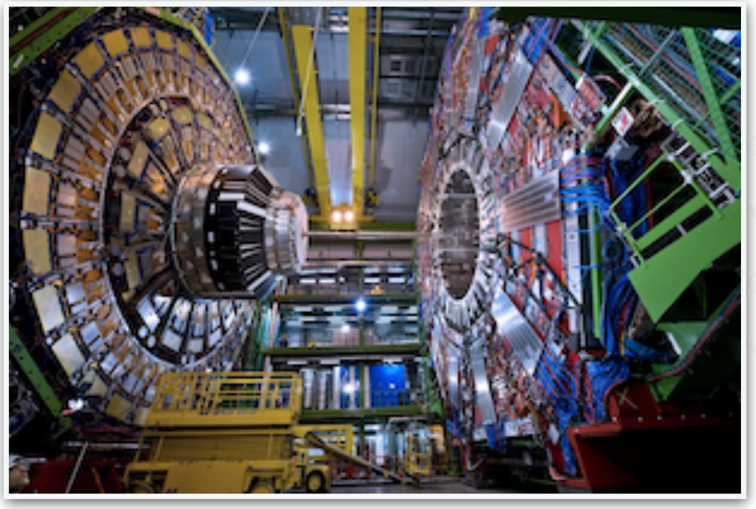
30k CPU-y

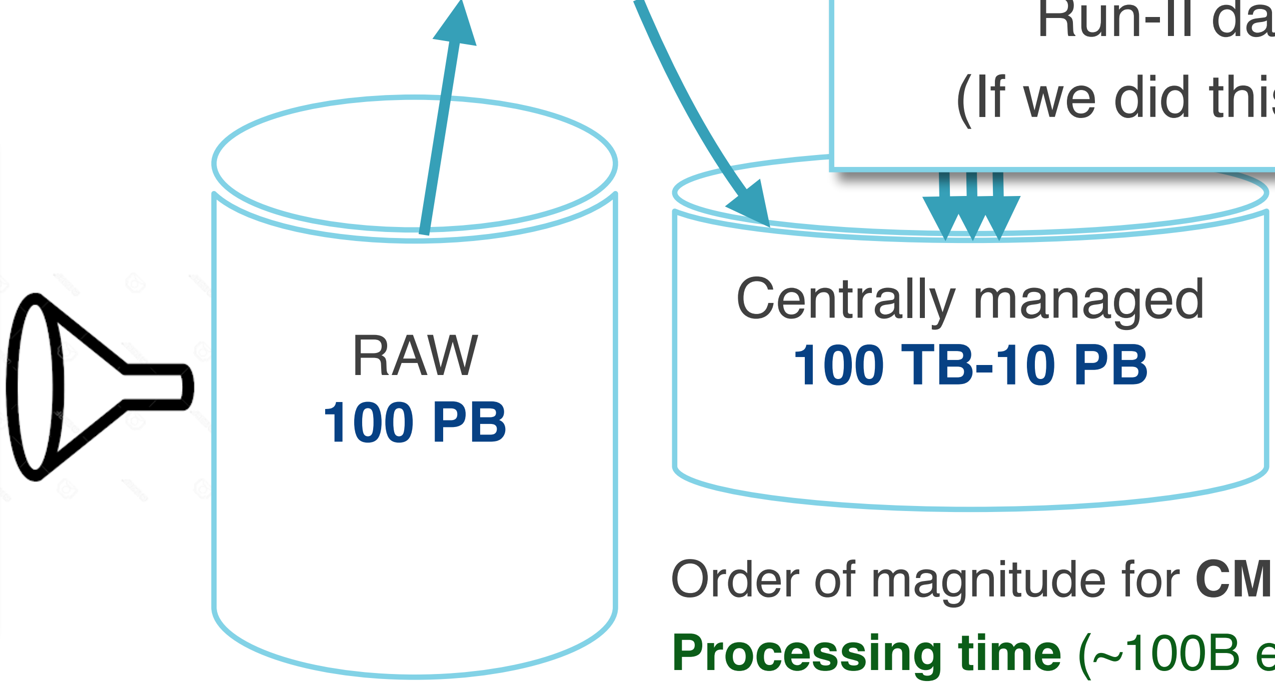
30k CPU-y



Sustained high throughput for Run-II data reprocessing

(If we did this on AWS: ~$40M)





RAW
100 PB

Centrally managed
100 TB-10 PB

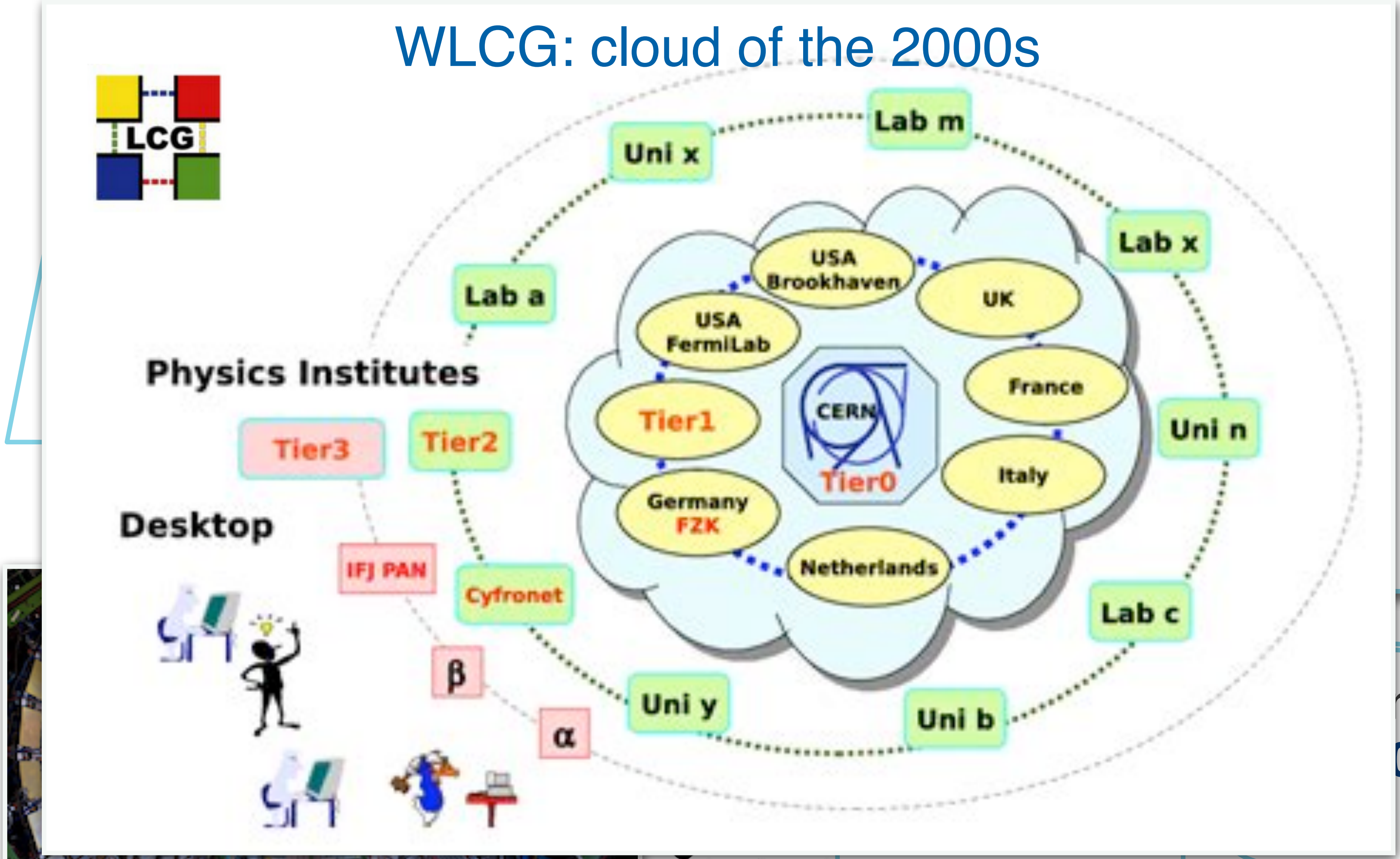
Order of magnitude for **CMS Run-II** (2016-18)

Processing time (~100B events)

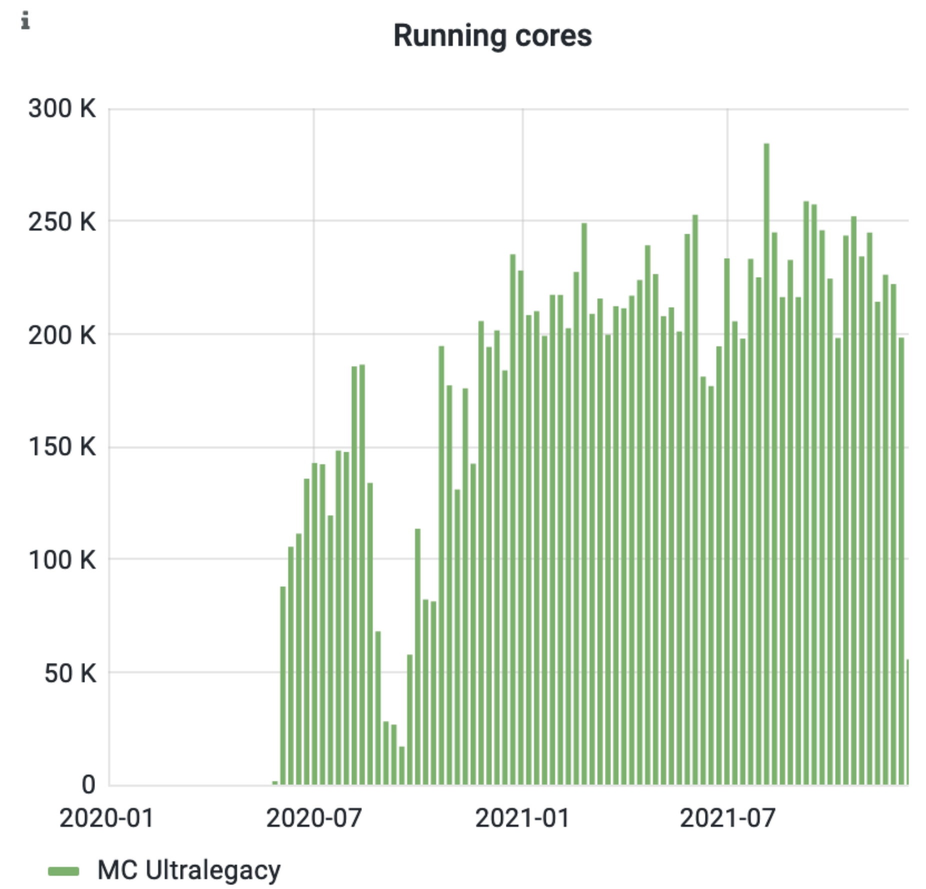
Data volume on disk

Our inference pipeline

Centrally planned, executed



WLCG: cloud of the 2000s



Sustained high throughput for Run-II data reprocessing

(If we did this on AWS: ~$40M)

rally managed

0 TB-10 PB

Order of magnitude for **CMS Run-II** (2016-18)

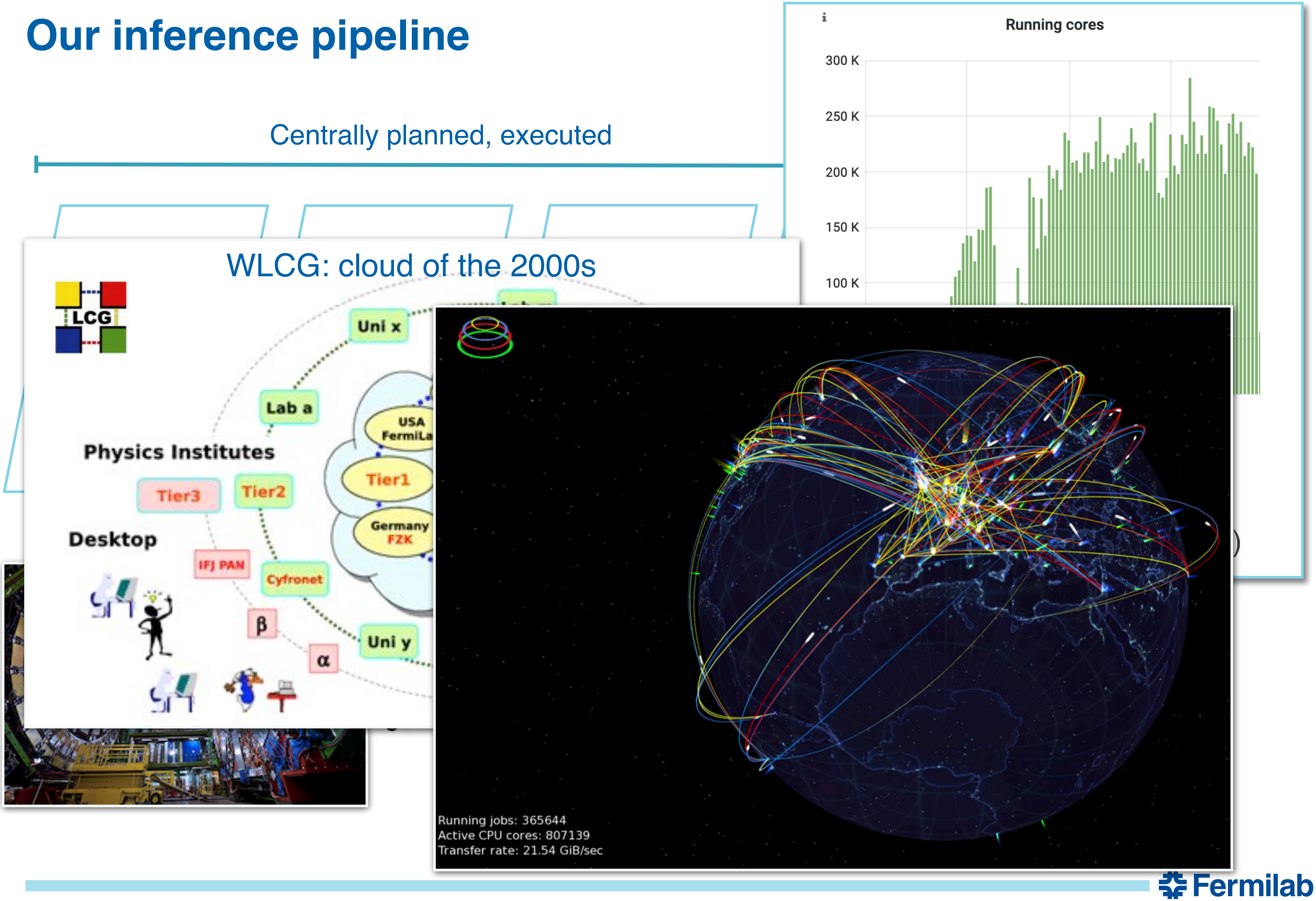
Processing time (~100B events)

Data volume on disk

Fermilab

Our inference pipeline

Centrally planned, executed



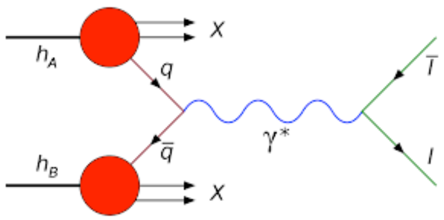
WLCG: cloud of the 2000s

Running jobs: 365644
Active CPU cores: 807139
Transfer rate: 21.54 GiB/sec

Centrally managed data

Primary dataset

Abstract, “what kind of events.”
e.g. hard scatter process for simulation, trigger filter for data



Data tiers

AOD

1e5/event

Data columns pertaining to
low-level reconstruction

.root

1e9/file

MiniAOD

1e4/event

Calibrated physics objects
Particle-flow candidates

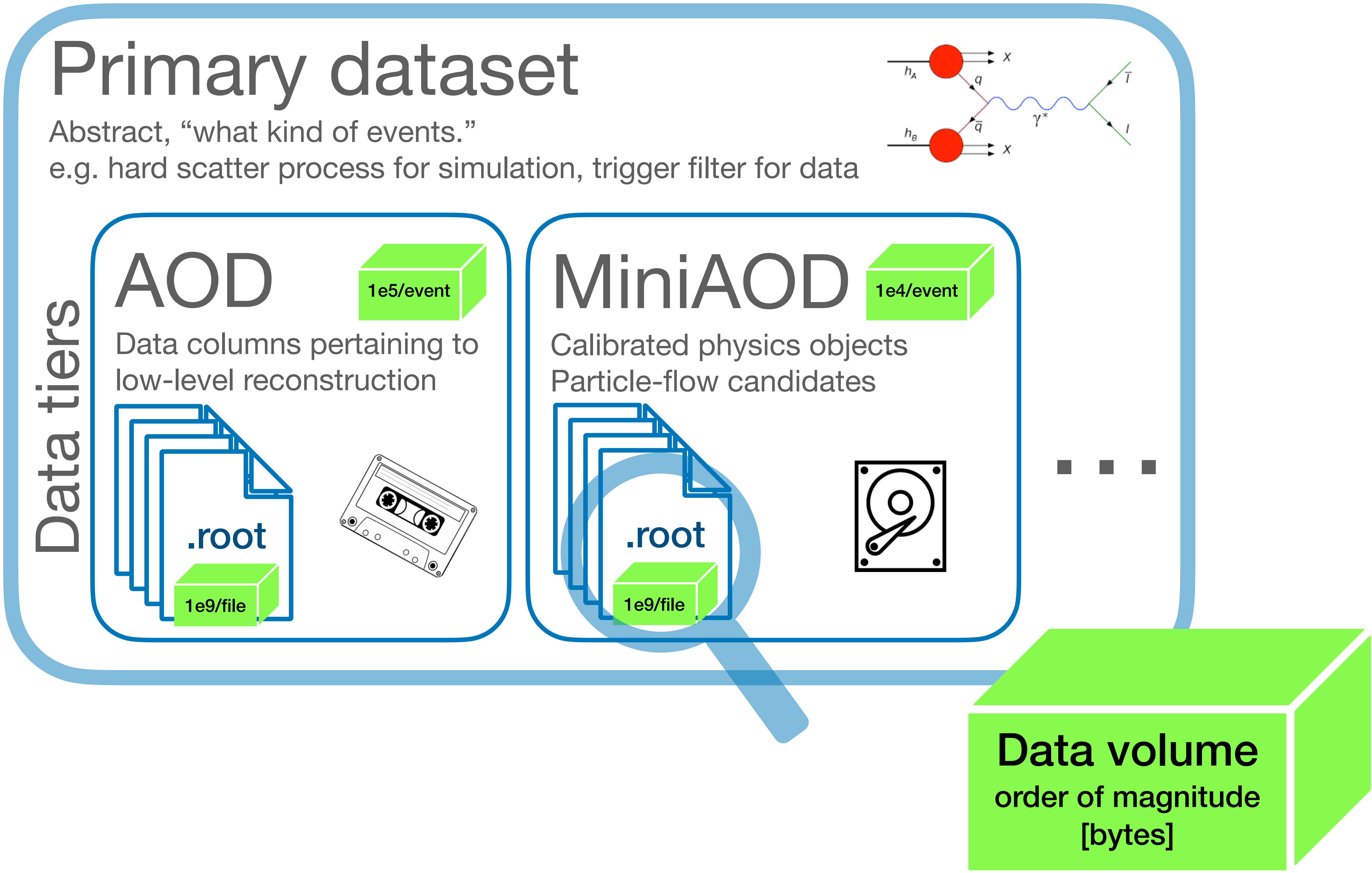
.root

1e9/file

. . .

Data volume
order of magnitude
[bytes]

Centrally managed data



Primary dataset

Abstract, “what kind of events.”
e.g. hard scatter process for simulation, trigger filter for data

Data tiers

AOD

1e5/event

Data columns pertaining to low-level reconstruction

.root

1e9/file

MiniAOD

1e4/event

Calibrated physics objects
Particle-flow candidates

.root

1e9/file

. . .

Data volume
order of magnitude
[bytes]

Centrally managed da

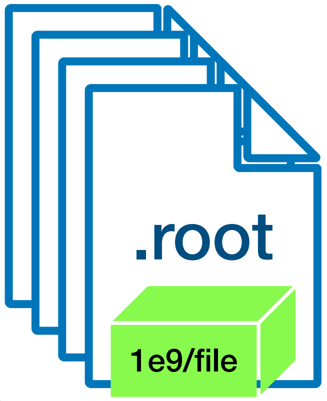
Primary d

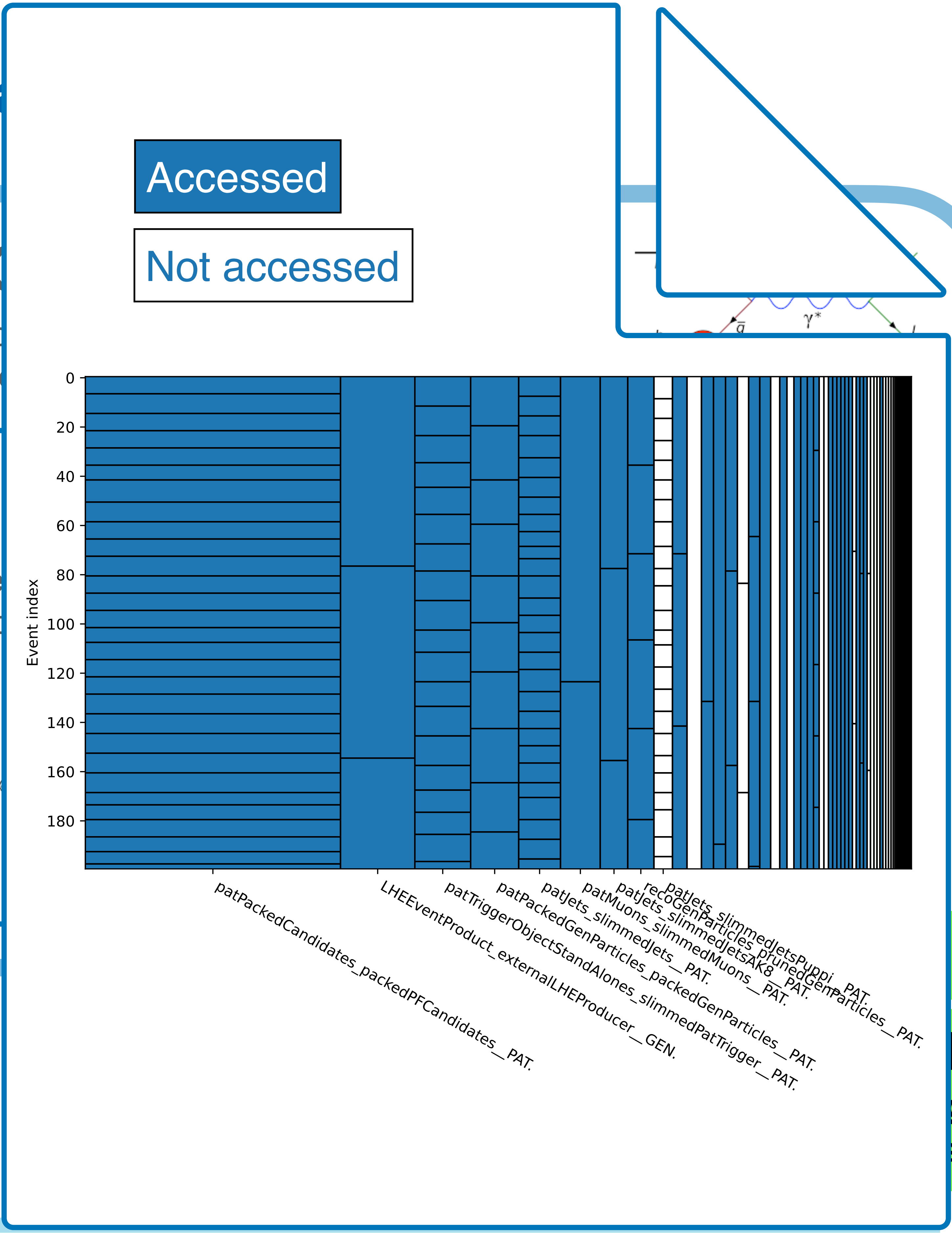
Abstract, “what kind o
e.g. hard scatter proc

Data tiers

AOD

Data columns pe
low-level reconst





lume
gnitude
s]

Centrally managed da

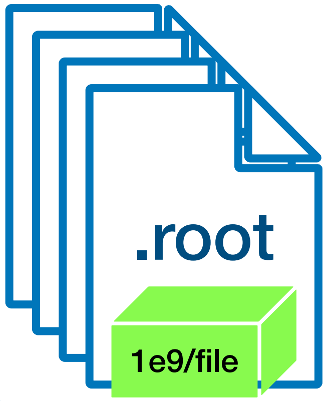
Primary d

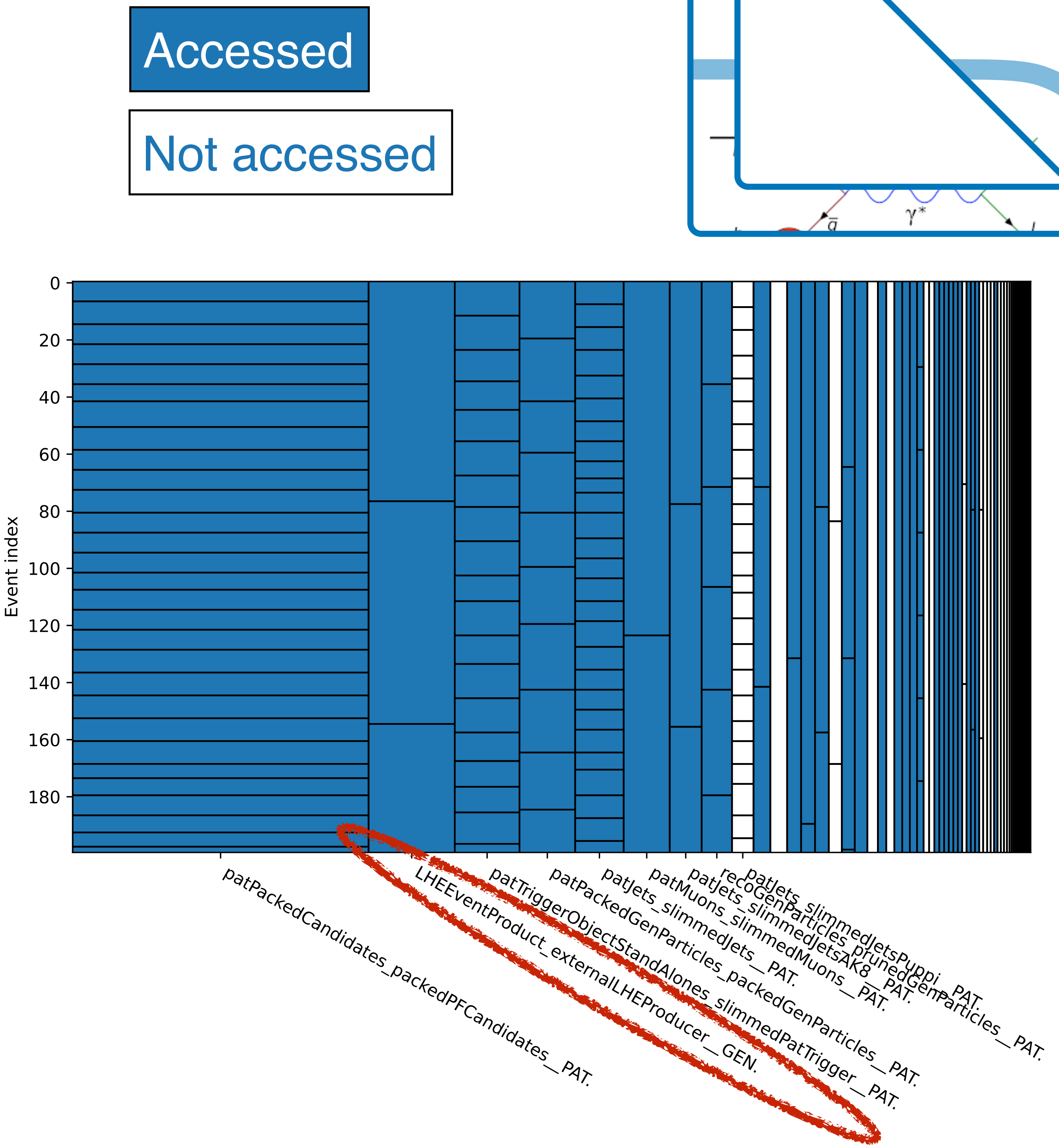
Abstract, “what kind o
e.g. hard scatter proc

Data tiers

AOD

Data columns pe
low-level reconst



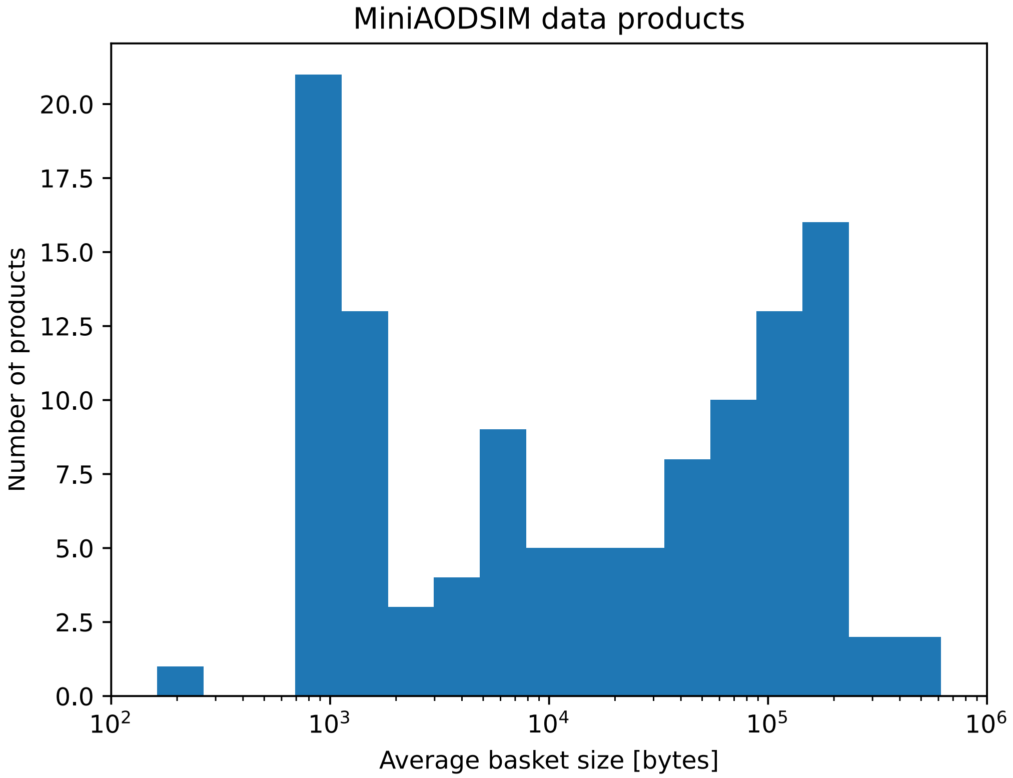


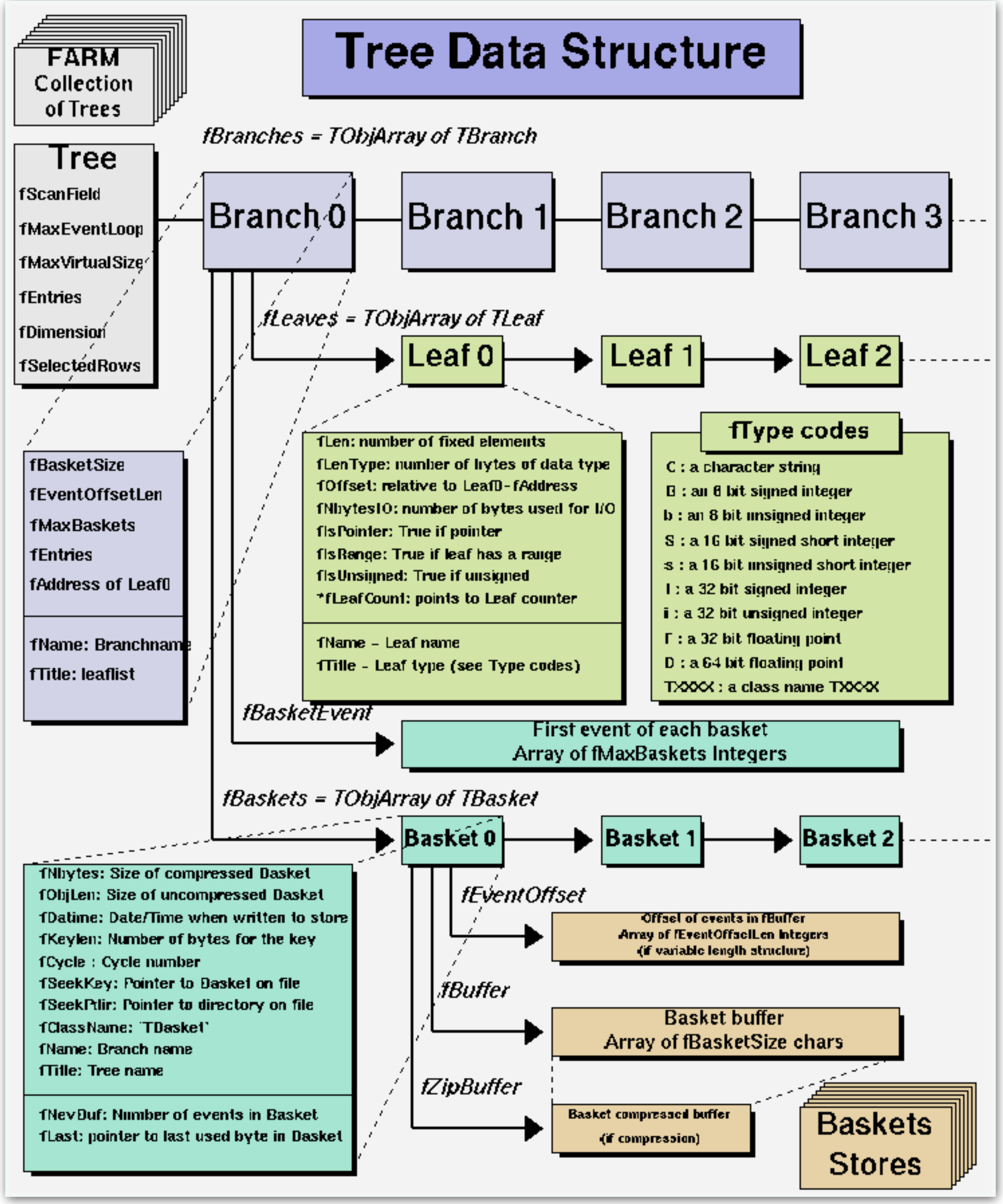
Copied forward each data tier

lume
gnitude
s]

File format

- Event Data Model (TTree)
- Branch: metadata about C++ data type, basket positions
- Basket: serialized C++ objects stored contiguously*





Tree Data Structure

FARM Collection of Trees

Tree
fScanField
fMaxEventLoop
fMaxVirtualSize
fEntries
fDimension
fSelectedRows

fBranches = TObjArray of TBranch

Branch 0 Branch 1 Branch 2 Branch 3

fBasketSize
fEventOffsetLen
fMaxBaskets
fEntries
fAddress of Leaf0

fName: Branchname
fTitle: leaflist

fLeaves = TObjArray of TLeaf

Leaf 0 Leaf 1 Leaf 2

fLen: number of fixed elements
fLenType: number of bytes of data type
fOffset: relative to Leaf0-fAddress
fNbytesIO: number of bytes used for I/O
fIsPointer: True if pointer
fIsRange: True if leaf has a range
fIsUnsigned: True if unsigned
*fLeafCount: points to Leaf counter

fName – Leaf name
fTitle – Leaf type (see Type codes)

fType codes
C : a character string
B : an 8 bit signed integer
b : an 8 bit unsigned integer
S : a 16 bit signed short integer
s : a 16 bit unsigned short integer
I : a 32 bit signed integer
i : a 32 bit unsigned integer
F : a 32 bit floating point
D : a 64 bit floating point
TXXXX : a class name TXXXX

fBasketEvent

First event of each basket
Array of fMaxBaskets Integers

fBaskets = TObjArray of TBasket

Basket 0 Basket 1 Basket 2

fNbytes: Size of compressed Basket
fObjLen: Size of uncompressed Basket
fDatime: Date/Time when written to store
fKeylen: Number of bytes for the key
fCycle : Cycle number
fSeekKey: Pointer to Basket on file
fSeekPdir: Pointer to directory on file
fClassName: 'TBasket'
fName: Branch name
fTitle: Tree name

fNevBuf: Number of events in Basket
fLast: pointer to last used byte in Basket

fEventOffset

Offset of events in fBuffer
Array of fEventOffsetLen Integers
(if variable length structure)

fBuffer

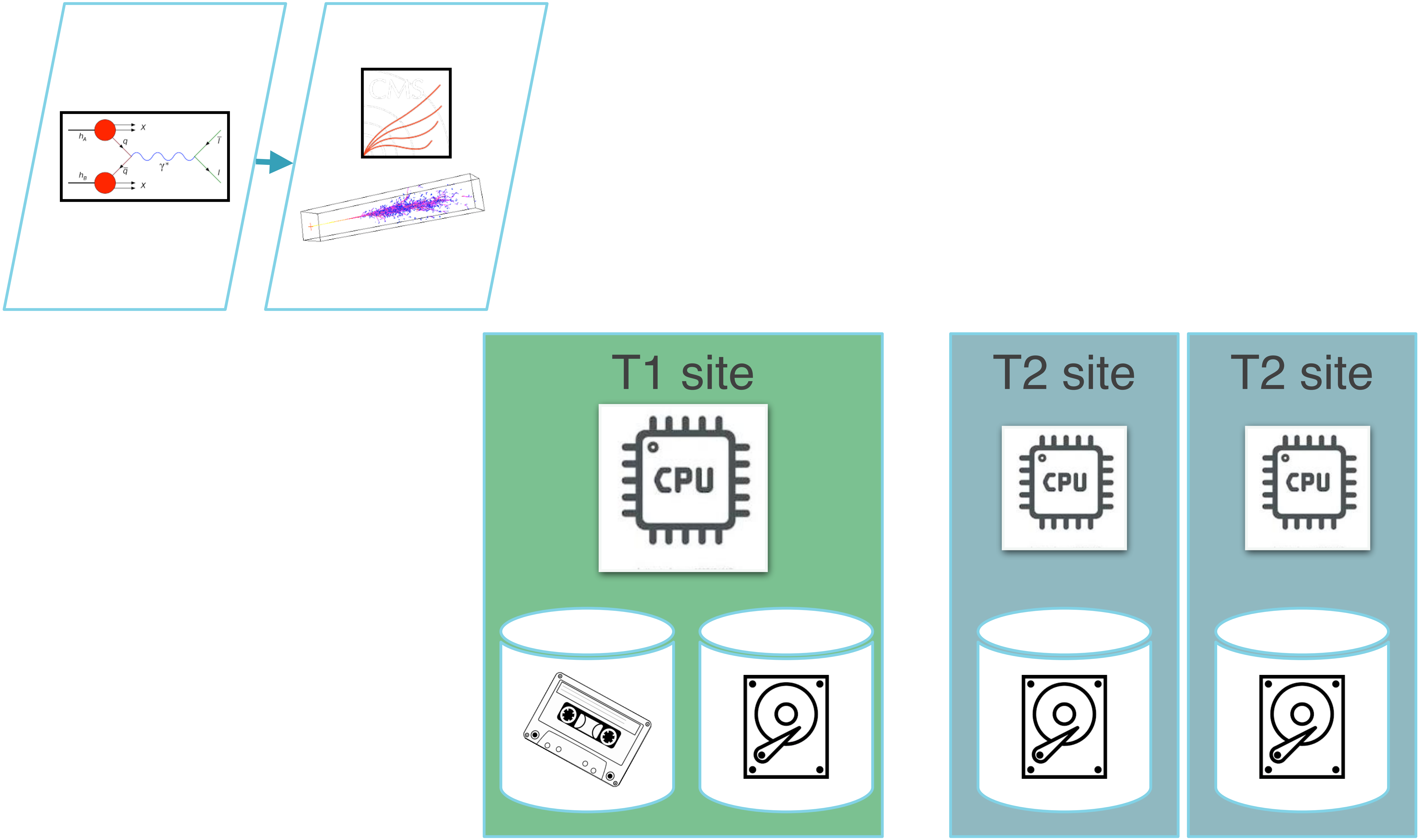
Basket buffer
Array of fBasketSize chars

fZipBuffer

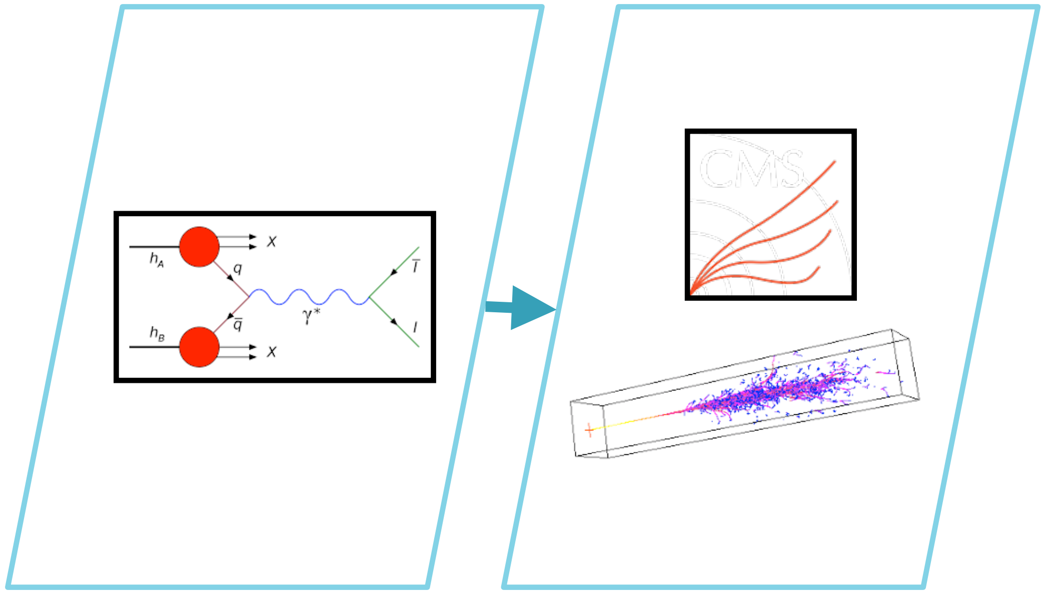
Basket compressed buffer
(if compression)

Baskets Stores

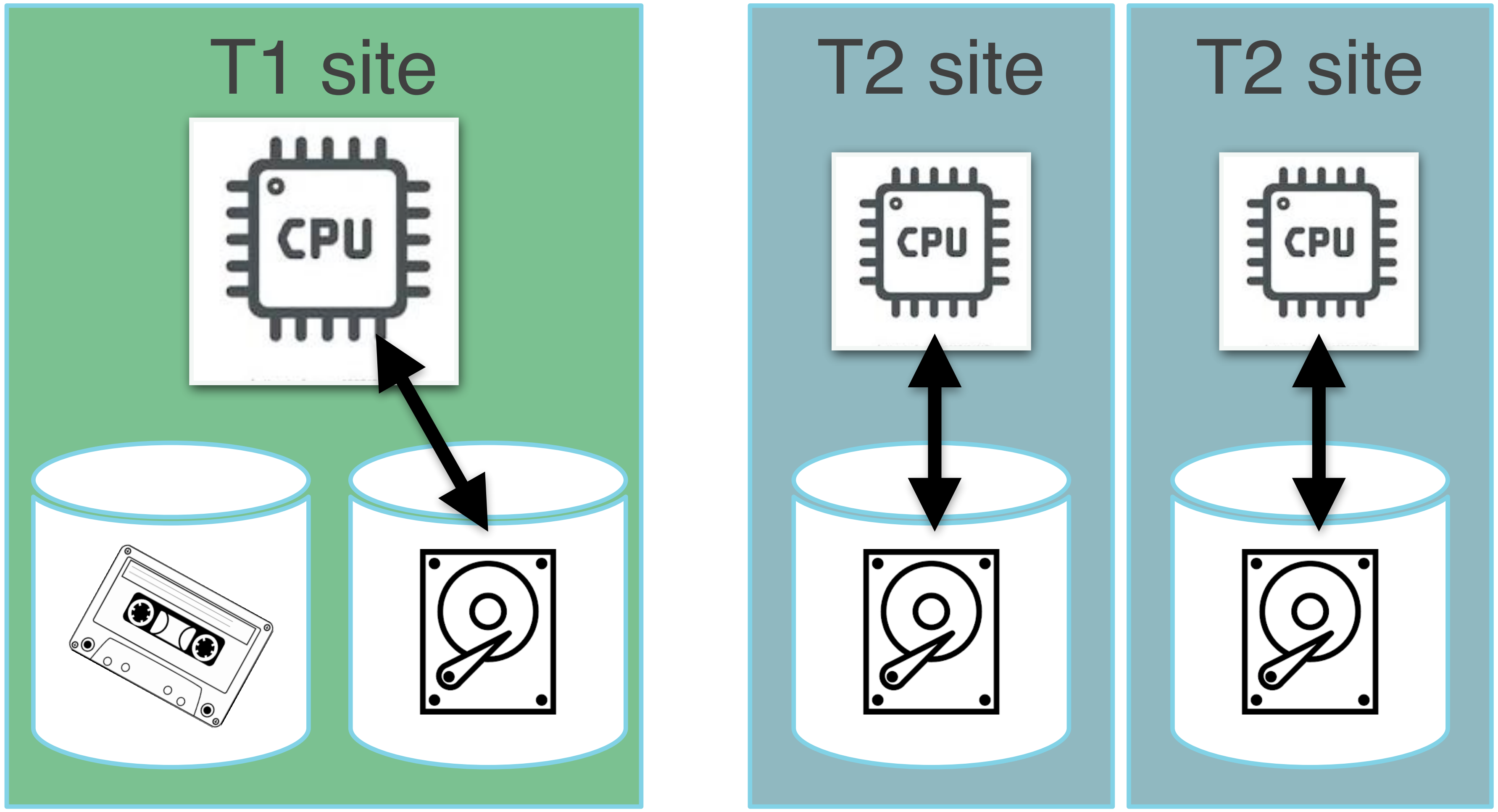
Simulation processing



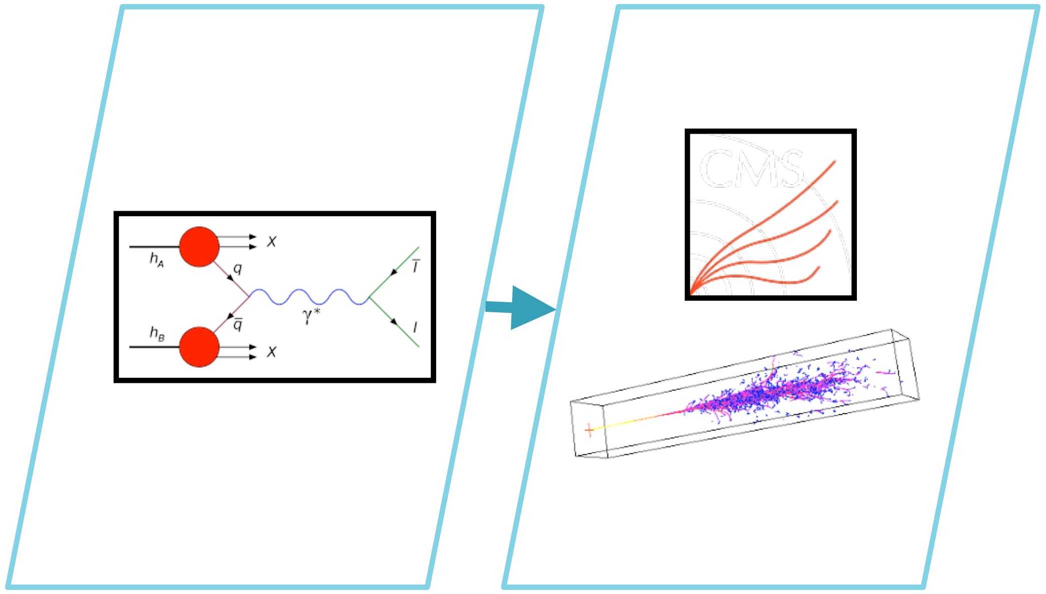
Simulation processing



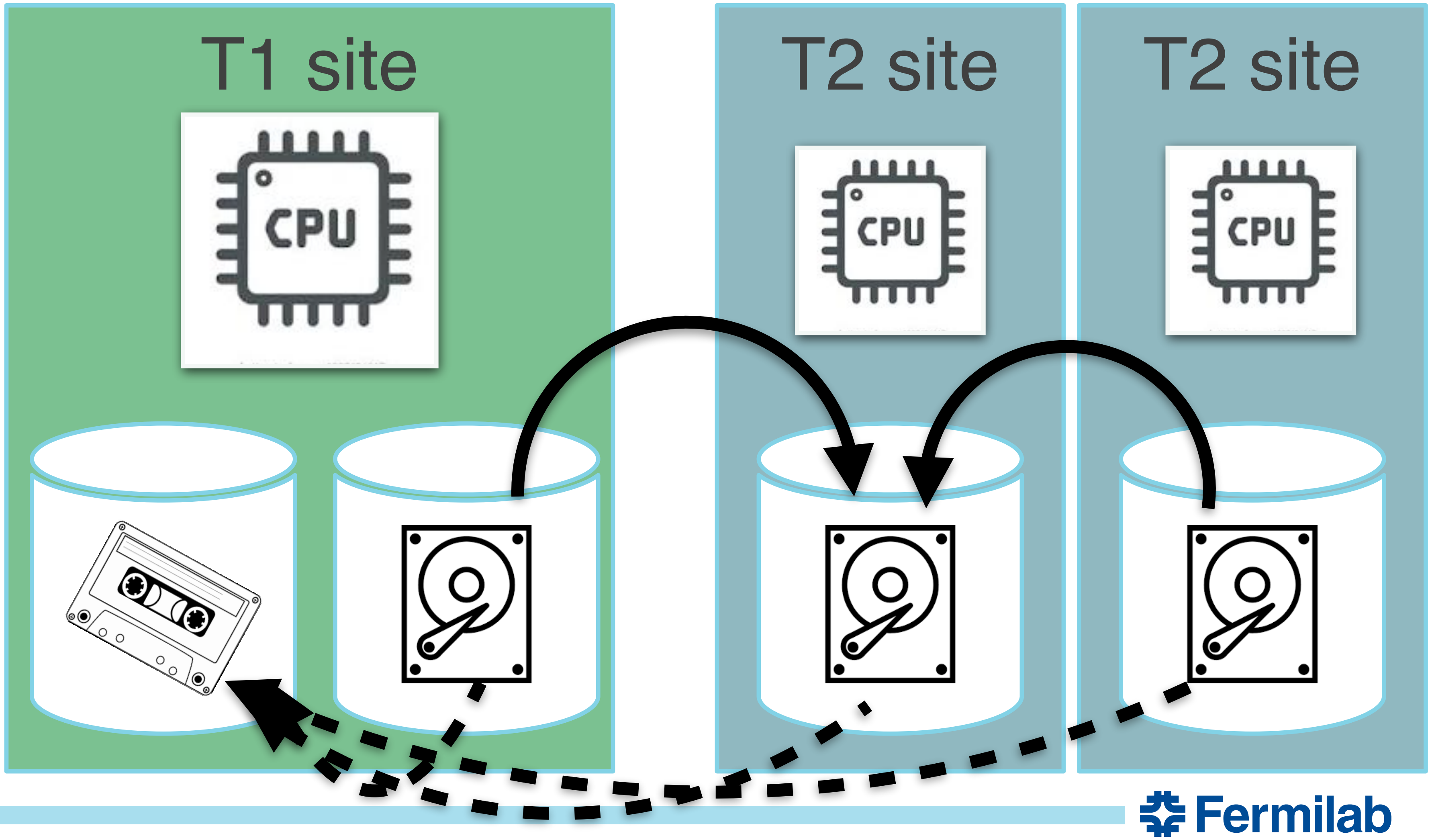
- Generate events at many sites



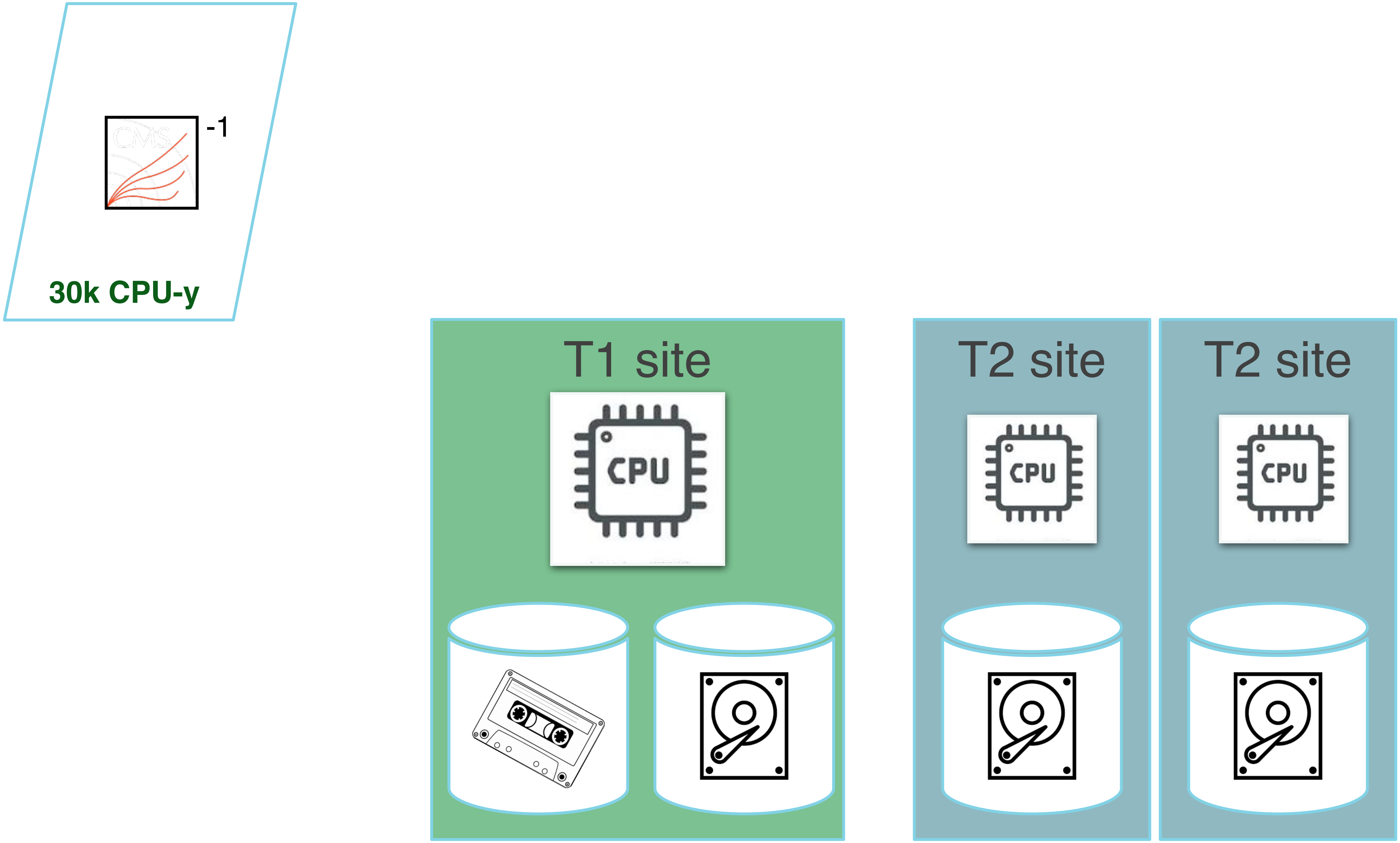
Simulation processing



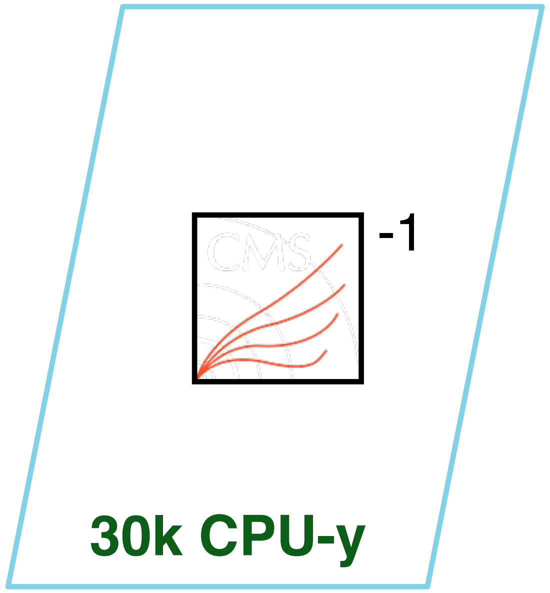
- Generate events at many sites
- Collect output on disk & tape



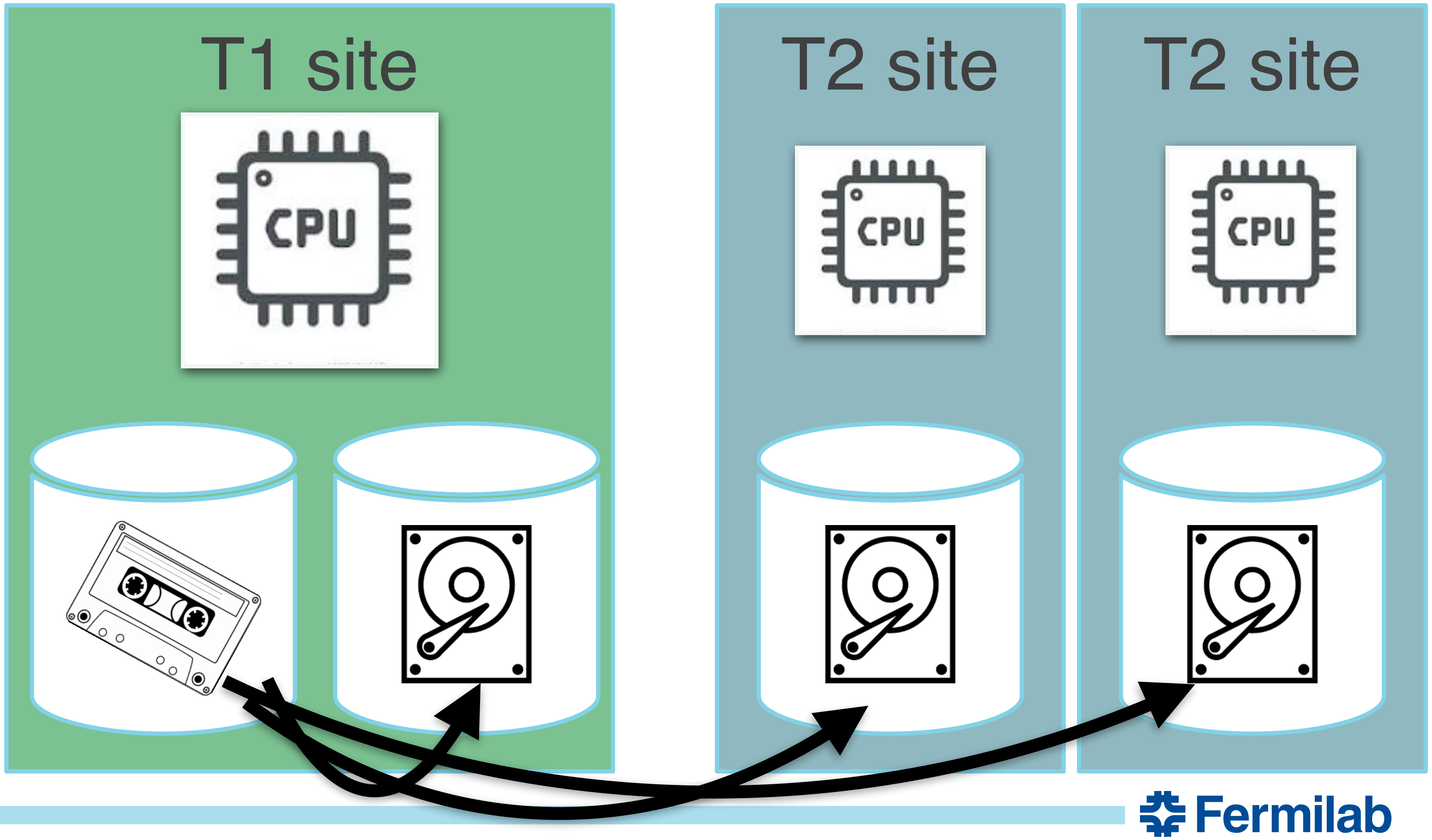
Data reconstruction



Data reconstruction

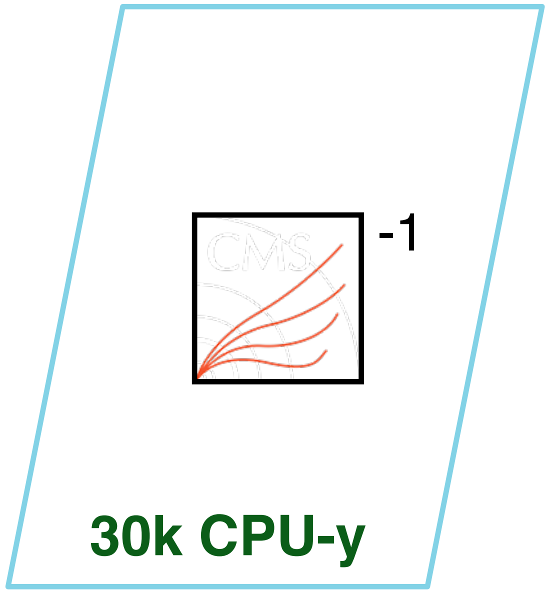


- Copy raw input from tape to many sites



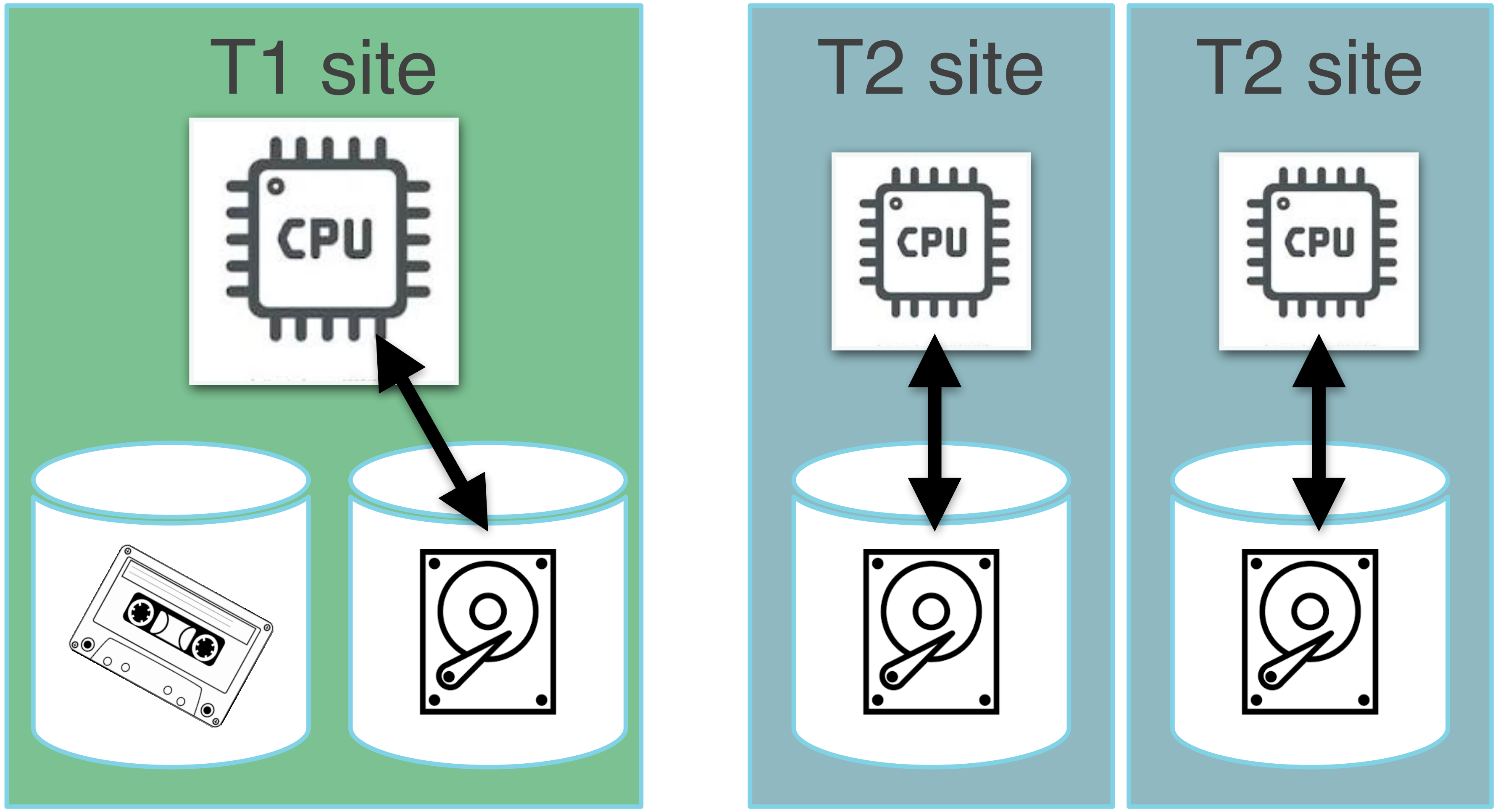
Data reconstruction

- Copy raw input from tape to many sites
- Process data



-1

30k CPU-y

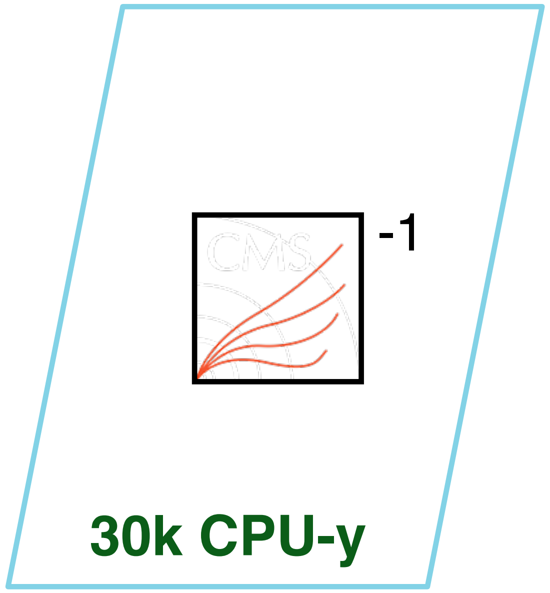


T1 site

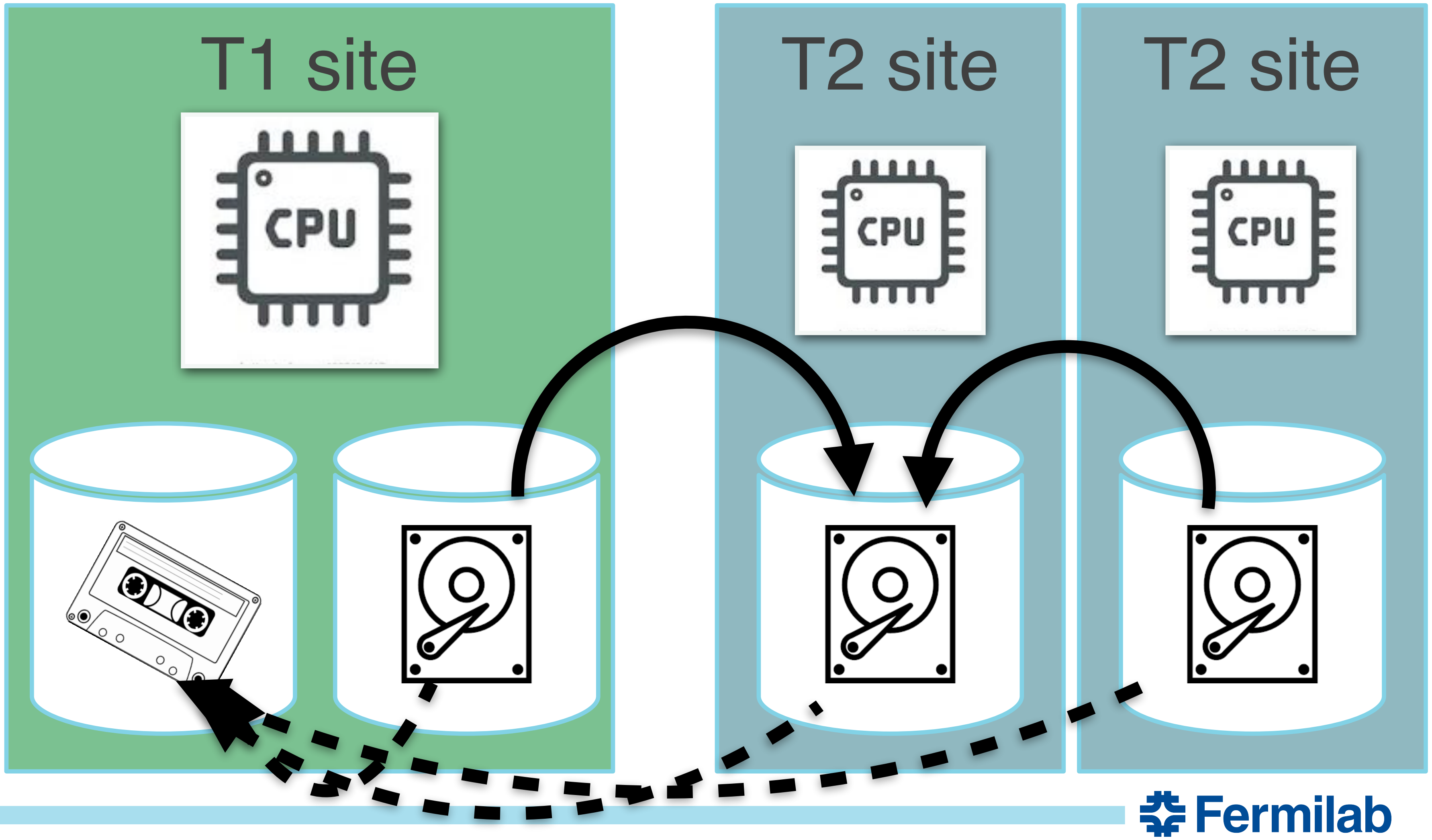
T2 site

T2 site

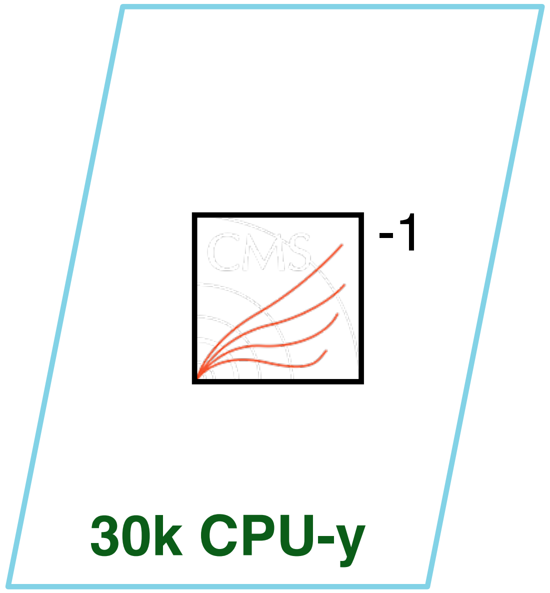
Data reconstruction



- Copy raw input from tape to many sites
- Process data
- Collect output on disk & tape



Data reconstruction


-1

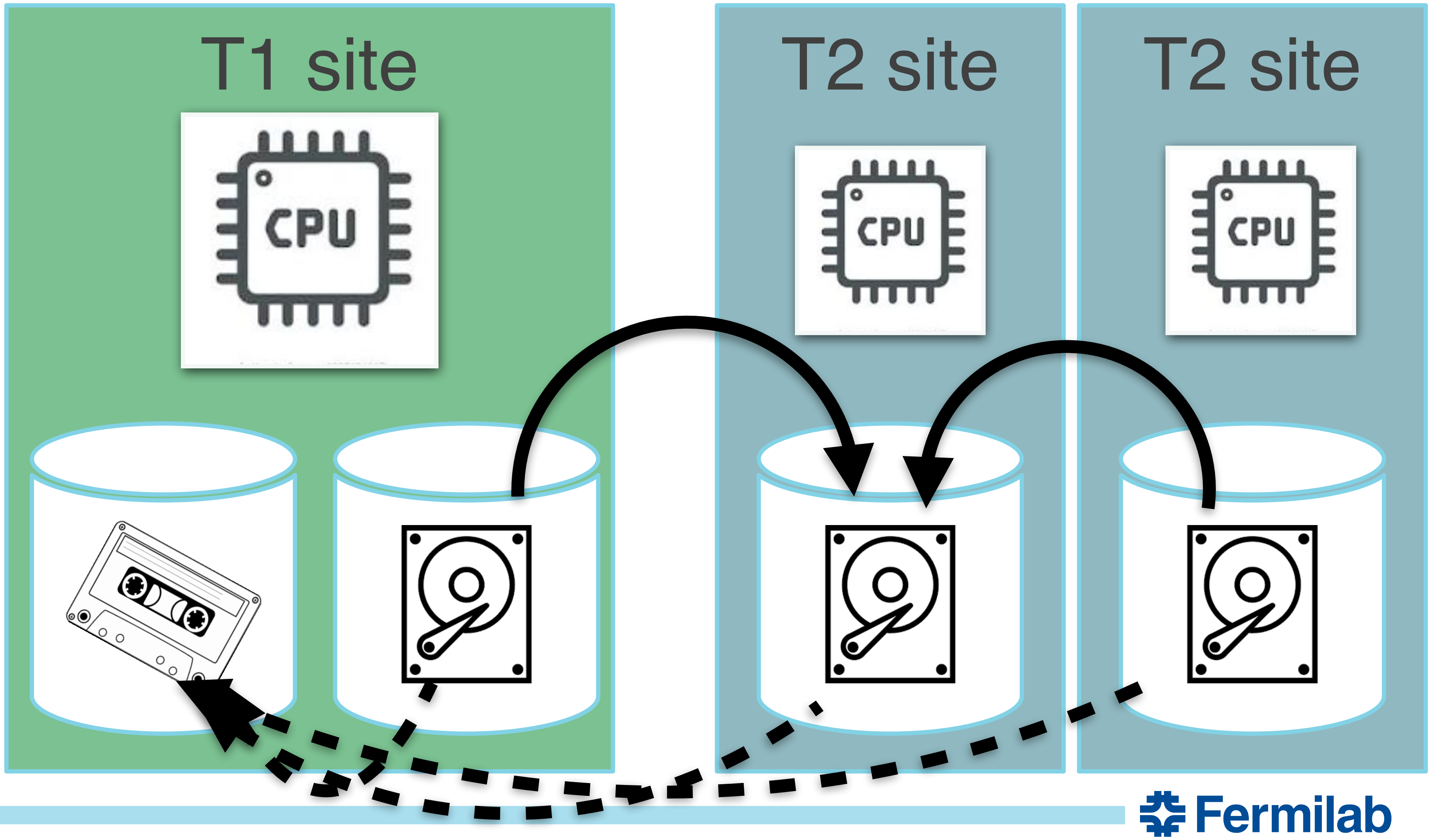
30k CPU-y

- Copy raw input from tape to many sites
- Process data
- Collect output on disk & tape

Transfers via File Transfer Service “third-party copy” between storage; orchestrated via Rucio data management

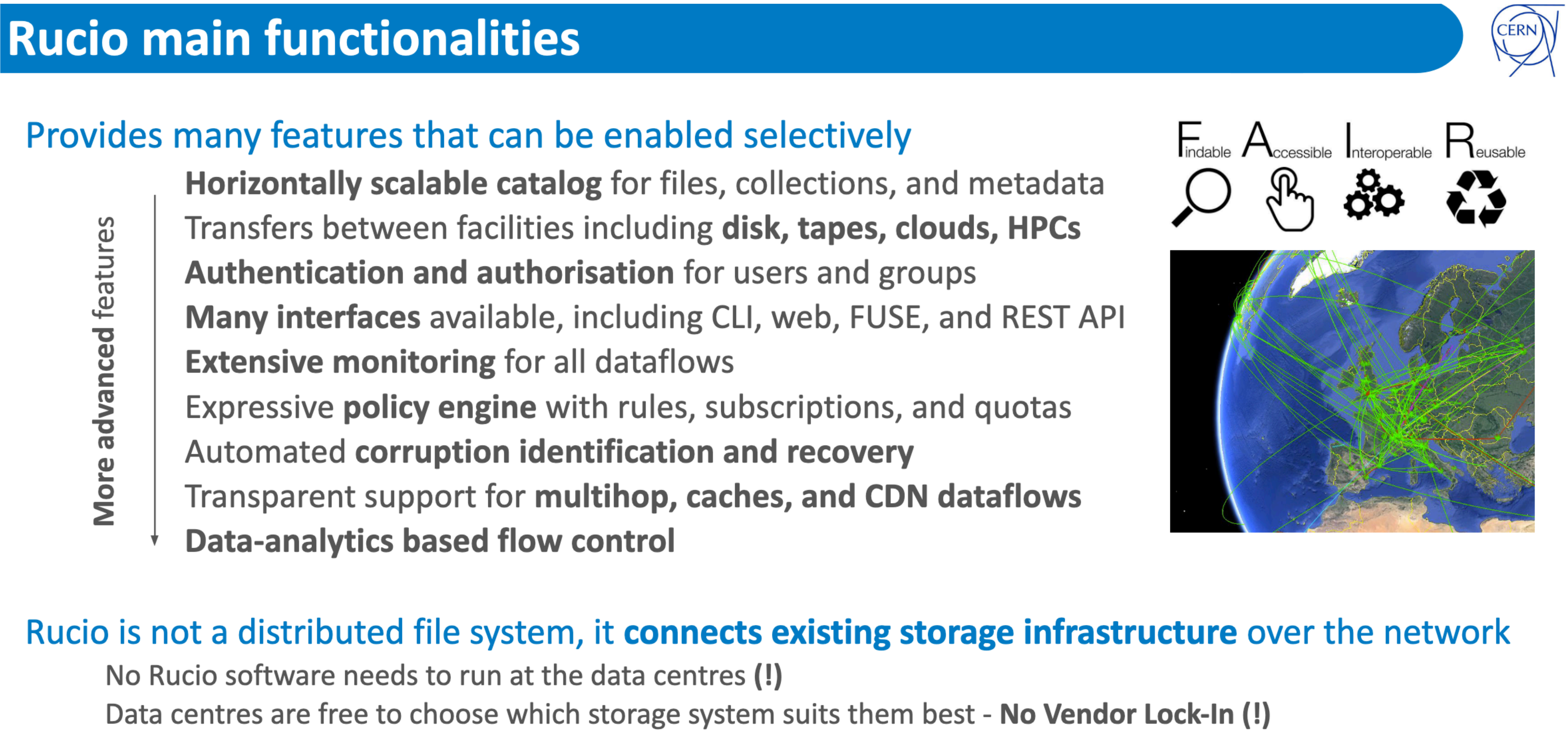
FTS

SCIENTIFIC DATA MANAGEMENT RUCIO



Data cataloging

Mario Lassnig



Rucio main functionalities

Provides many features that can be enabled selectively

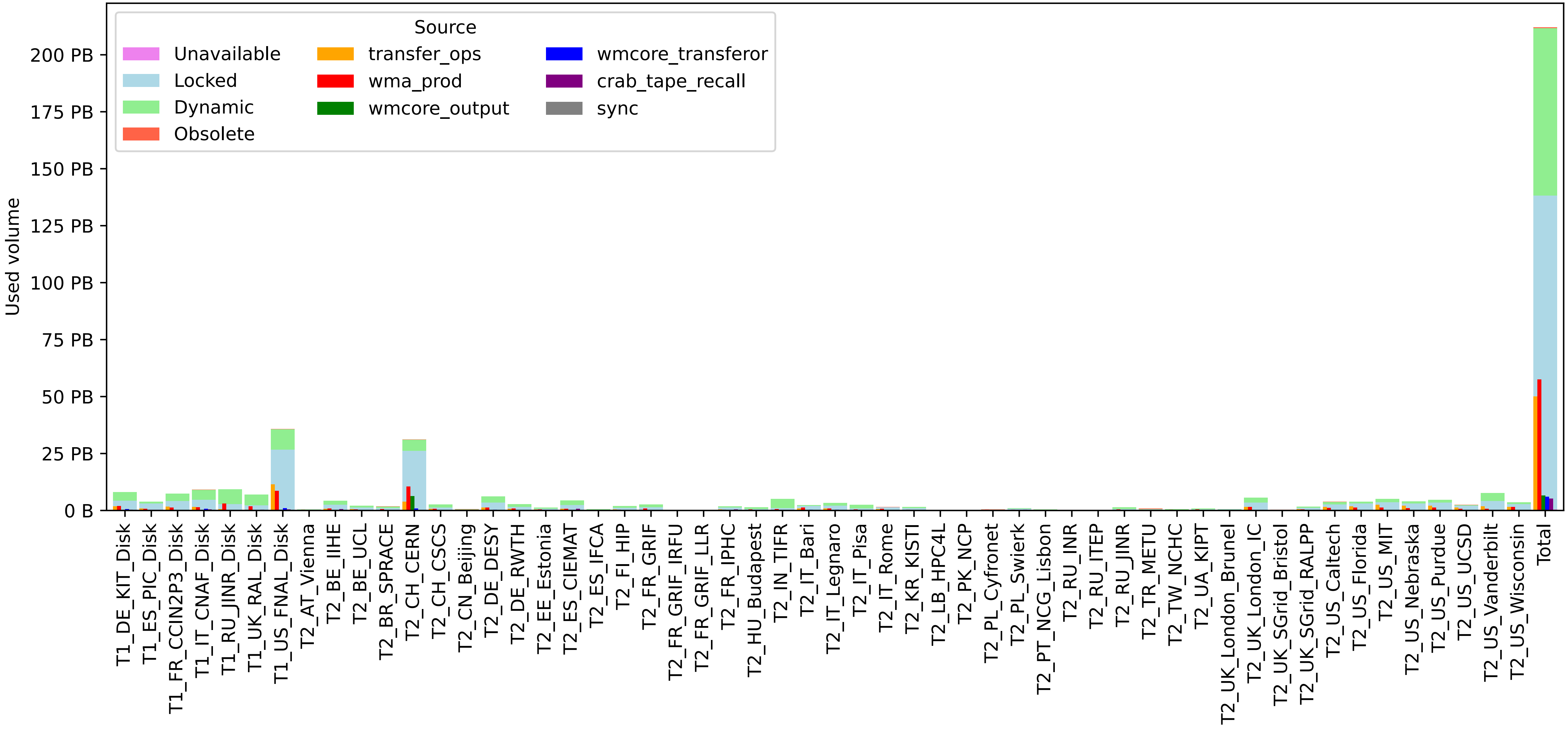
More advanced features

- **Horizontally scalable catalog** for files, collections, and metadata
- Transfers between facilities including **disk, tapes, clouds, HPCs**
- **Authentication and authorisation** for users and groups
- **Many interfaces** available, including CLI, web, FUSE, and REST API
- **Extensive monitoring** for all dataflows
- Expressive **policy engine** with rules, subscriptions, and quotas
- Automated **corruption identification and recovery**
- Transparent support for **multihop, caches, and CDN dataflows**
- **Data-analytics based flow control**

Rucio is not a distributed file system, it **connects existing storage infrastructure** over the network

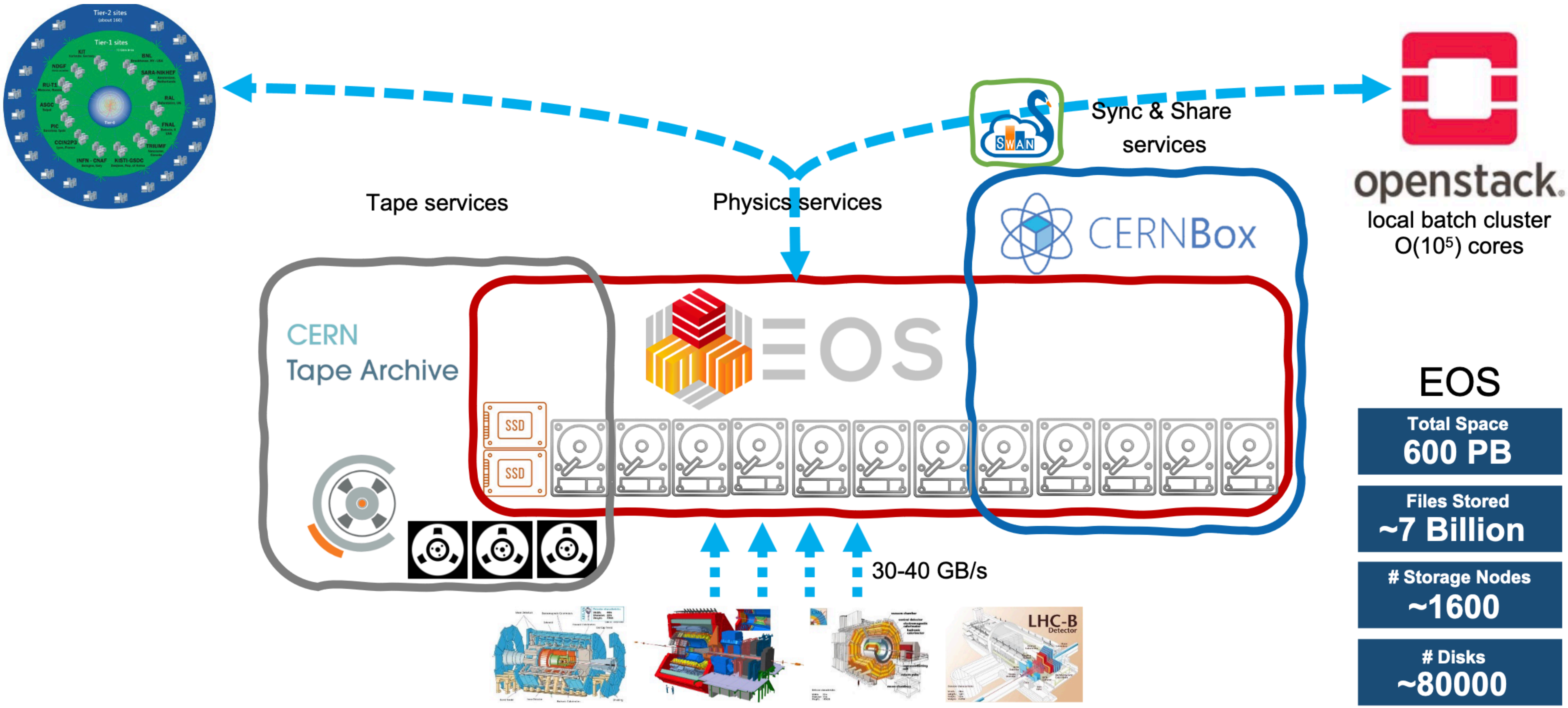
- No Rucio software needs to run at the data centres **(!)**
- Data centres are free to choose which storage system suits them best - **No Vendor Lock-In (!)**

Data cataloging



Disk storage at CERN

Vladimir Bayhl



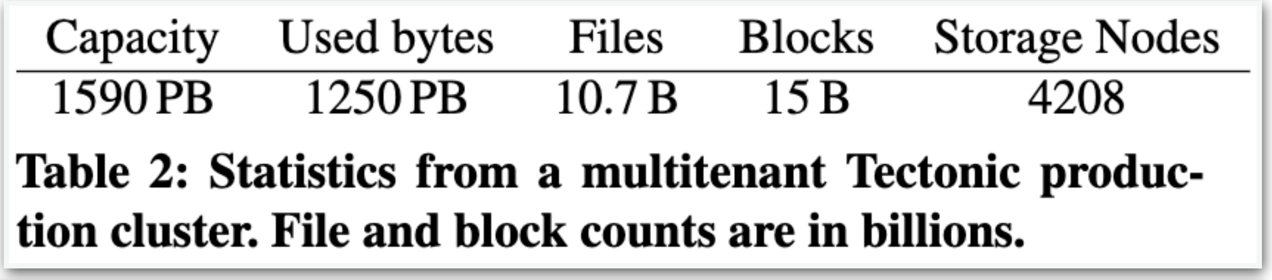
EOS

Total Space	600 PB
Files Stored	~7 Billion
# Storage Nodes	~1600
# Disks	~80000

Disk storage elsewhere

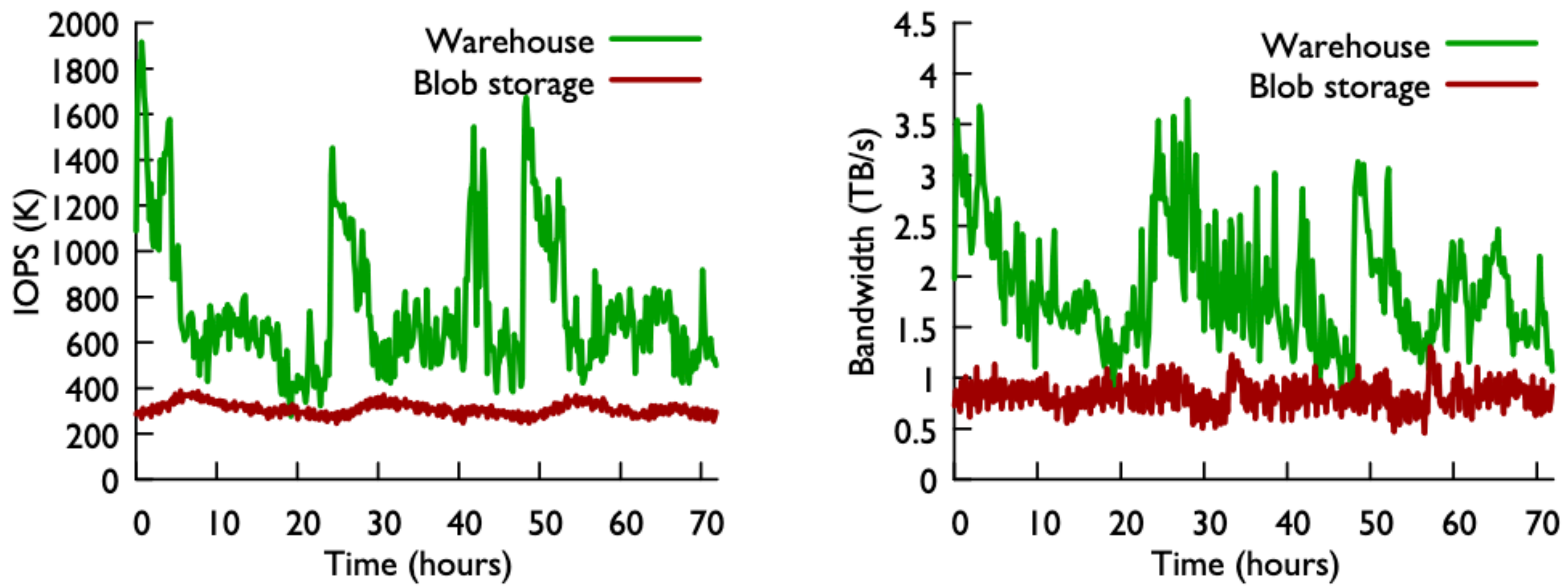
- Facebook Tectonic FS: one disk cluster per datacenter, two basic workloads:
  - Blob storage: pictures/videos
    - Steady-state IOPS, random access
  - Warehouse: engagement data (clicks/likes)
    - Bursty, more sequential access
- Potential analog:
  - Blob storage: pileup mixing in generation
  - Warehouse: analysis queries
- Many spindles!
  - Load-balance➔performance
  - Scalability via indirection
    - 3 (!) metadata queries /access

- https://www.usenix.org/system/files/fast21-pan.pdf



Capacity	Used bytes	Files	Blocks	Storage Nodes
1590 PB	1250 PB	10.7 B	15 B	4208

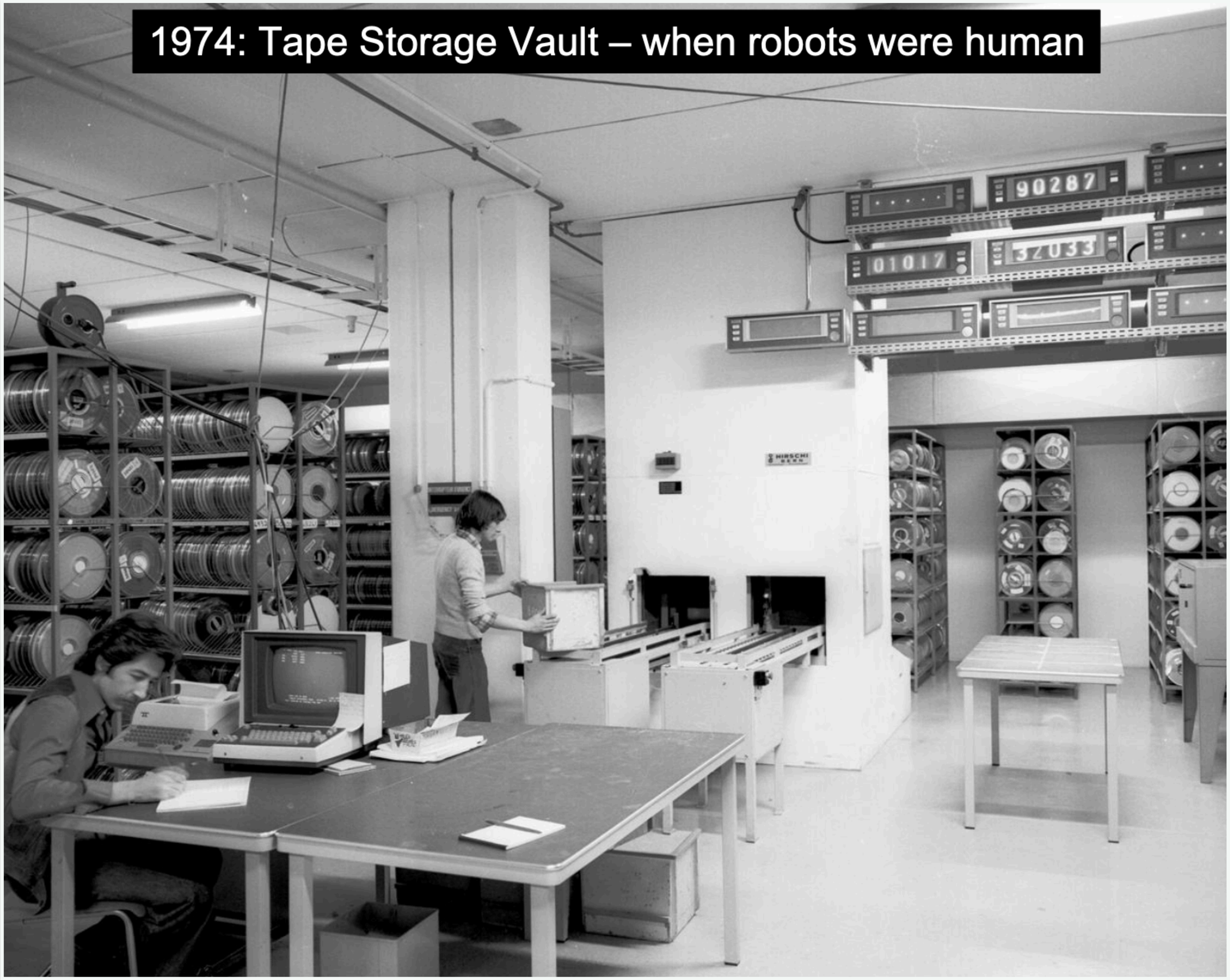
Table 2: Statistics from a multitenant Tectonic production cluster. File and block counts are in billions.



(a) Aggregate cluster IOPS **(b) Aggregate cluster bandwidth**

Tape storage

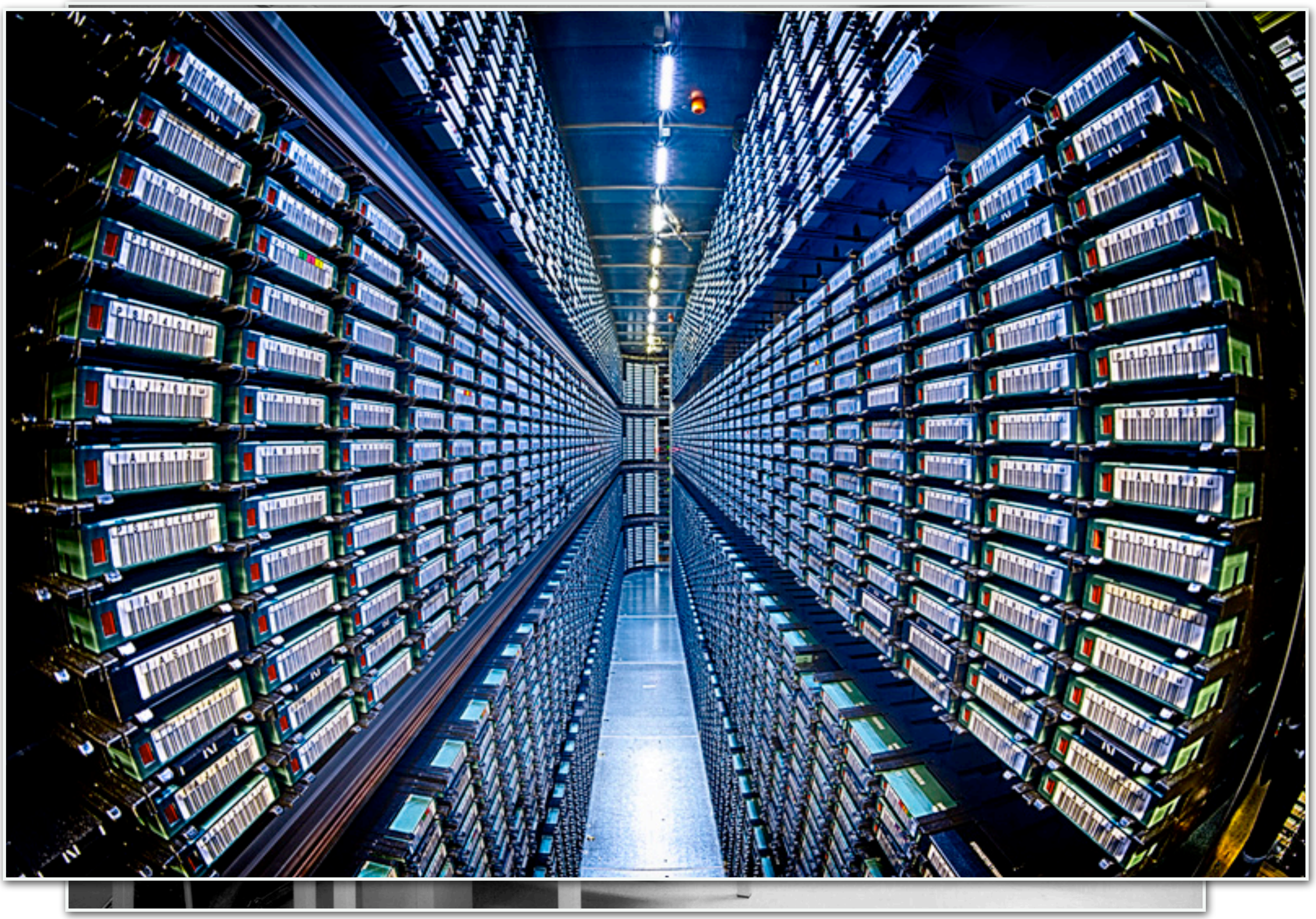
Michael Davis



1974: Tape Storage Vault – when robots were human

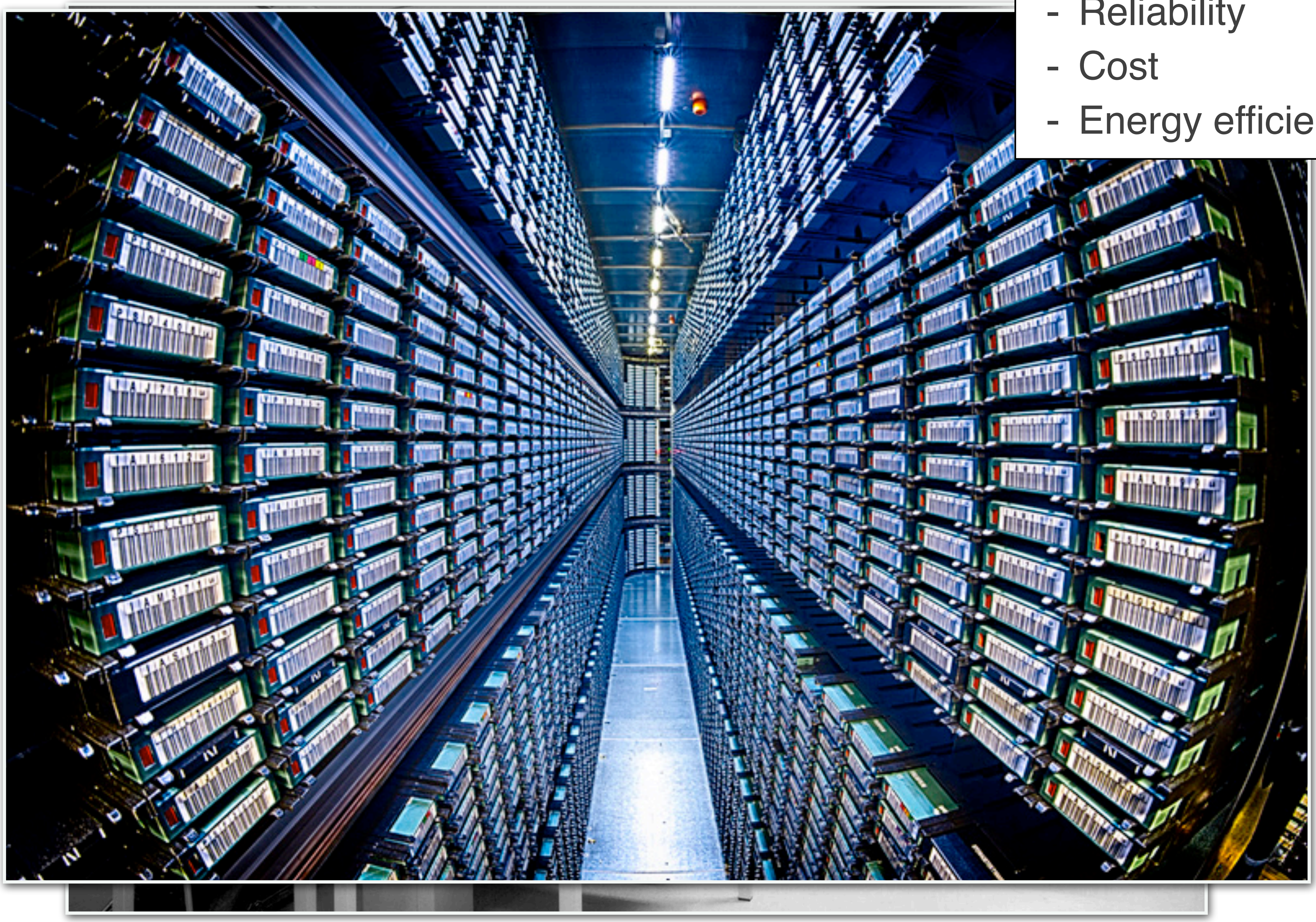
Tape storage

Michael Davis



Tape storage

Michael Davis



- Advantages:
  - Reliability
  - Cost
  - Energy efficiency

Tape storage

Michael Davis

- Advantages:
  - Reliability
  - Cost
  - Energy efficiency

The Outlook for Tape Technology

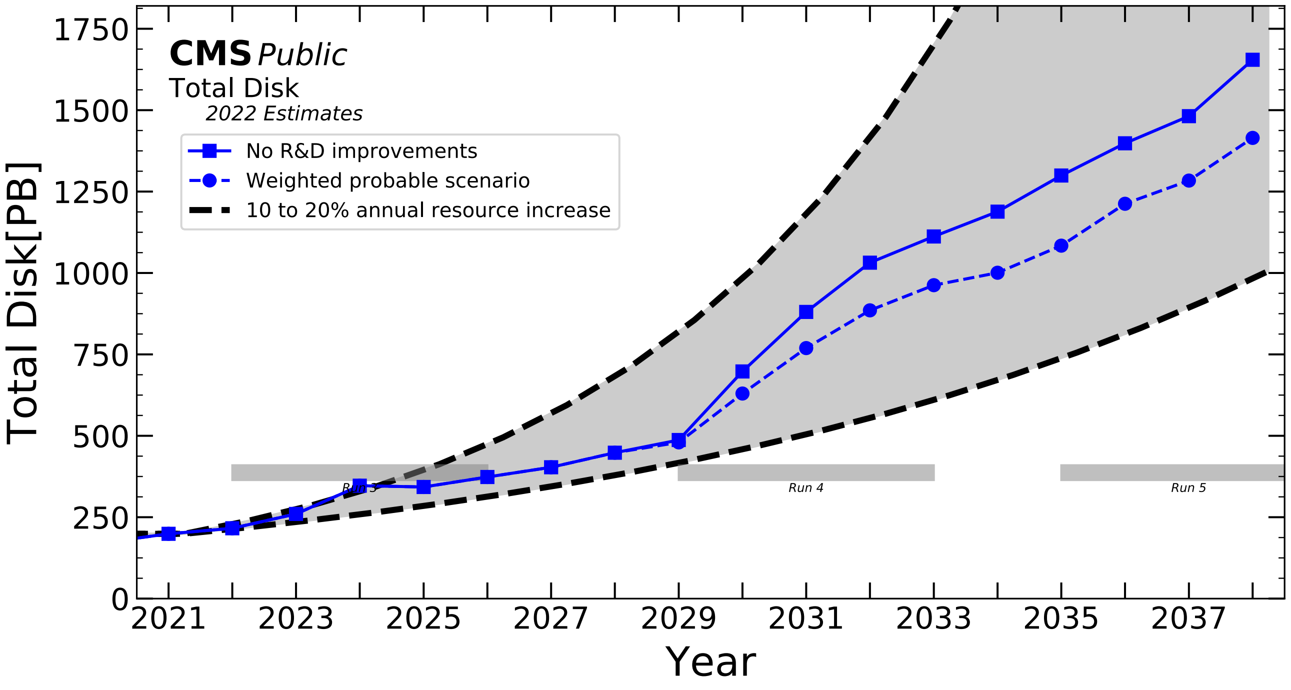
New Advanced Materials

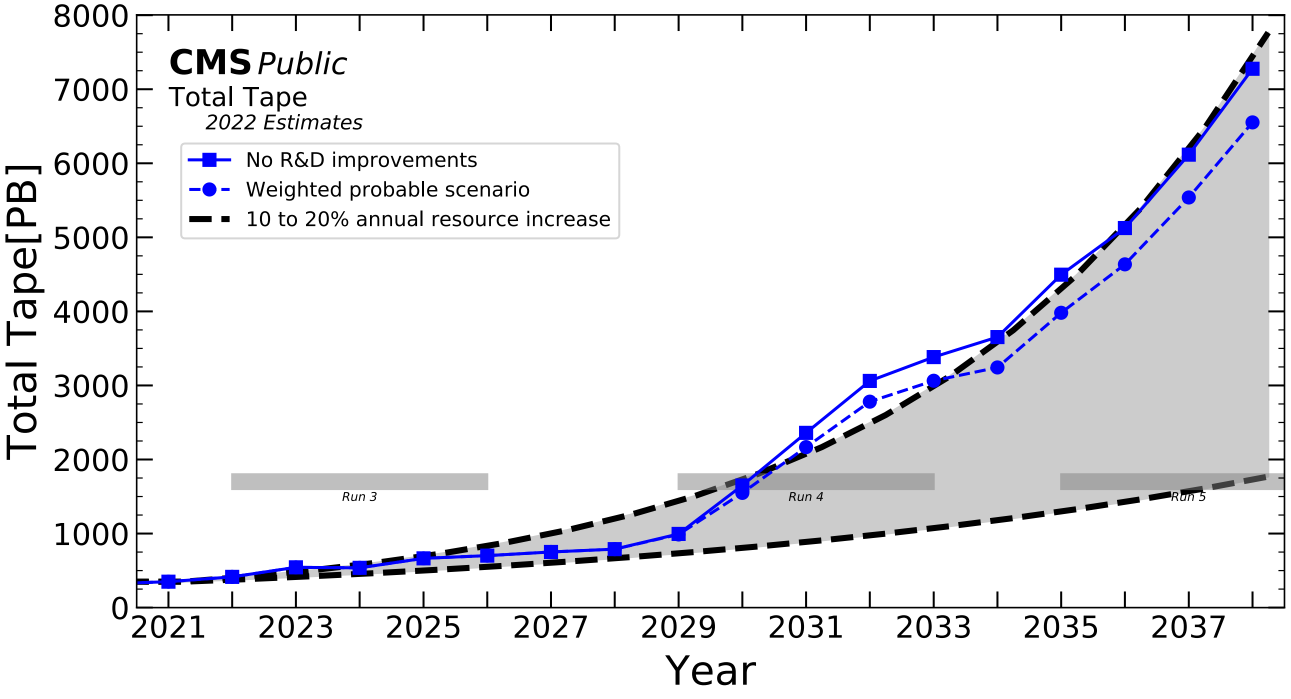
- Very fine magnetic particles
- Smooth surfaces with low friction
- 3D stacking of magnetic particles

Disk technologies are pushing the limits of storage density. Tapes have plenty of room to improve capacity.

- **The cost advantages of tape will increase over time** https://twiki.cern.ch/twiki/bin/view/CMSPublic/CMSOfflineComputingResults

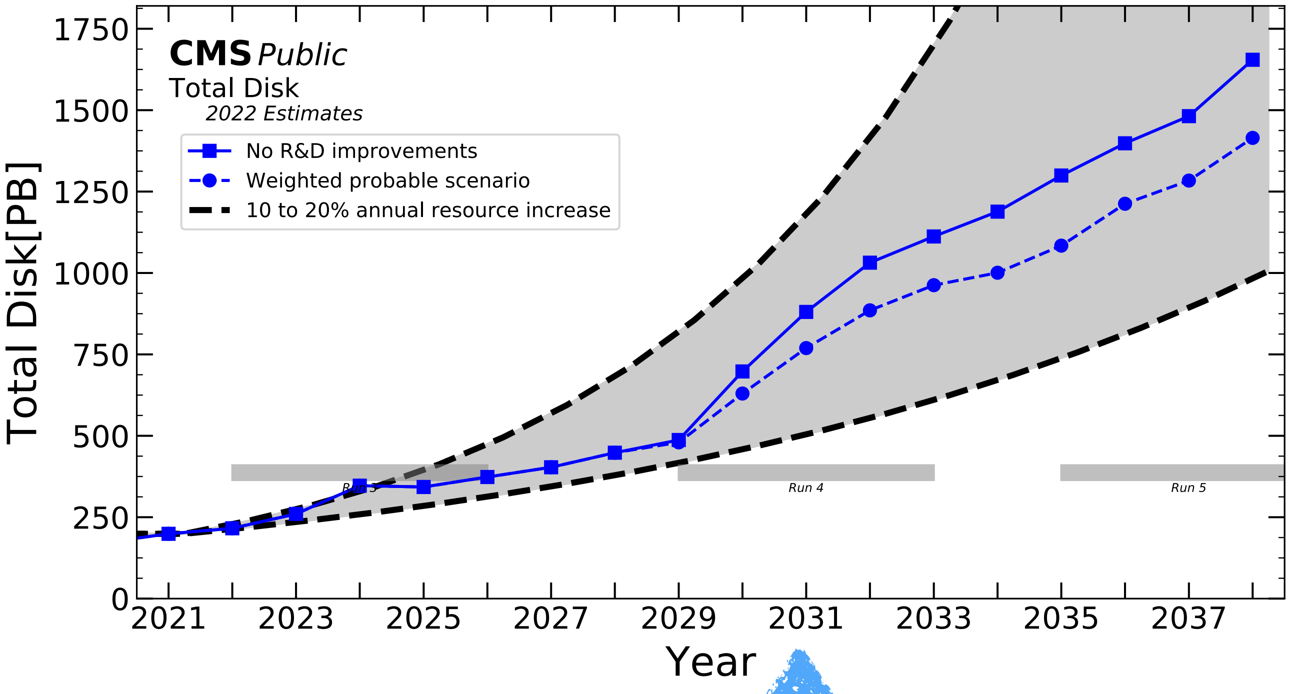
Projected usage

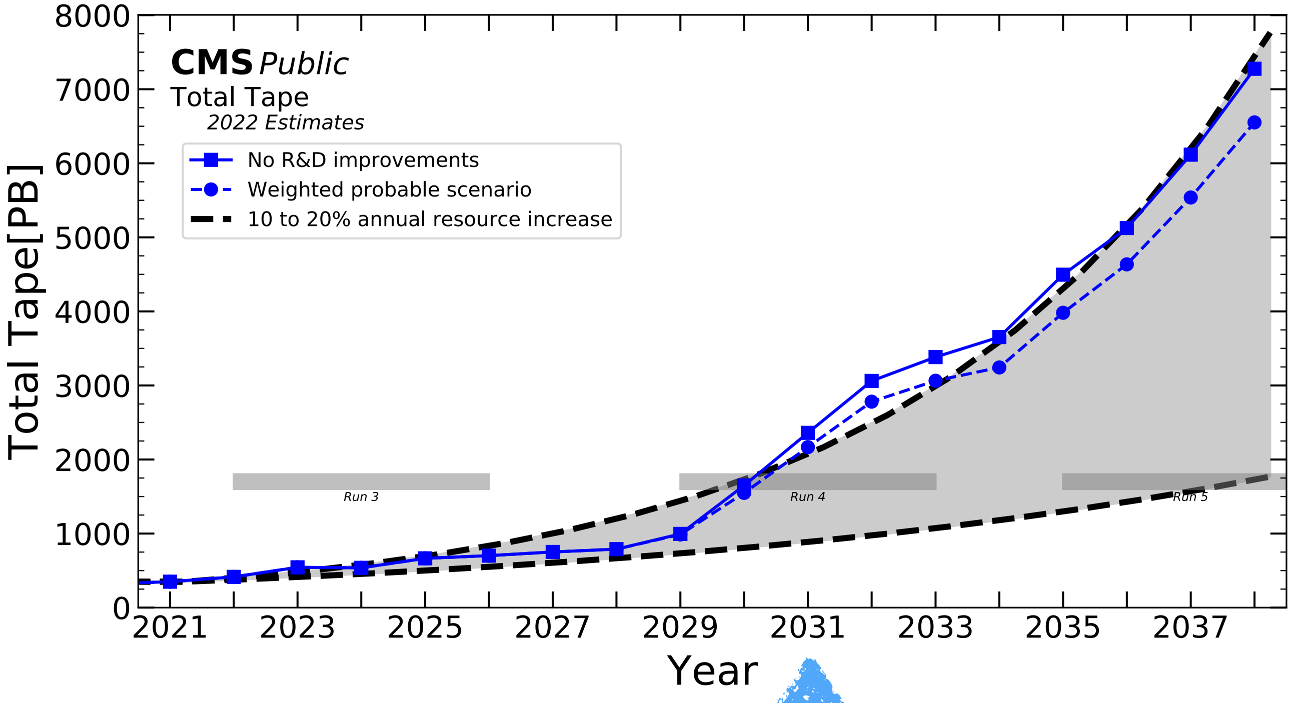


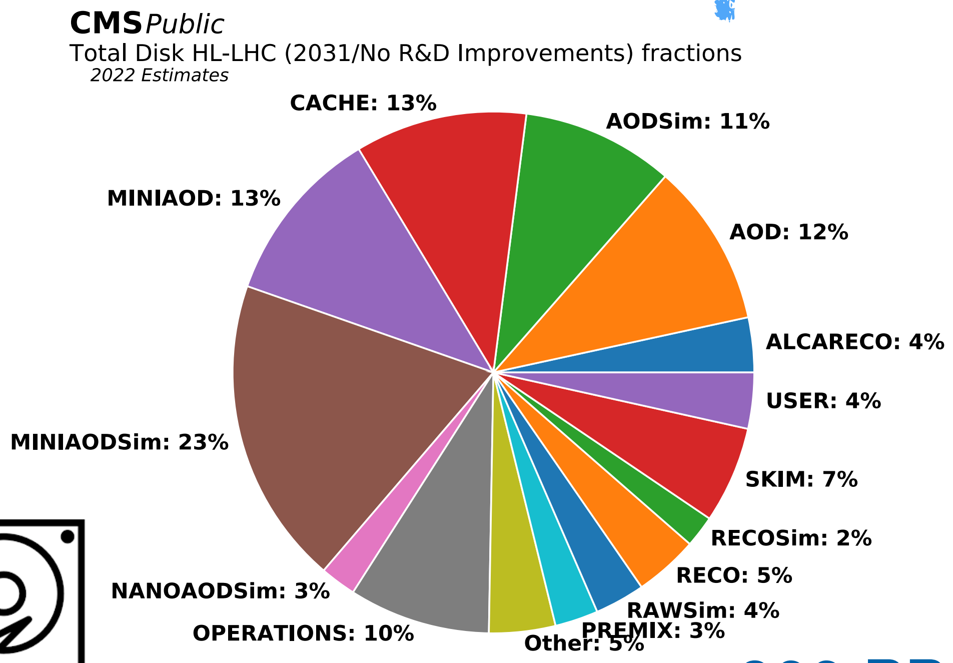


https://twiki.cern.ch/twiki/bin/view/CMSPublic/CMSOfflineComputingResults

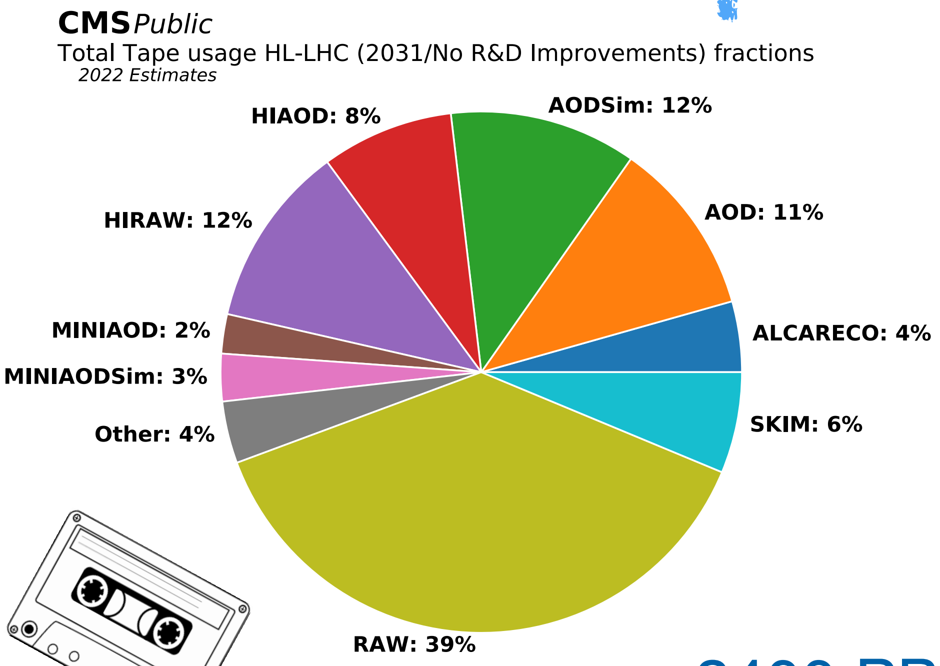
Projected usage







900 PB

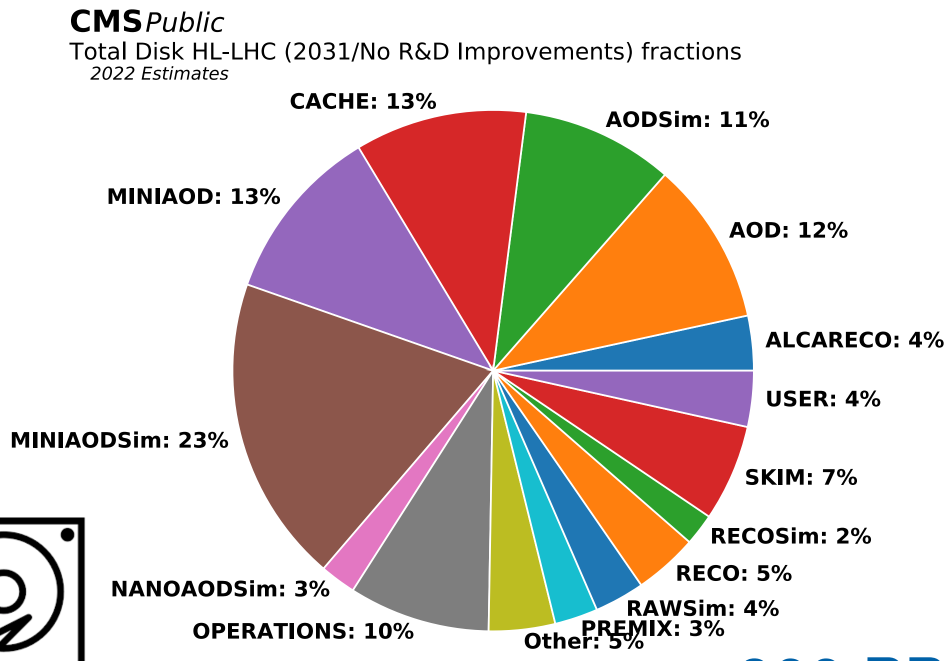


2400 PB

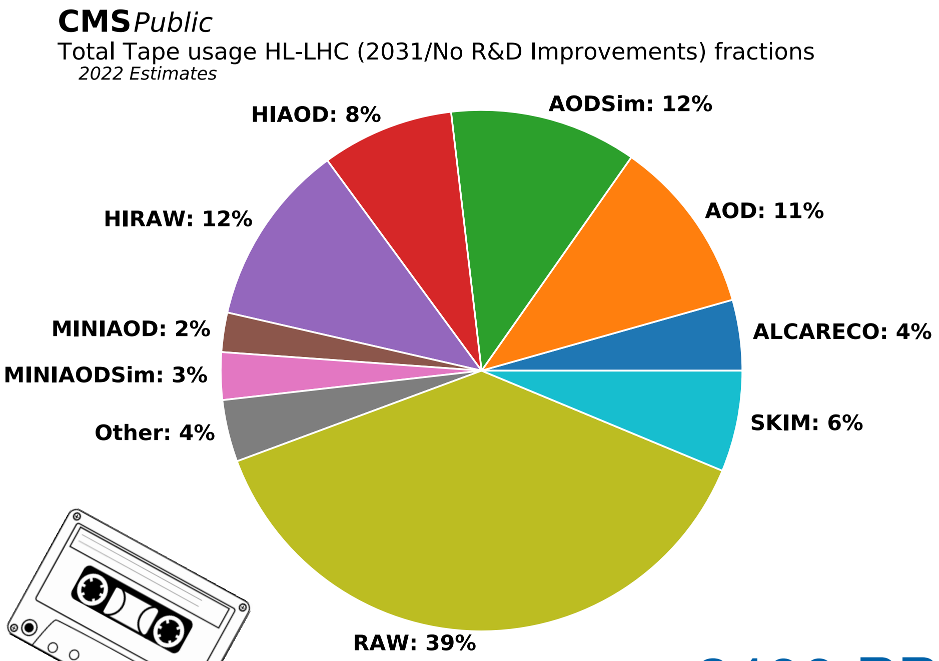
Fermilab

Disk as a cache

- Disk is expensive (vs. tape)
  - Only MiniAOD, NanoAOD data tiers reliably on disk now
    - Ok because of 10+y experience with detector to know what we need
    - For HL-LHC, new detectors may require more time with low-level information
- Best cache: all the columns you need, none you don’t
  - Different set of columns needed for different PDs, analyses
  - Not all rows read if filtering (skimming)
  - **How much can we reduce disk use from PD*tier granularity we have now?**

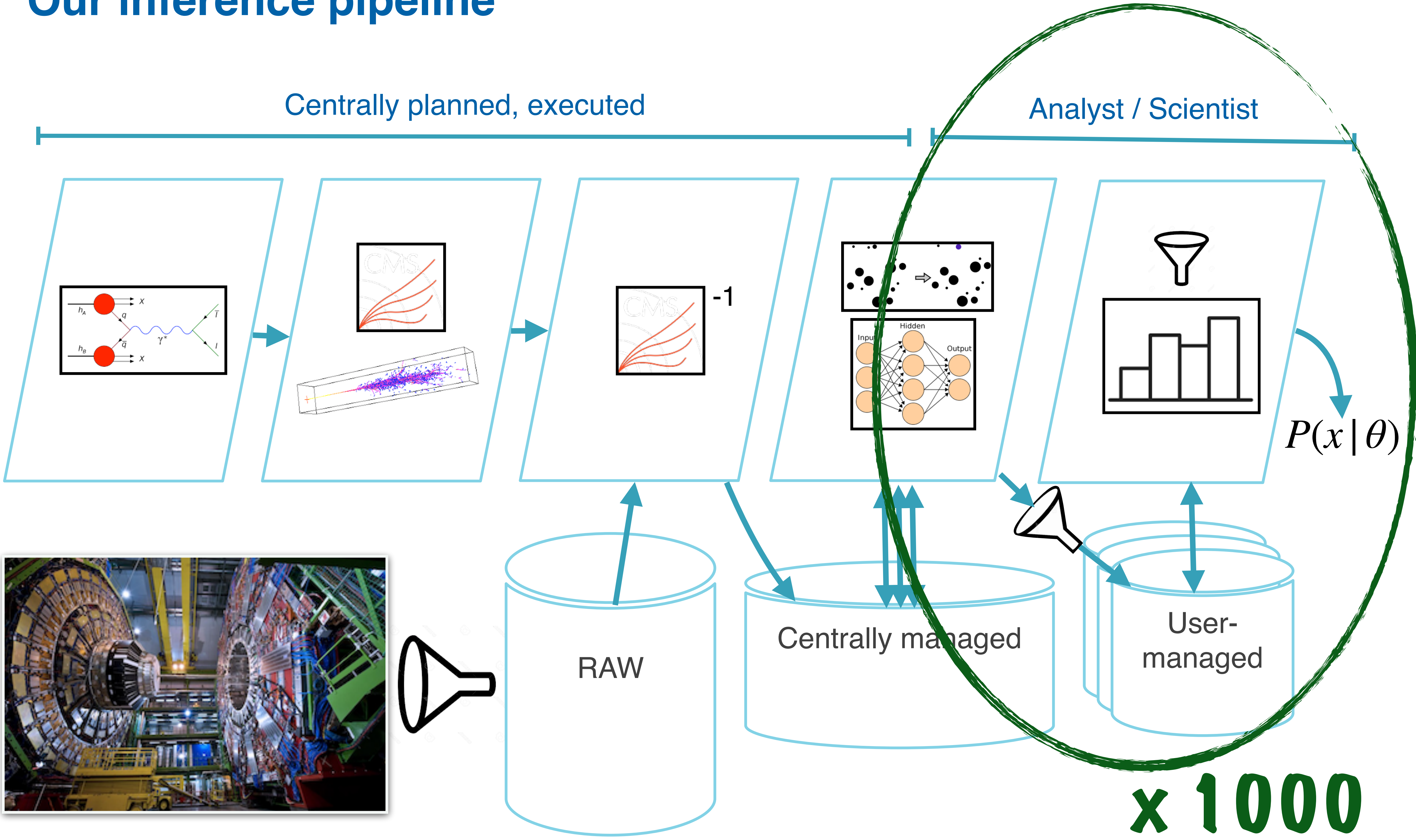


900 PB



2400 PB

Our inference pipeline



Centrally planned, executed

Analyst / Scientist

RAW

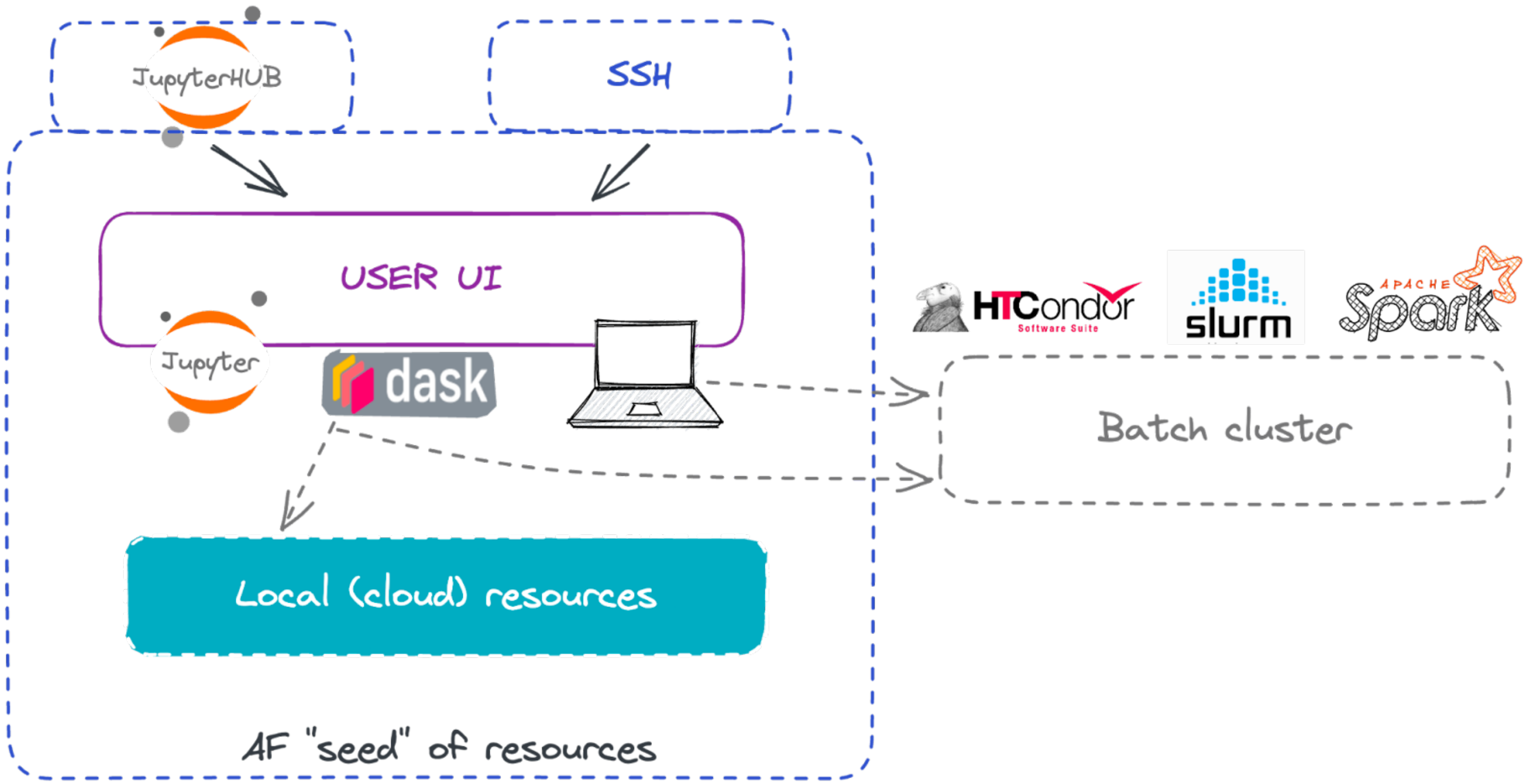
Centrally managed

User-managed

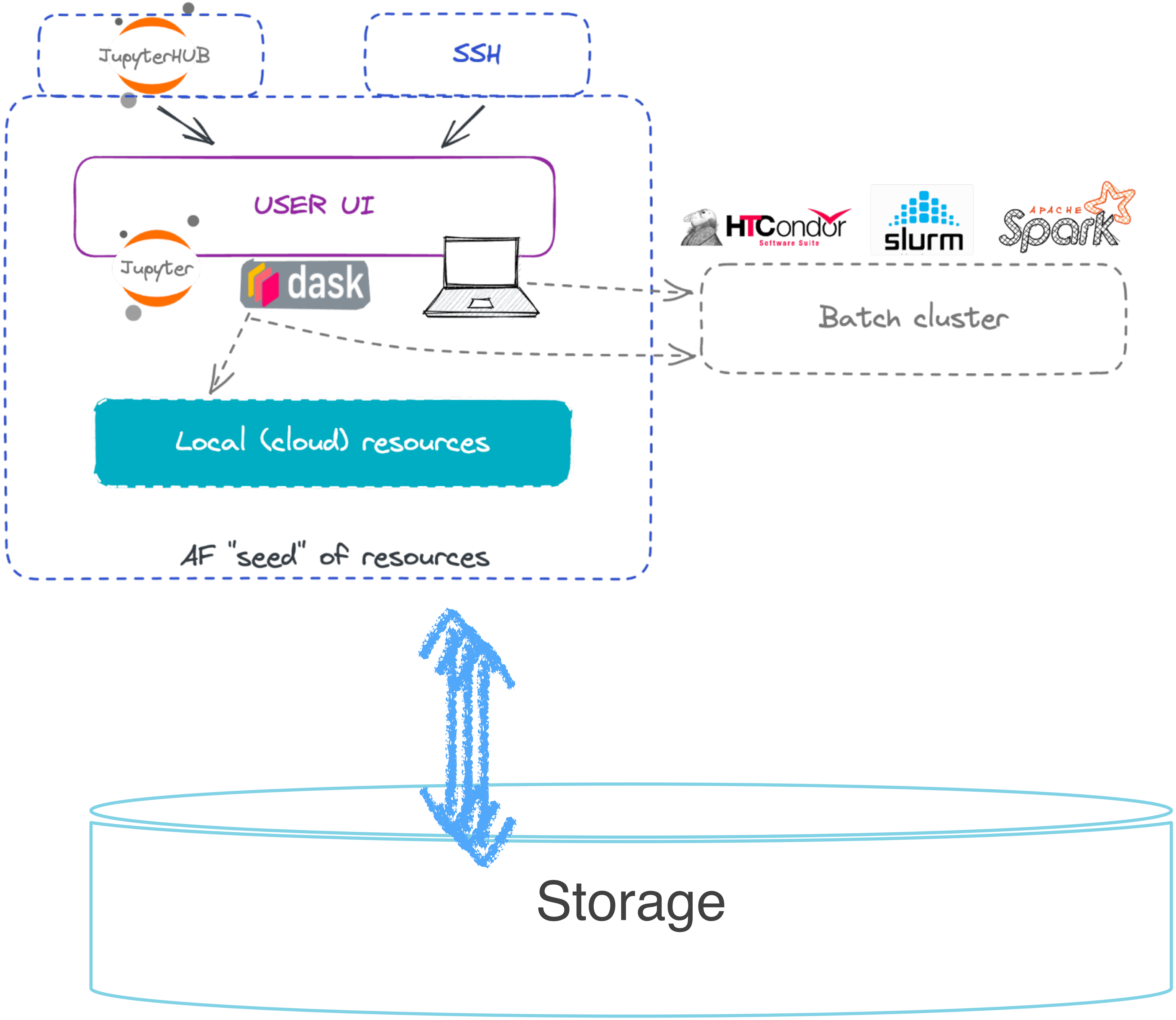
$P(x\,|\,\theta)$

x 1000

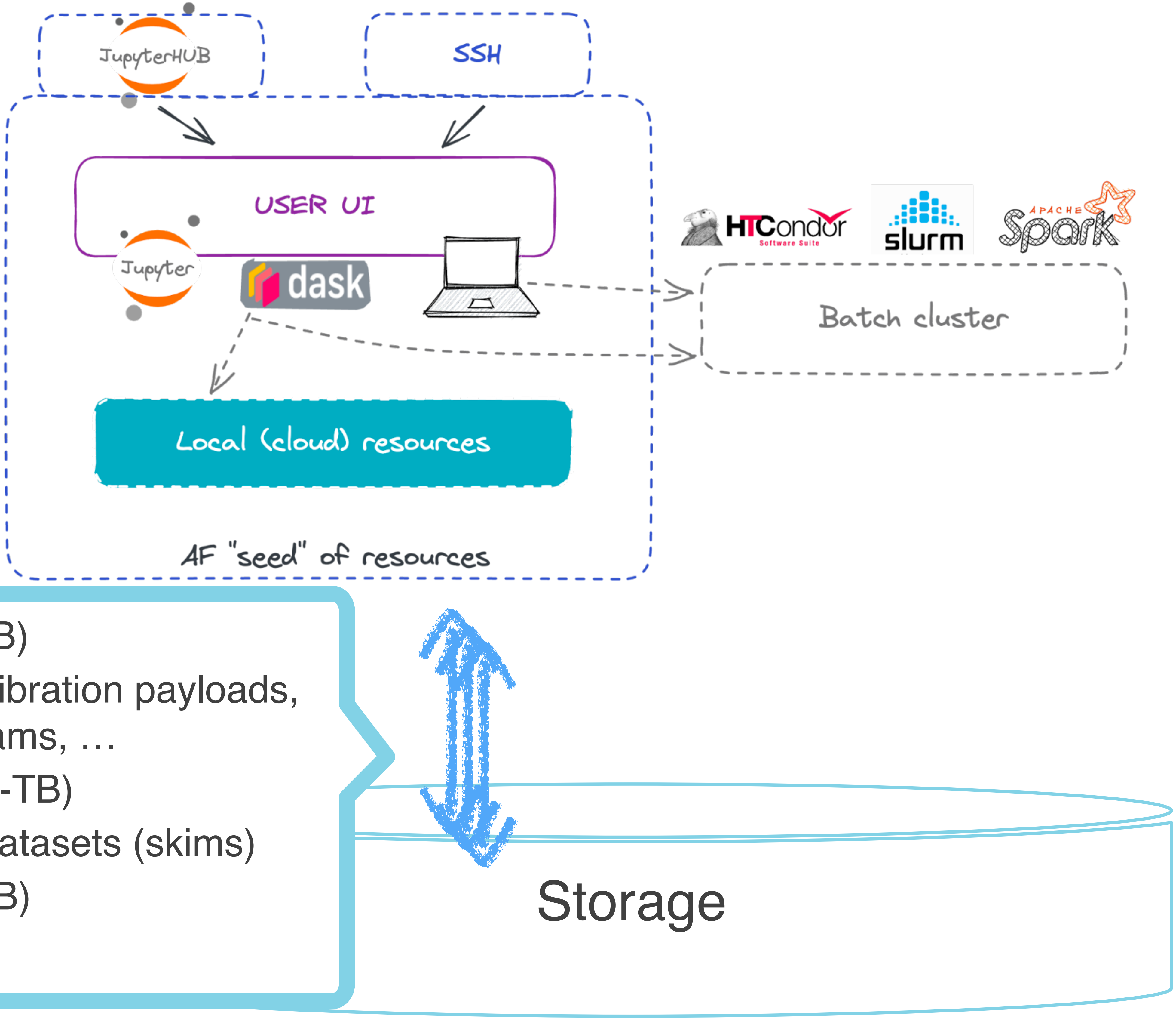
Facility view



Facility view

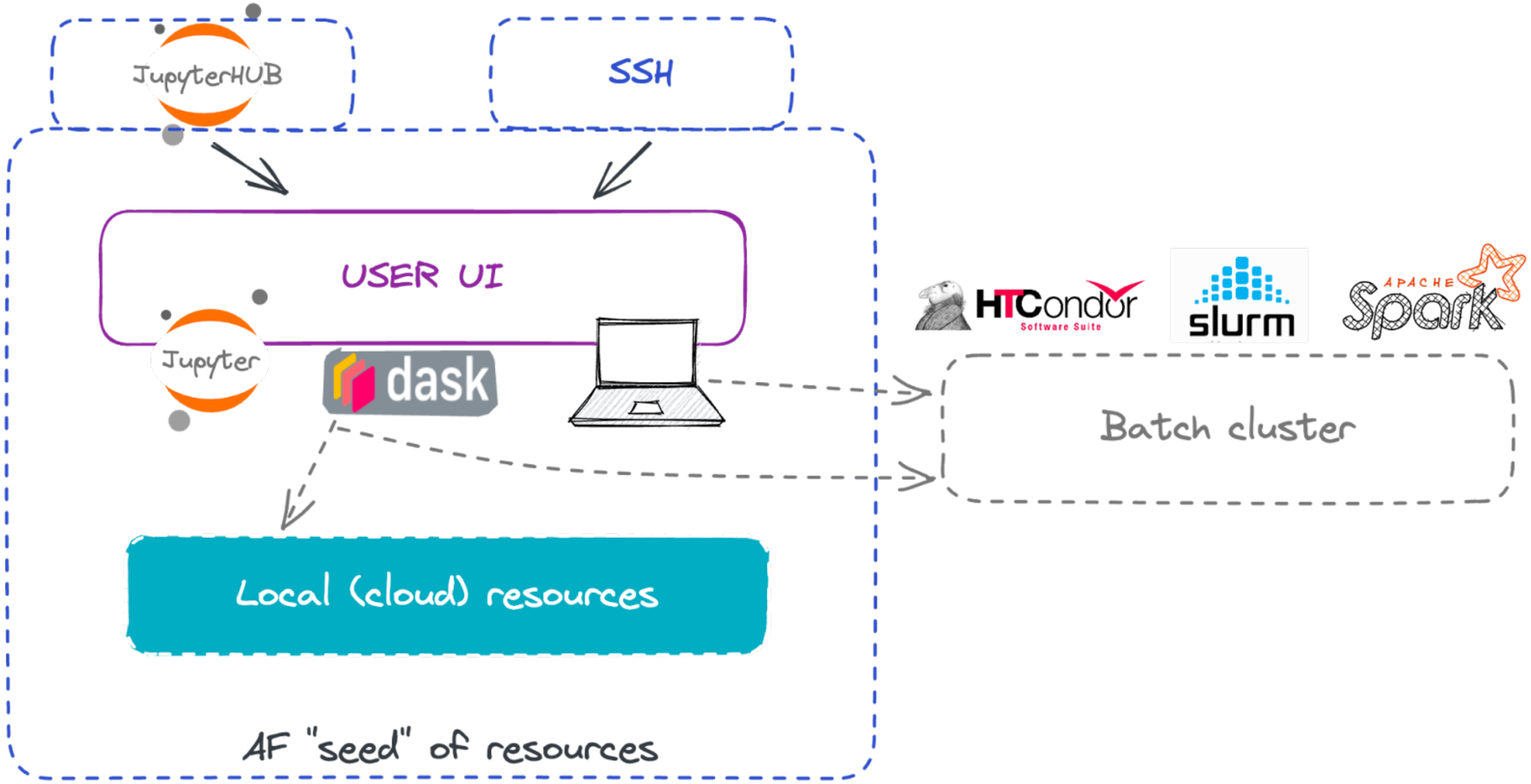


Facility view

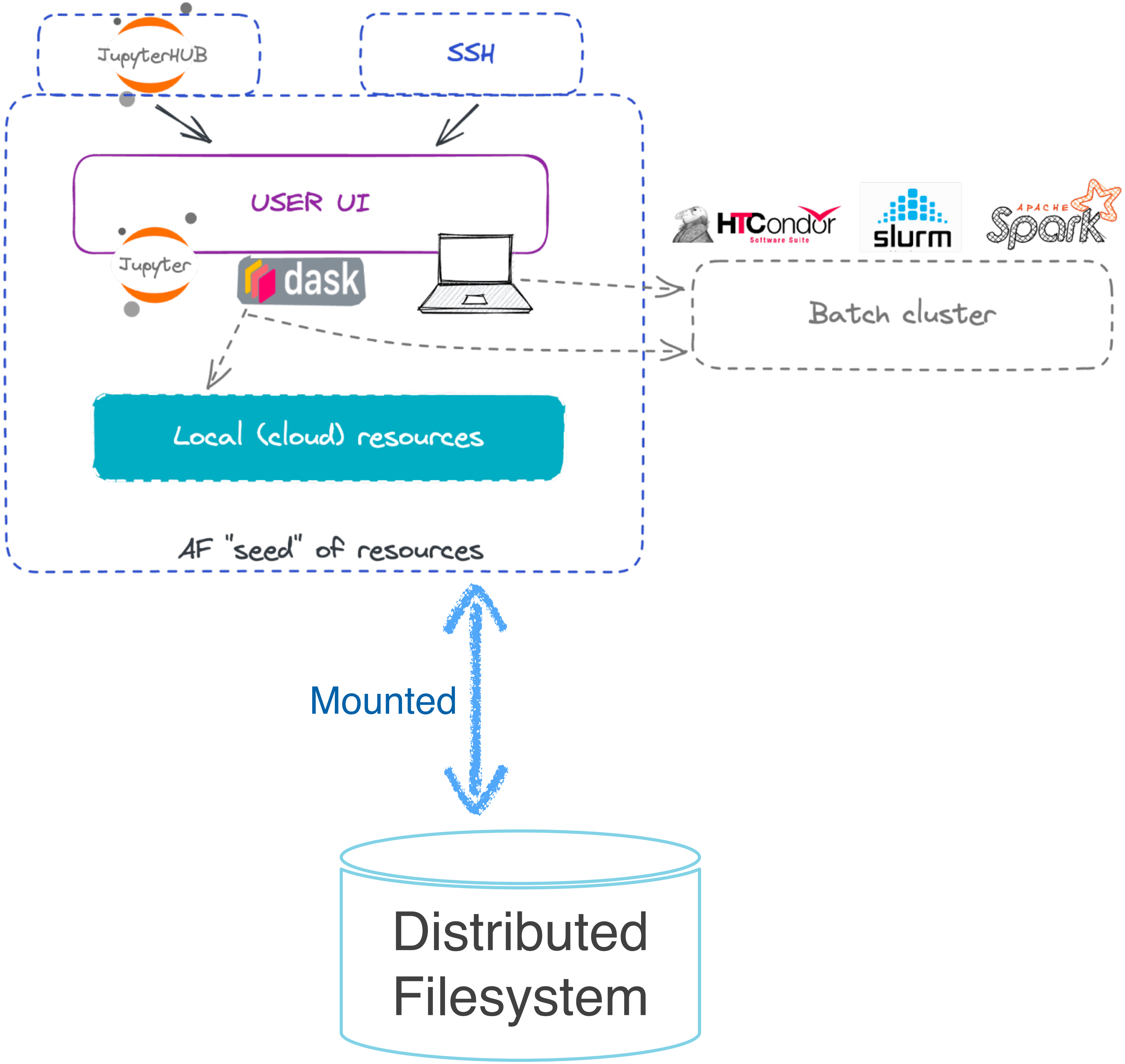


- Small data (kB-GB)
  - User code, calibration payloads, output histograms, …
- Medium data (GB-TB)
  - Intermediate datasets (skims)
- Large data (TB-PB)
  - Input datasets

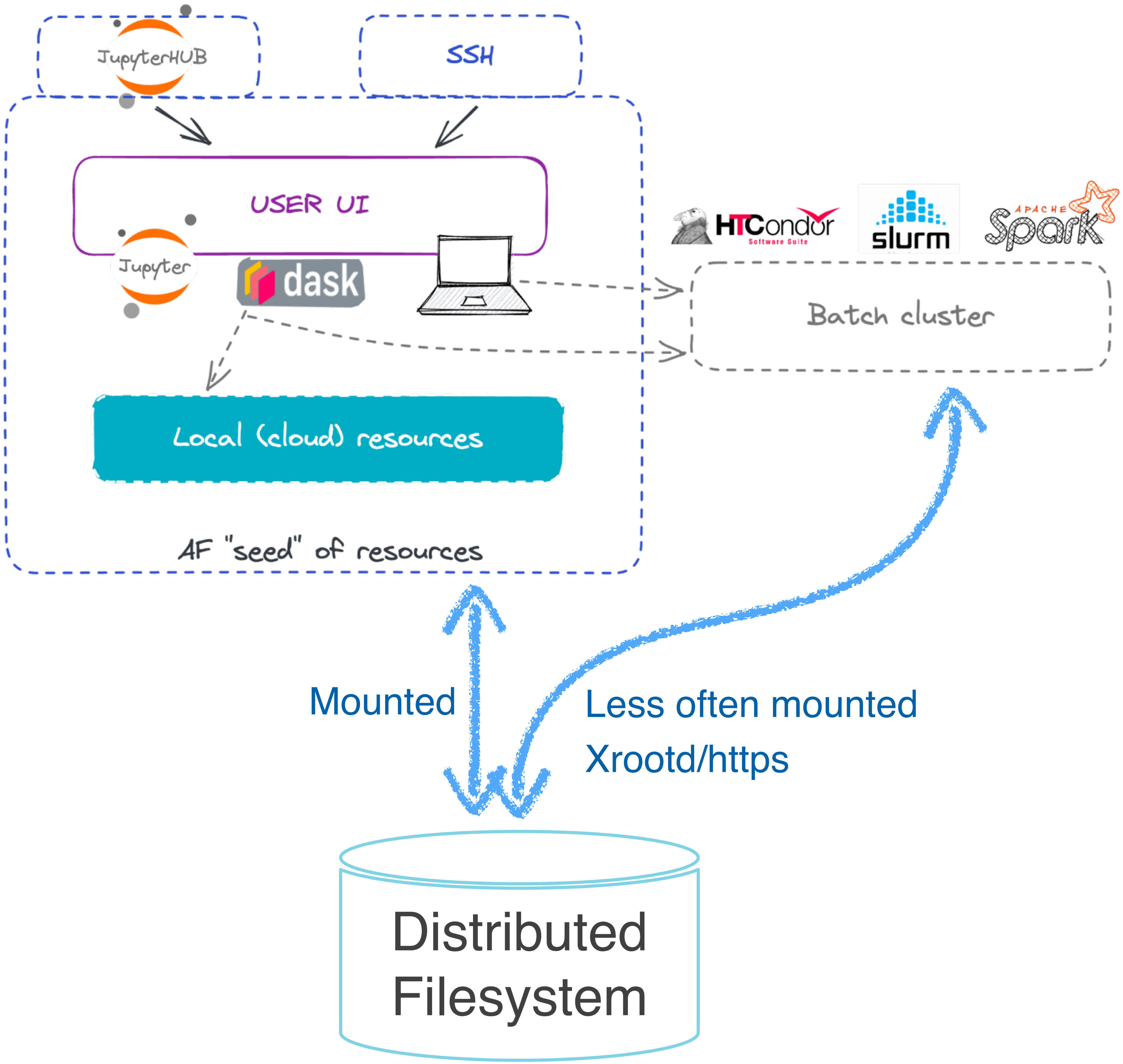
Small data


JupyterHUB
SSH
USER UI
Jupyter
dask
HTCondor
Software Suite
slurm
APACHE Spark
Batch cluster
Local (cloud) resources
AF "seed" of resources

Small data


JupyterHUB
SSH
USER UI
Jupyter
dask
HTCondor Software Suite
slurm
Apache Spark
Batch cluster
Local (cloud) resources
AF "seed" of resources
Mounted
Distributed Filesystem

Small data



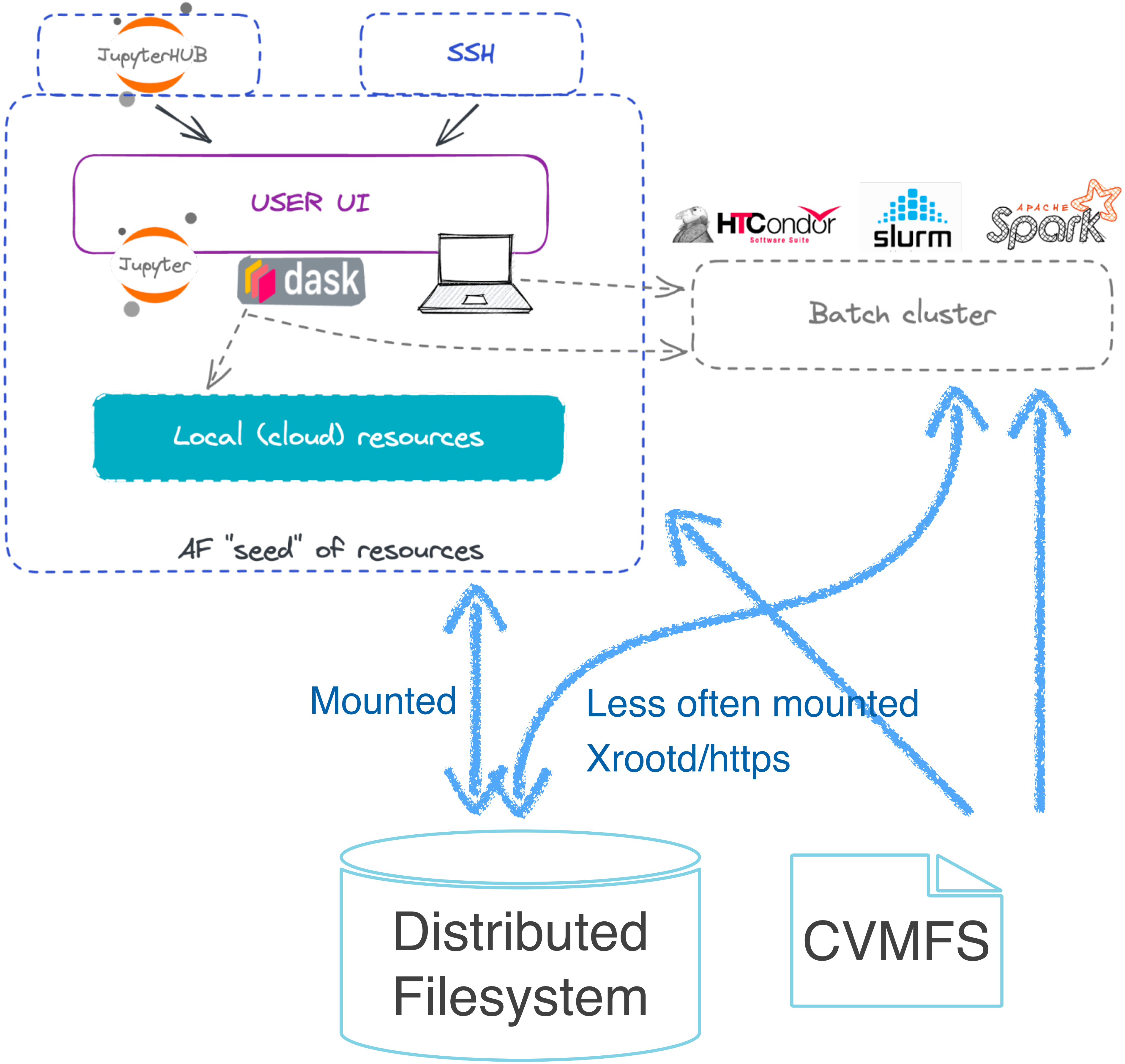
Mounted

Less often mounted

Xrootd/https

Distributed Filesystem

Small data



Mounted

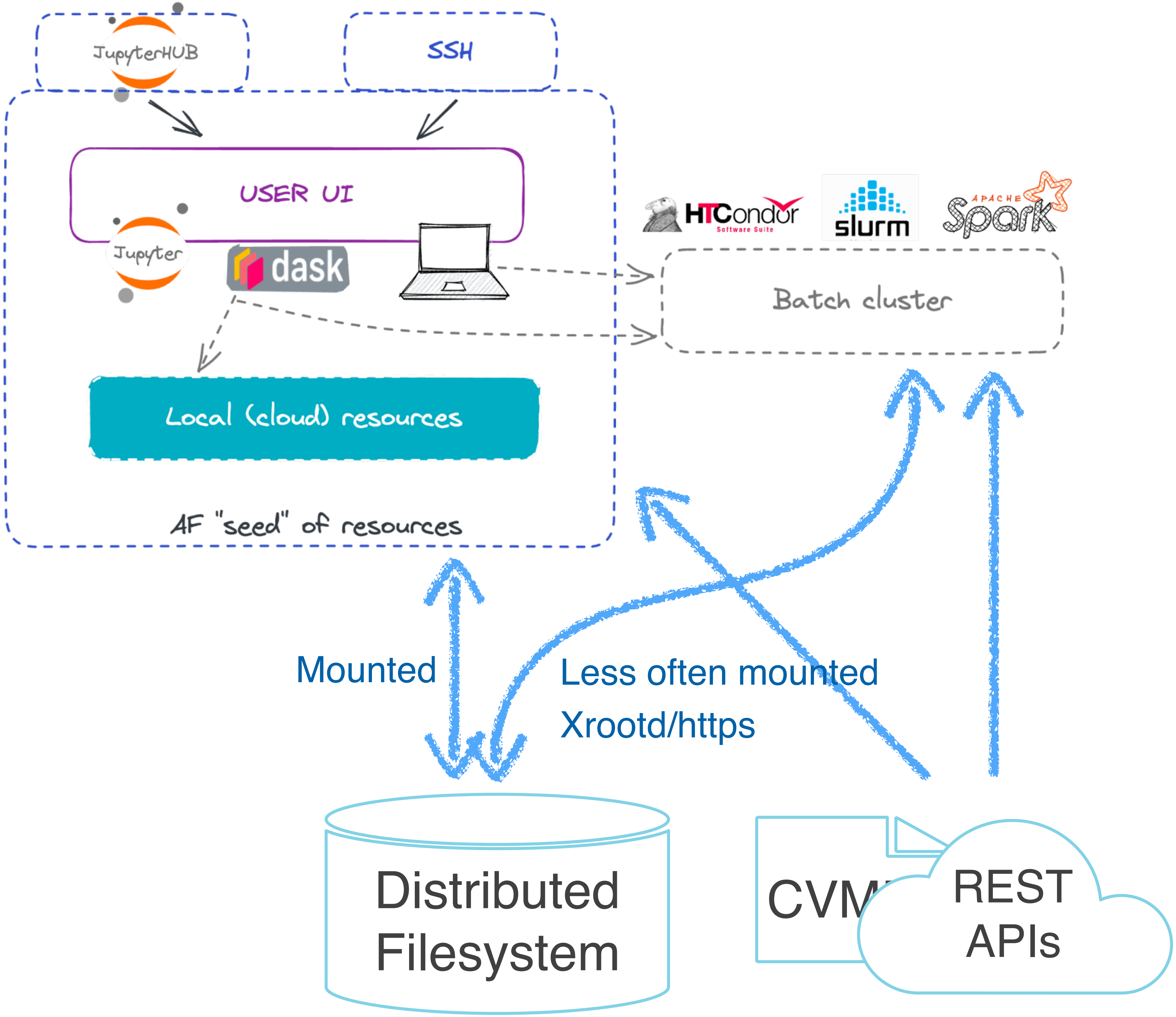
Less often mounted

Xrootd/https

Distributed Filesystem

CVMFS

Small data



Mounted

Less often mounted

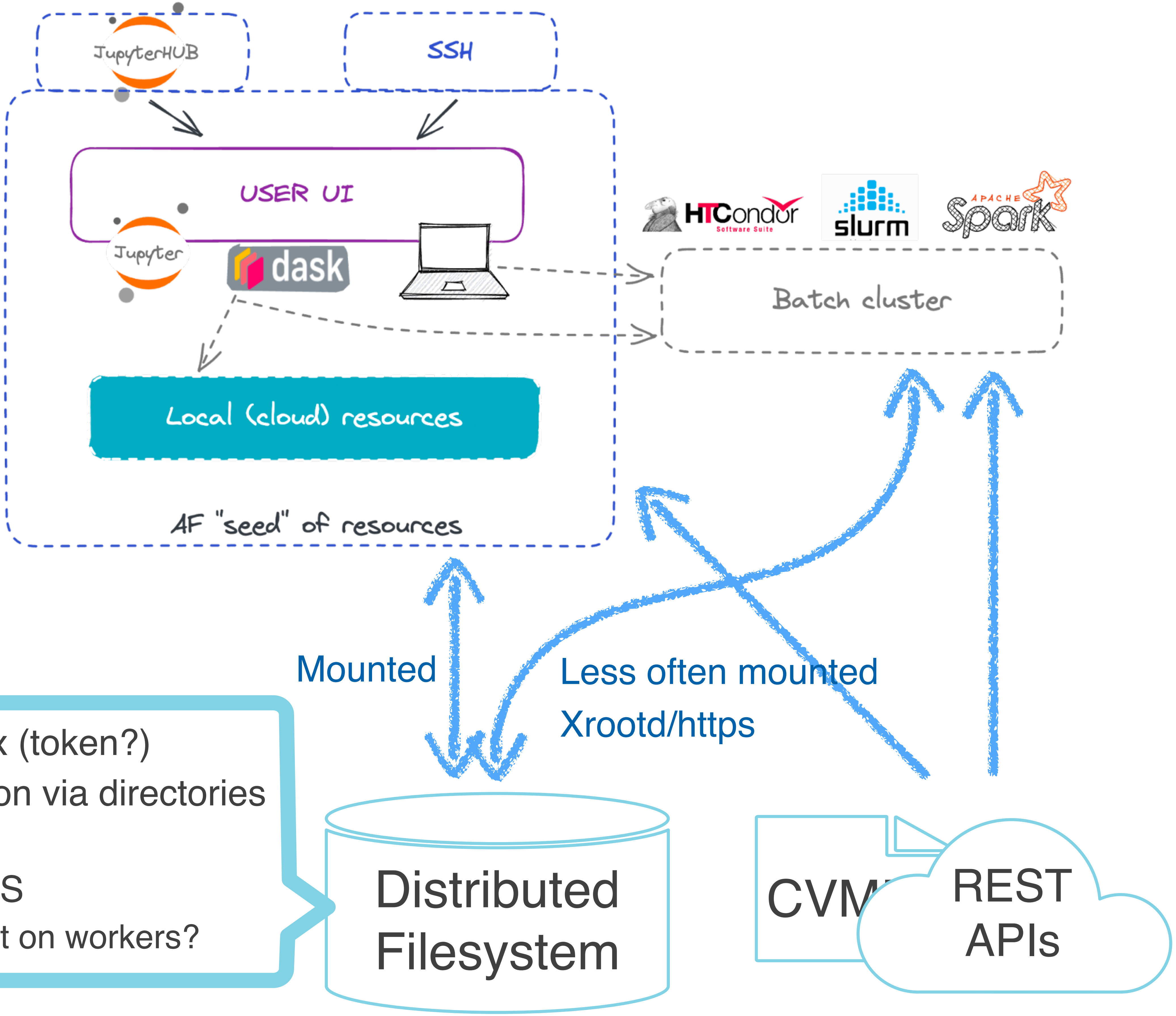
Xrootd/https

Distributed Filesystem

CVM

REST APIs

Small data



Mounted

Less often mounted

Xrootd/https

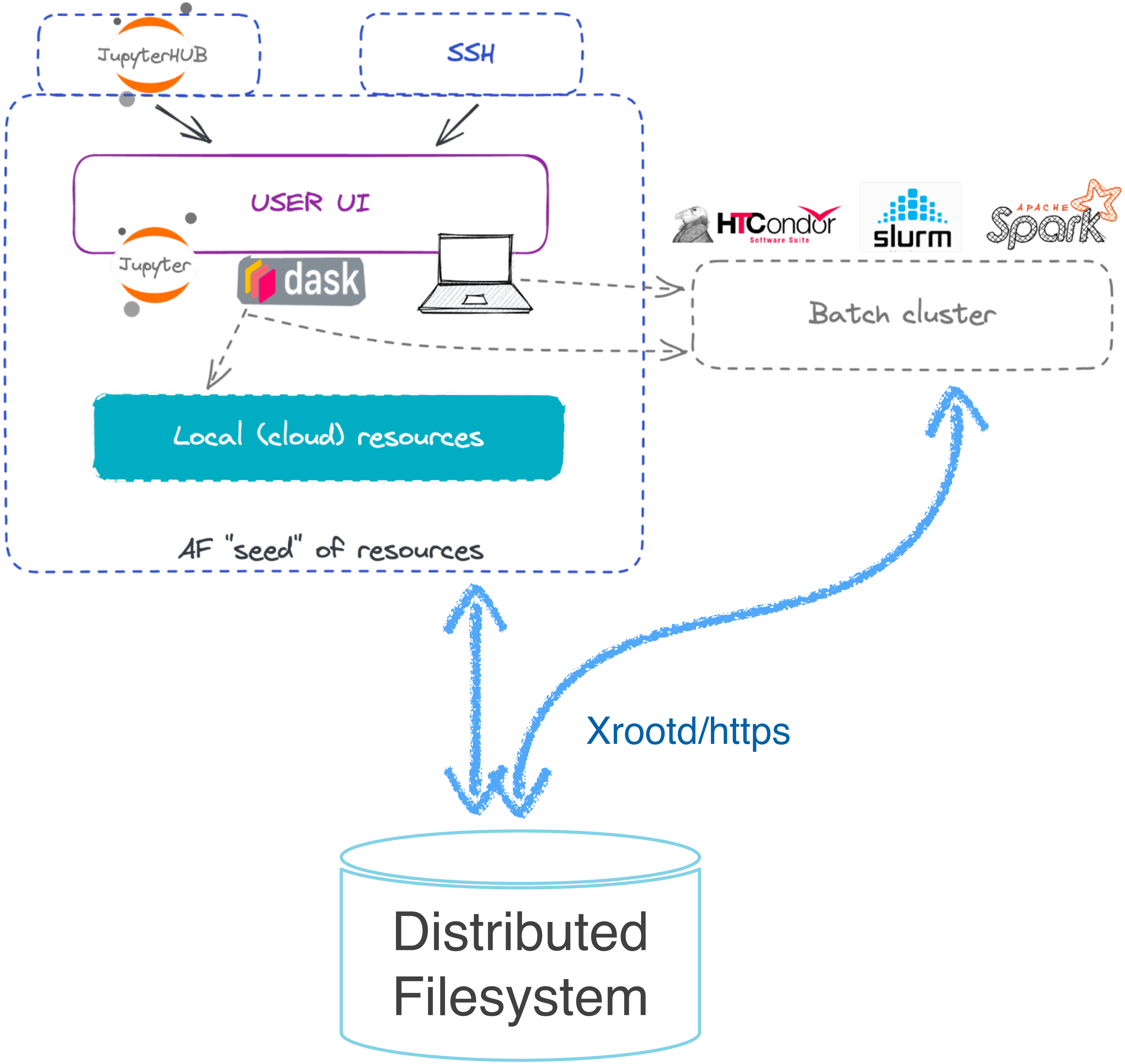
Distributed Filesystem

CVMFS

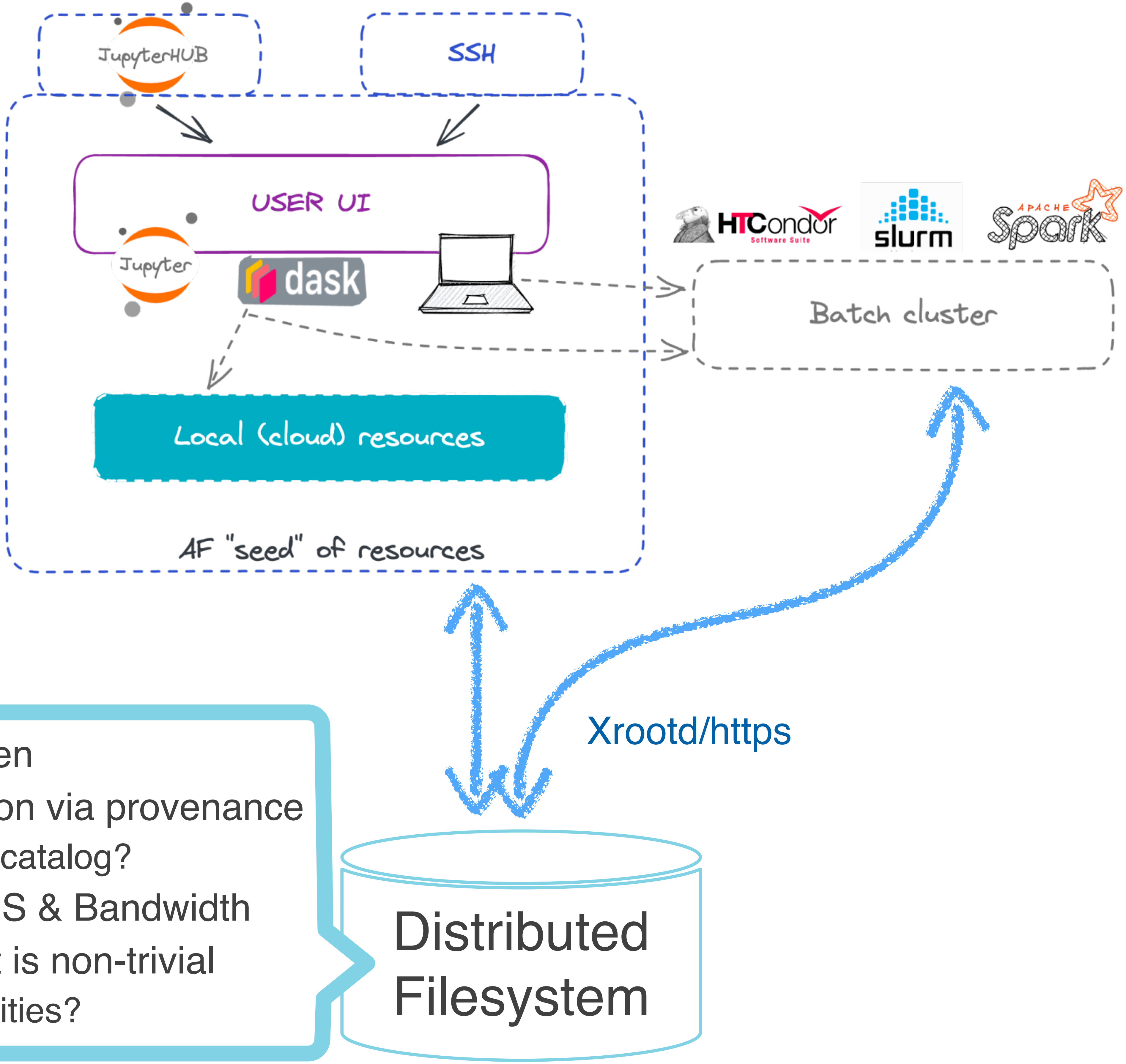
REST APIs

- Authorization: unix (token?)
- Logical organization via directories
  - Directory quotas
- Performance: IOPS
  - Read-only mount on workers?

Medium data


JupyterHUB
SSH
USER UI
Jupyter
dask
HTCondor Software Suite
slurm
APACHE Spark
Batch cluster
Local (cloud) resources
AF "seed" of resources
Xrootd/https
Distributed Filesystem

Medium data



JupyterHUB

SSH

USER UI

Jupyter

dask

HTCondor Software Suite

slurm

APACHE Spark

Batch cluster

Local (cloud) resources

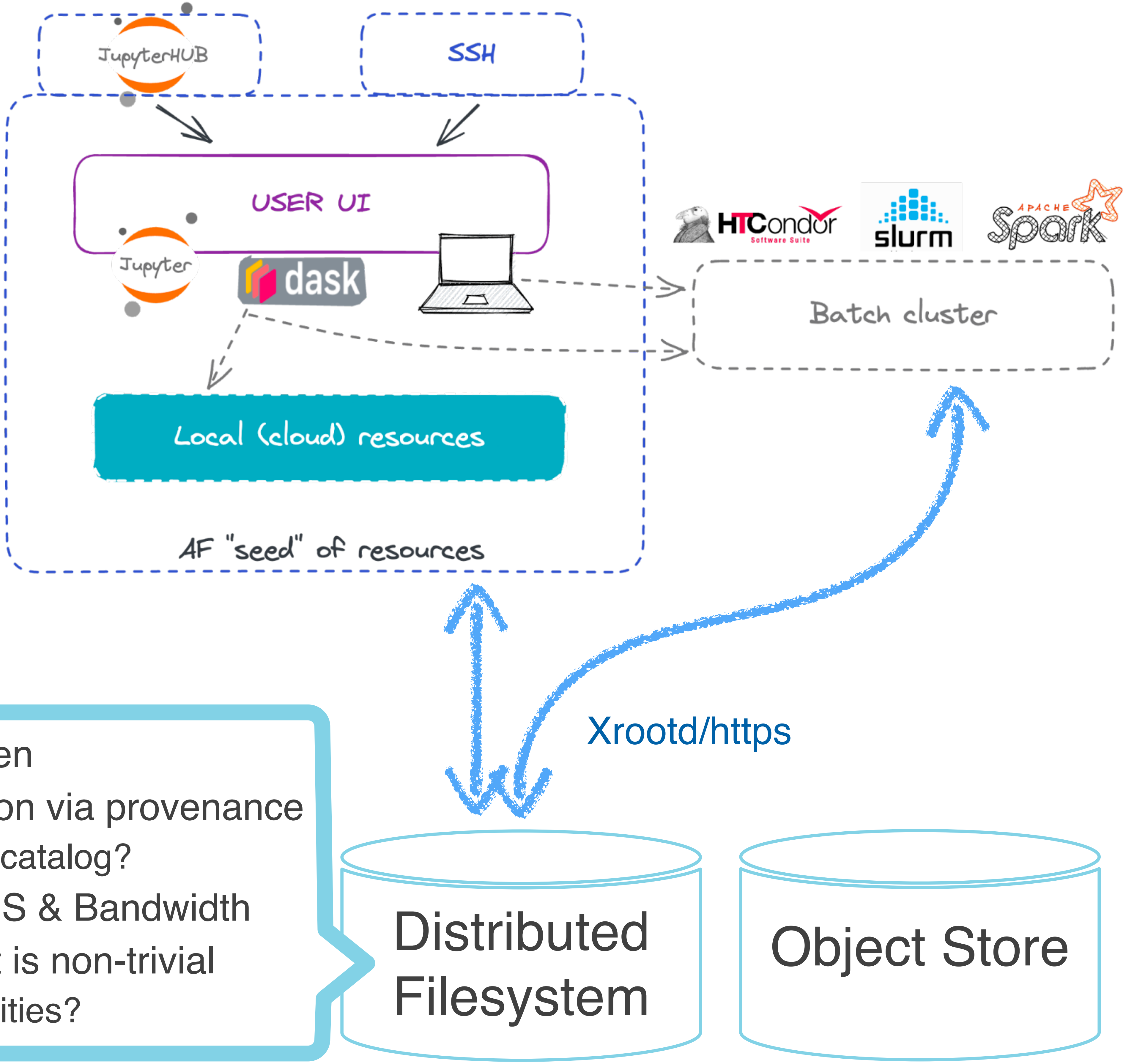
AF "seed" of resources

Xrootd/https

Distributed Filesystem

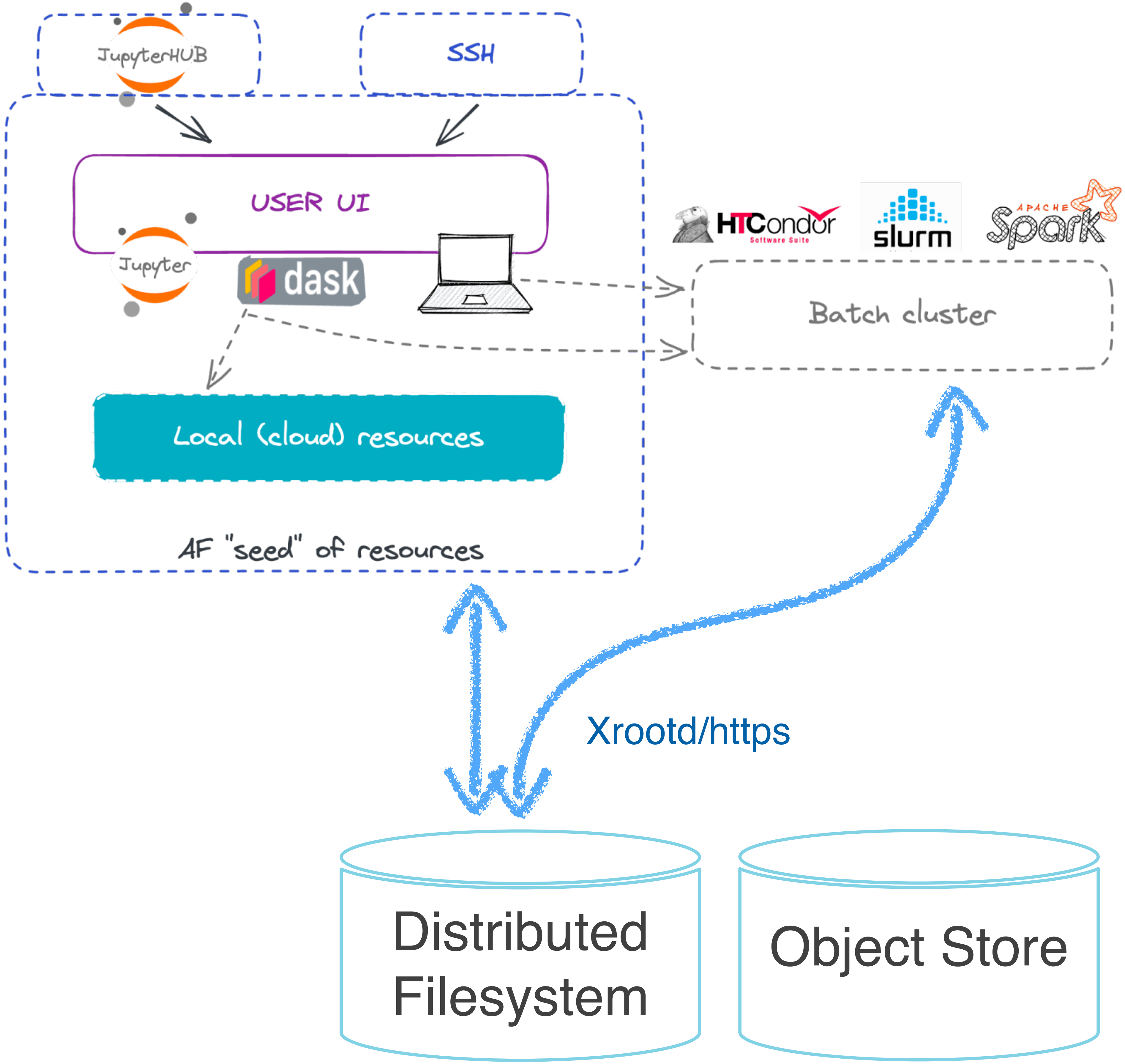
- Authorization: token
- Logical organization via provenance
  - Derived dataset catalog?
- Performance: IOPS & Bandwidth
- Lateral movement is non-trivial
  - TPC across facilities?

Medium data

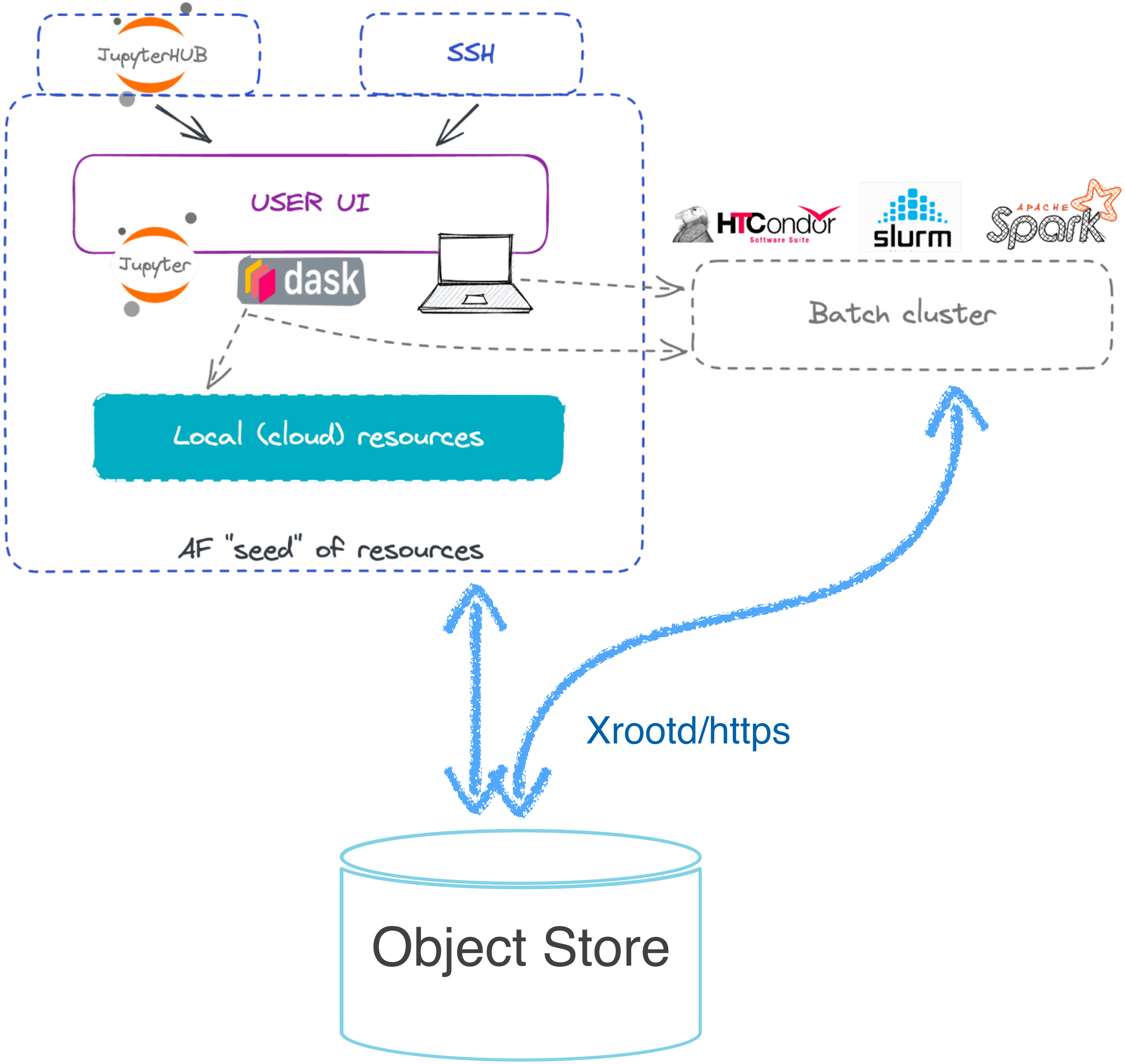


- Authorization: token
- Logical organization via provenance
  - Derived dataset catalog?
- Performance: IOPS & Bandwidth
- Lateral movement is non-trivial
  - TPC across facilities?

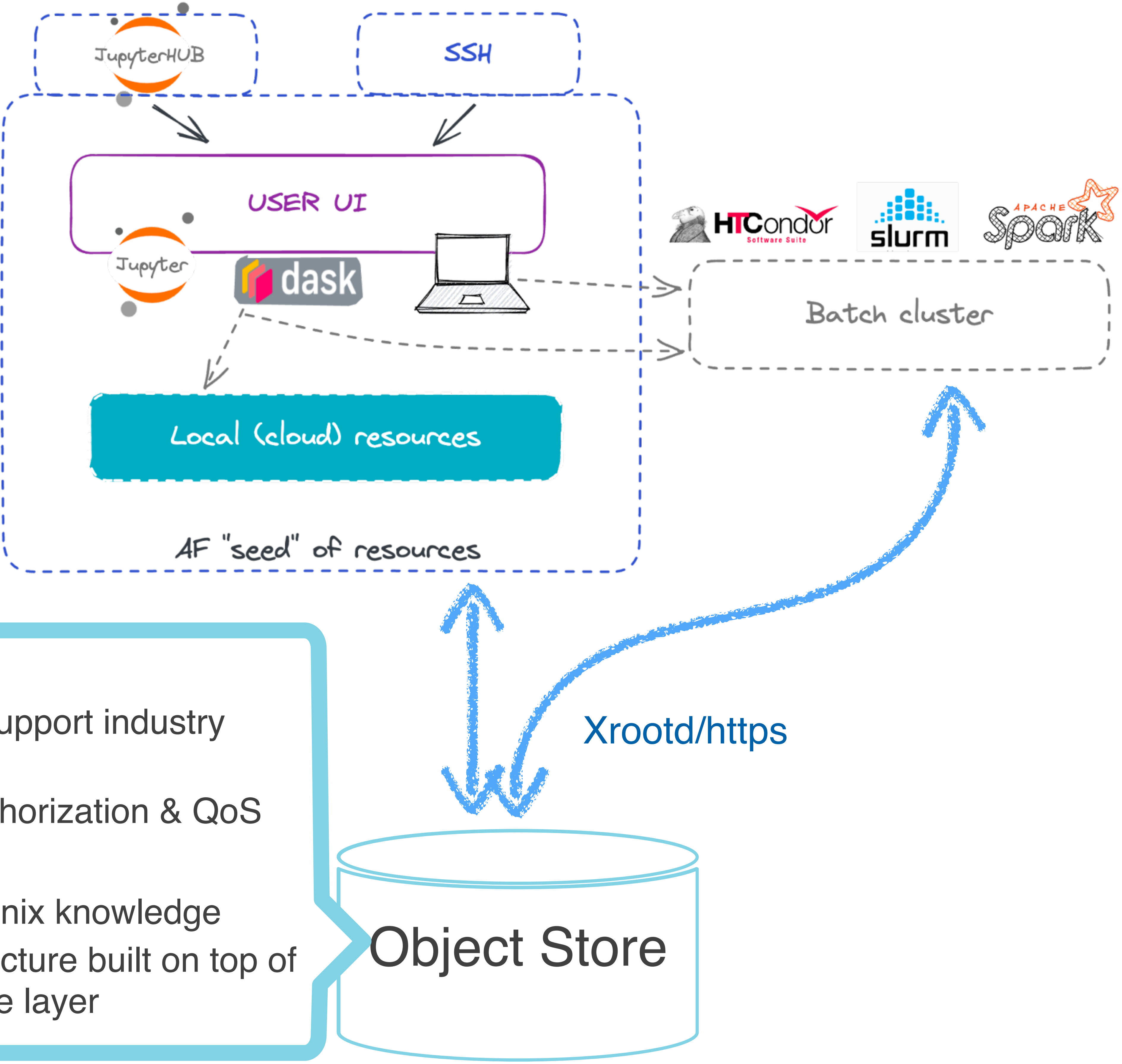
Medium data


JupyterHUB
SSH
USER UI
Jupyter
dask
HTCondor Software Suite
slurm
Apache Spark
Batch cluster
Local (cloud) resources
AF "seed" of resources
Xrootd/https
Distributed Filesystem
Object Store

Medium data

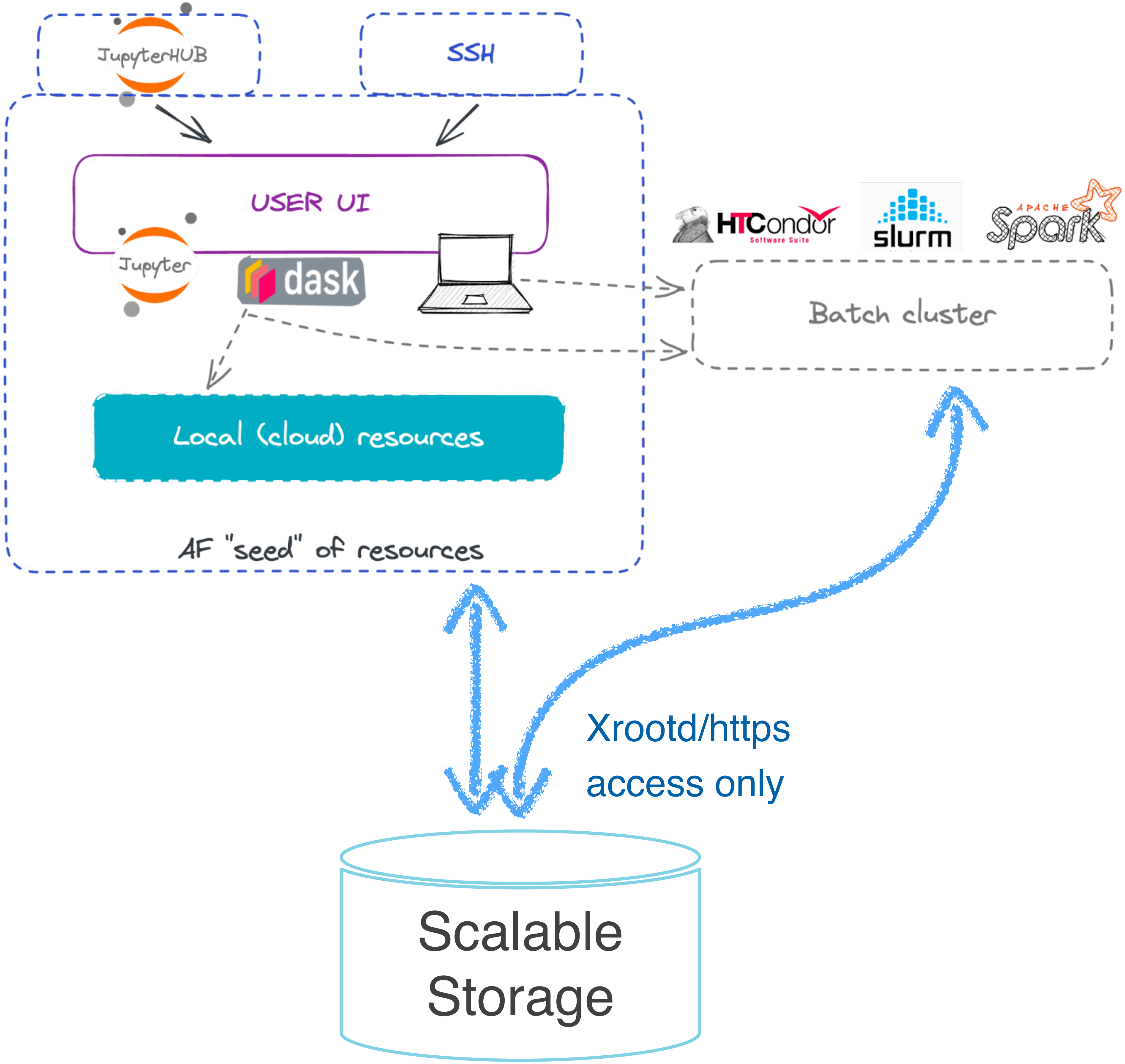

JupyterHUB
SSH
USER UI
Jupyter
dask
HTCondor
Software Suite
slurm
APACHE Spark
Batch cluster
Local (cloud) resources
AF "seed" of resources
Xrootd/https
Object Store

Medium data



- A few pros:
  - Cloud-friendly: support industry query platforms
  - More flexible authorization & QoS
- A few cons:
  - Fighting 50y of unix knowledge
  - Existing infrastructure built on top of POSIX(-ish) base layer

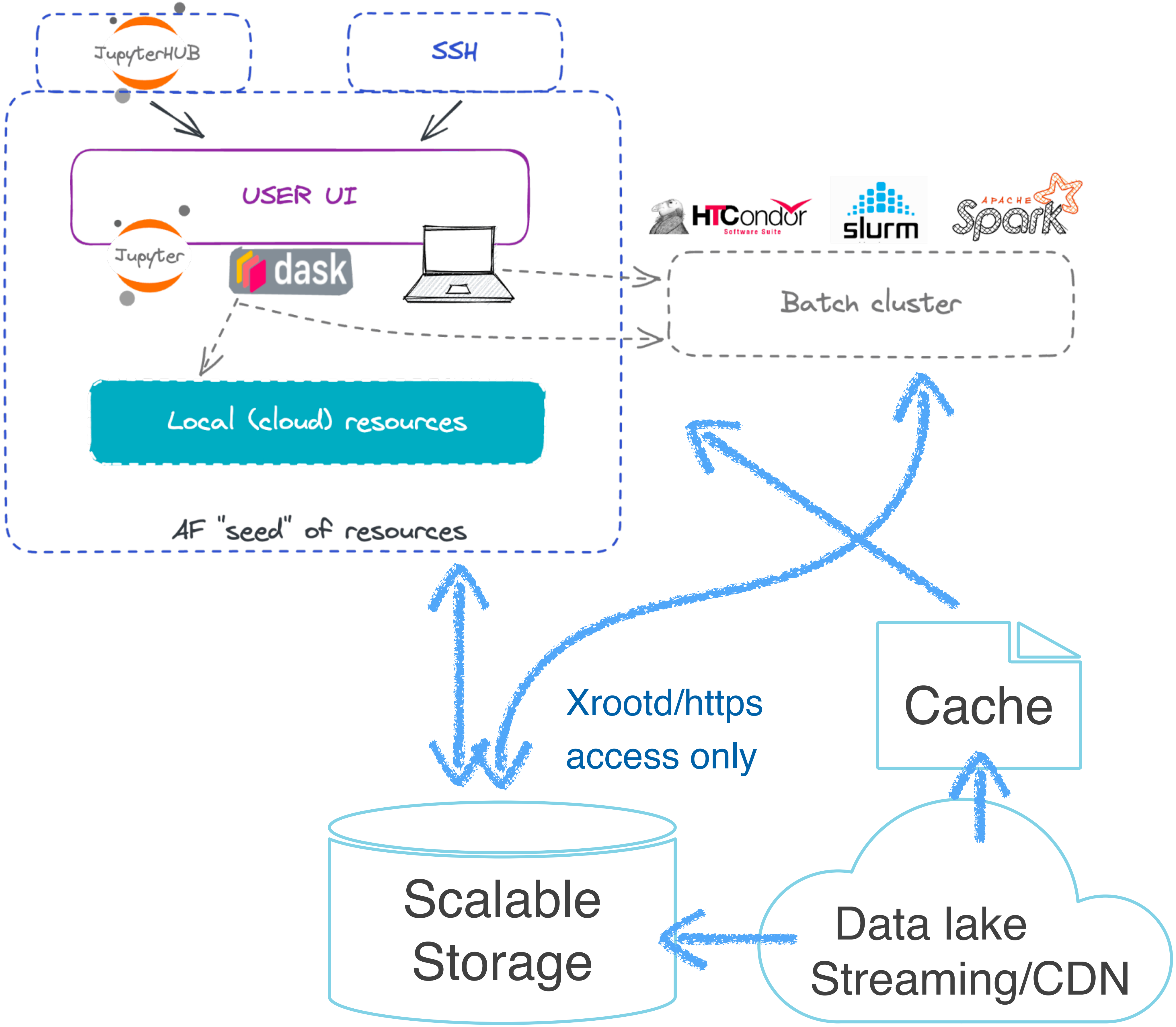
Large data



Xrootd/https access only

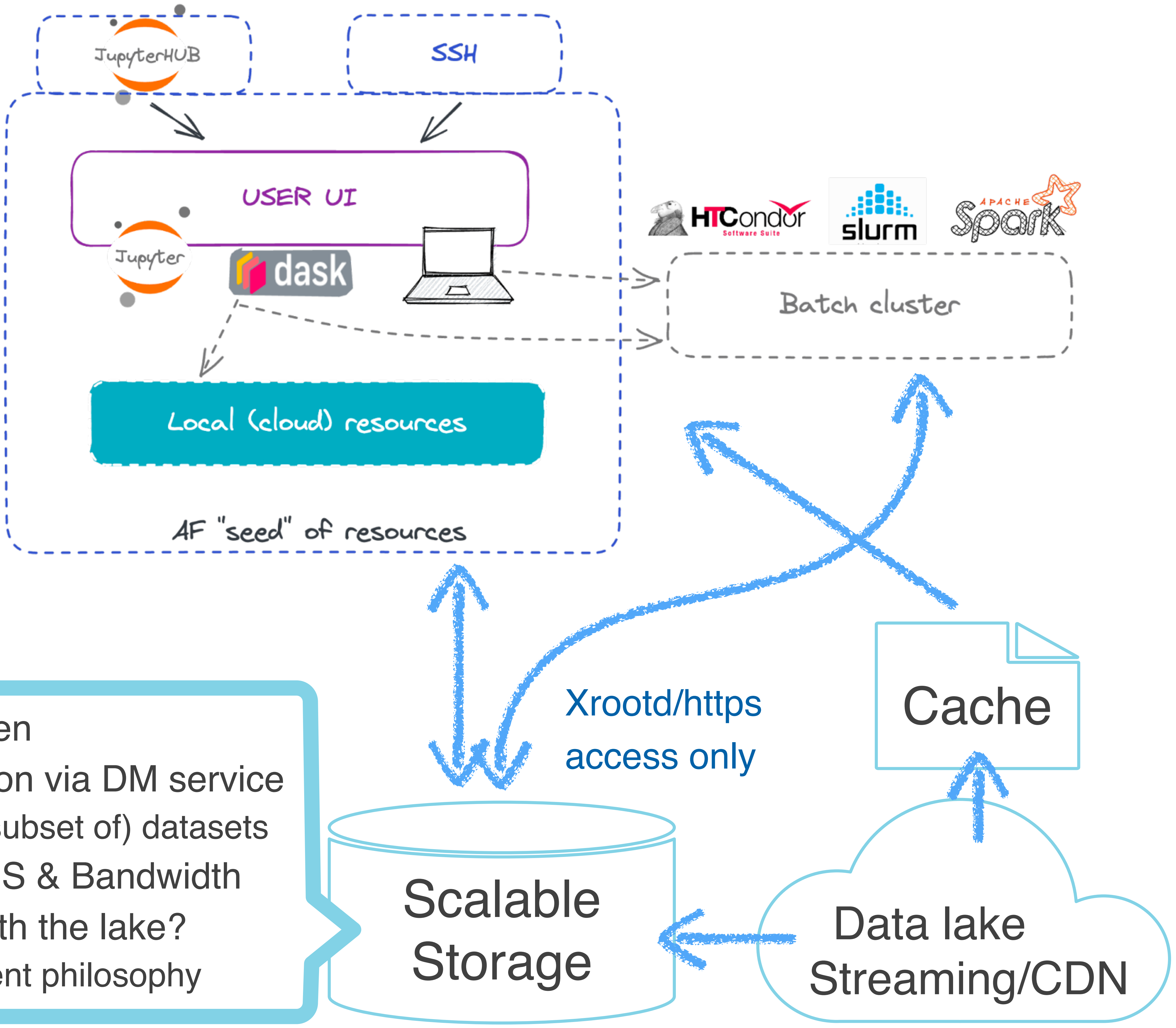
Scalable Storage

Large data



Xrootd/https access only

Large data



Xrootd/https access only

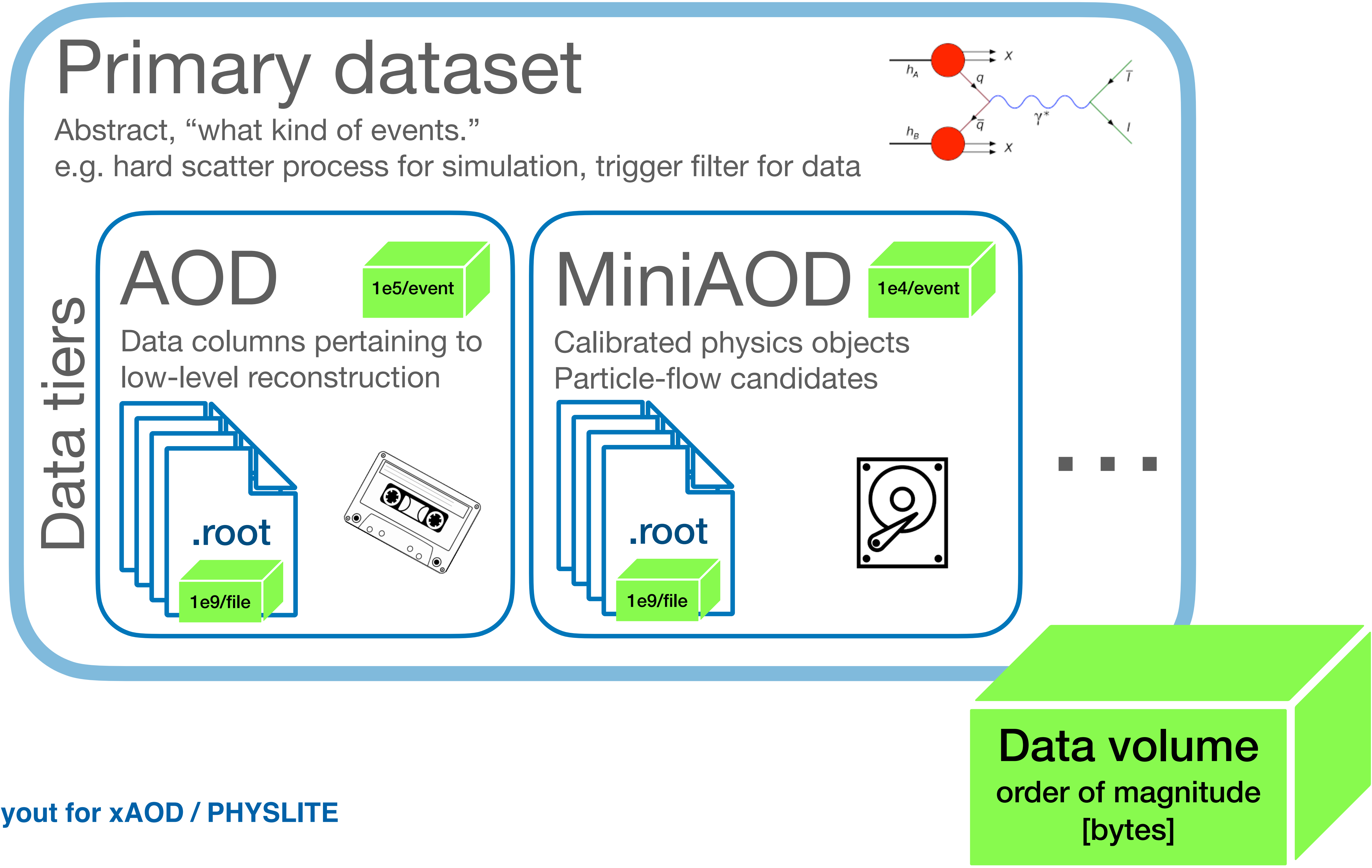
Cache

Scalable Storage

Data lake Streaming/CDN

- Authorization: token
- Logical organization via DM service
  - User *requests* (subset of) datasets
- Performance: IOPS & Bandwidth
- How to interact with the lake?
  - Data management philosophy

Data management philosophy



Primary dataset

Abstract, “what kind of events.”
e.g. hard scatter process for simulation, trigger filter for data

Data tiers

AOD

1e5/event

Data columns pertaining to low-level reconstruction

.root

1e9/file

MiniAOD

1e4/event

Calibrated physics objects
Particle-flow candidates

.root

1e9/file

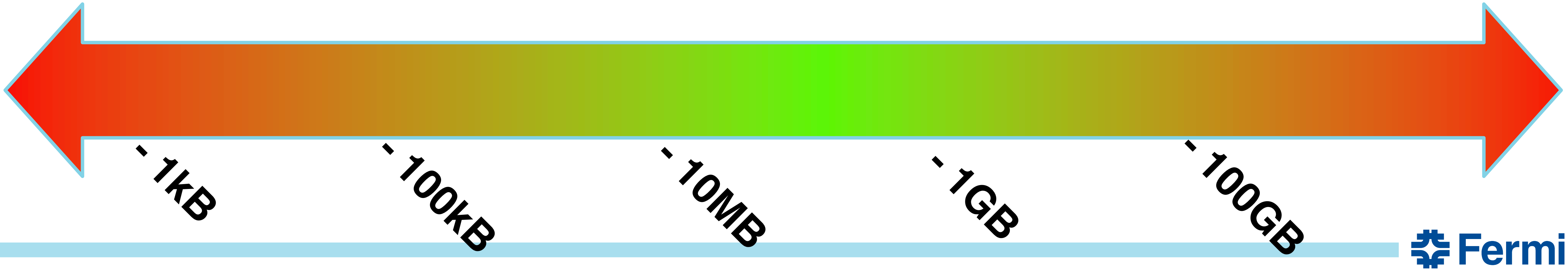
. . .

Data volume
order of magnitude
[bytes]

Similar layout for xAOD / PHYSLITE

Can we align unit of data access to unit of physics content?

- Dataset = list of 2-4 GB files, totaling 10GB-1PB. Why?

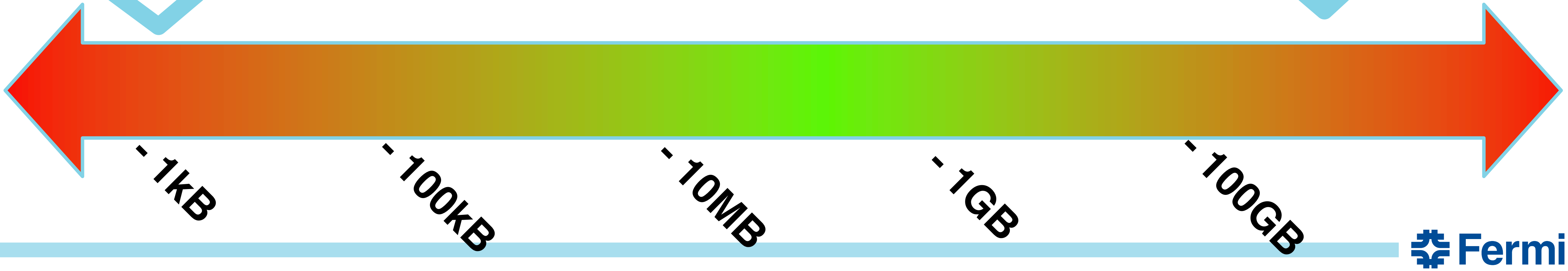


Can we align unit of data access to unit of physics content?

- Dataset = list of 2-4 GB files, totaling 10GB-1PB. Why?

- Lower limits:
  - IOPS!!
  - Erasure-code block size ~ 4-16kB
  - Catalog/filesystem overhead ~ 10MB-1GB?

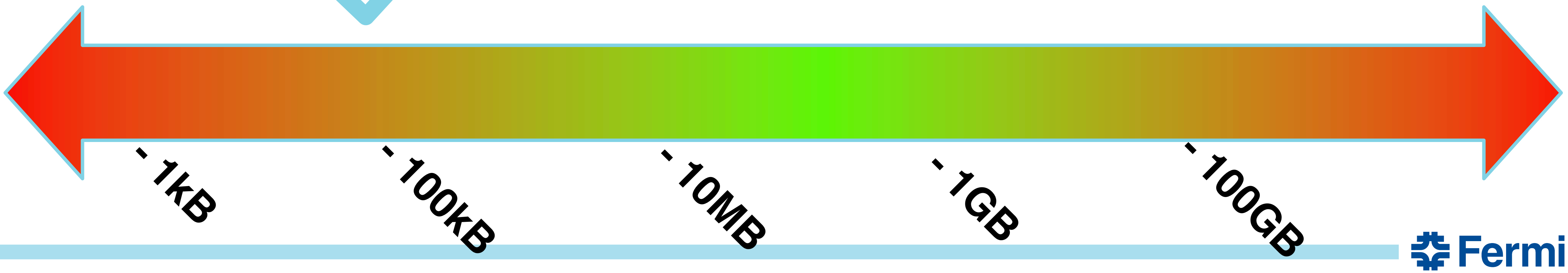
- Upper limits:
  - Third party copy timeout ~ 20 GB
  - Tape cartridge ~ 10 TB



Can we align unit of data access to unit of physics content?

- Dataset = list of 2-4 GB files, totaling 10GB-1PB. Why?

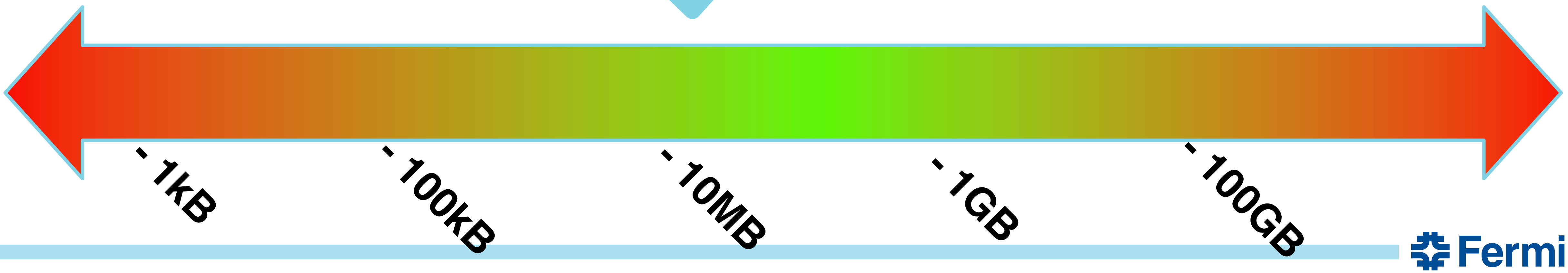
- TBasket / Page sizes ~ 10-100kB
  - This is physics-relevant
  - One float column for O(100k) events
  - One ragged column for O(10k) events
- Motivation for byte-range xcache



Can we align unit of data access to unit of physics content?

- Dataset = list of 2-4 GB files, totaling 10GB-1PB. Why?

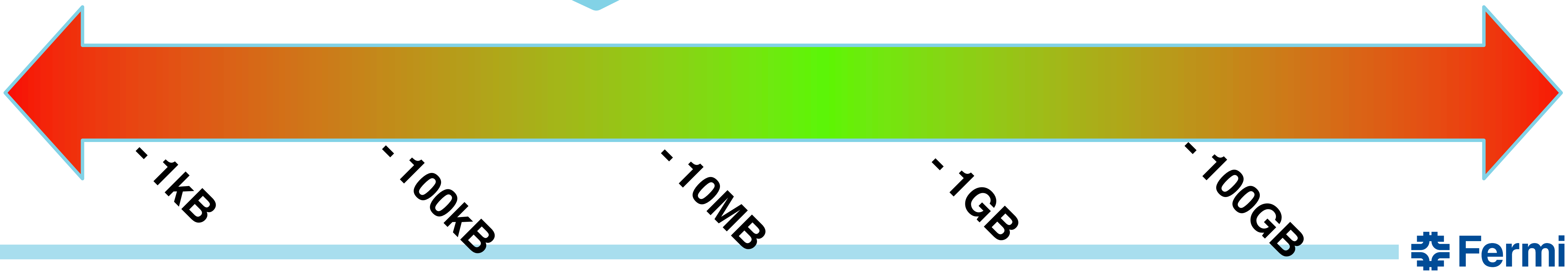
- Cluster size ~ 10MB
  - All columns pertaining to same group of events
- Good target for read-ahead buffer size
- Do we want to cluster *all* columns though?
  - Typical analysis accesses 10-50%
  - How will column joins be performed?



Can we align unit of data access to unit of physics content?

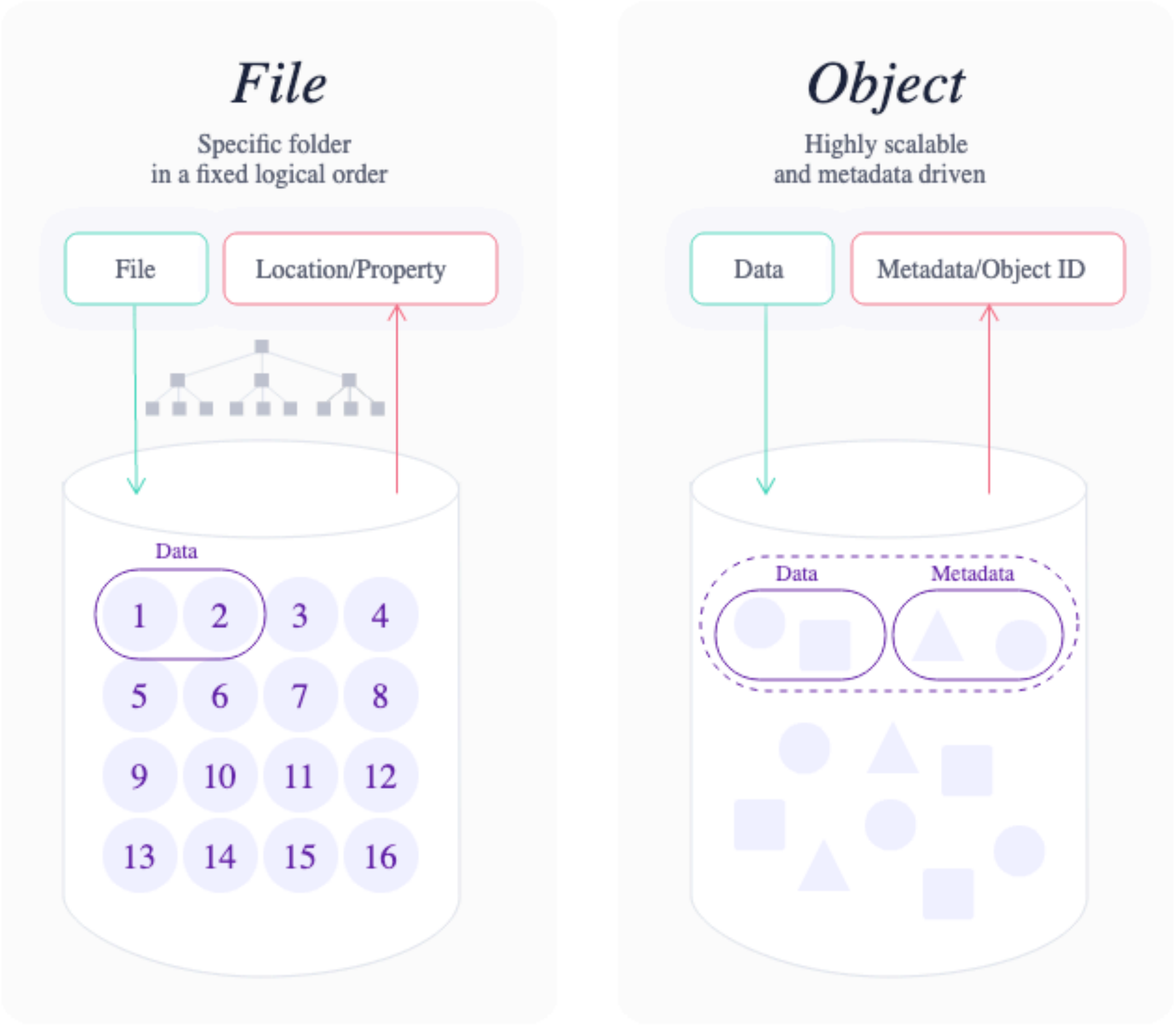
- Dataset = list of 2-4 GB files, totaling 10GB-1PB. Why?

- Sweet spot for access ~ 1MB
  - Few ragged columns for O(10k) events?
  - Many columns for O(1k) events?
  - Do we want small # events per unit?
- Whole-unit cache ➜ off-shelf solutions
- Catalog challenge: need indirection



Object store vs. filesystem

- Traditional data storage technology: distributed filesystem
  - e.g. NFS, EOS, dCache, Lustre, HDFS*, …
  - Often with remote access protocol (xrootd)
  - Files are concurrently read/writeable
- Popular new-ish technology: object store
  - Native remote access (http)
  - Objects are immutable (overwrite possible)



attrib

Breaking down the ROOT file

- Essentially storing (+ moving) smaller units
  - This is usually a bad thing

Intermodal container

?

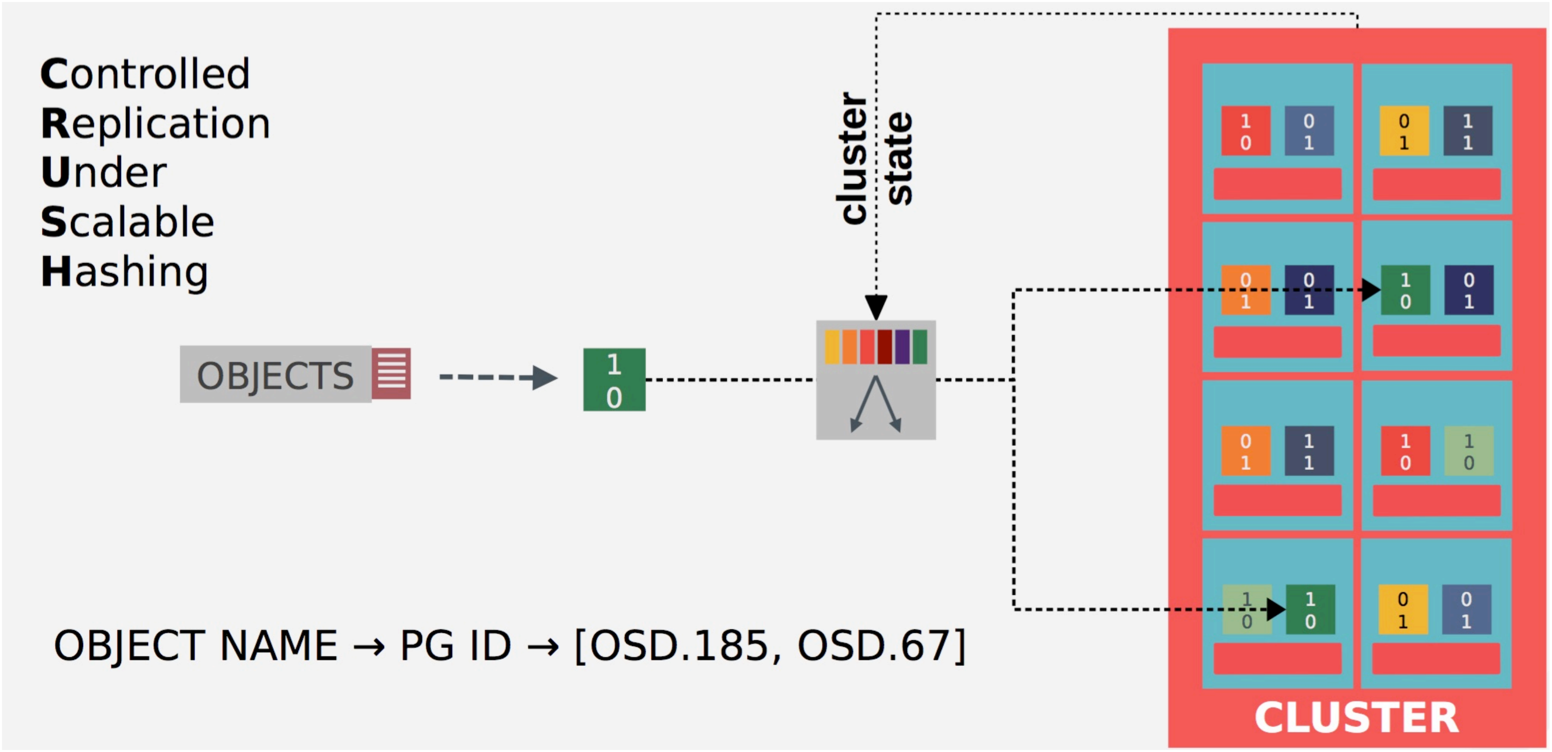
Break-bulk cargo

Breaking down the ROOT file

- Essentially storing (+ moving) smaller units
  - This is usually a bad thing
- Calculated placement
  - Like a hash, client-side
  - Downside: cluster state change causes reshuffle
    - Consistent hashing to minimize movement

Intermodal container

? ↓



Controlled
Replication
Under
Scalable
Hashing

OBJECTS

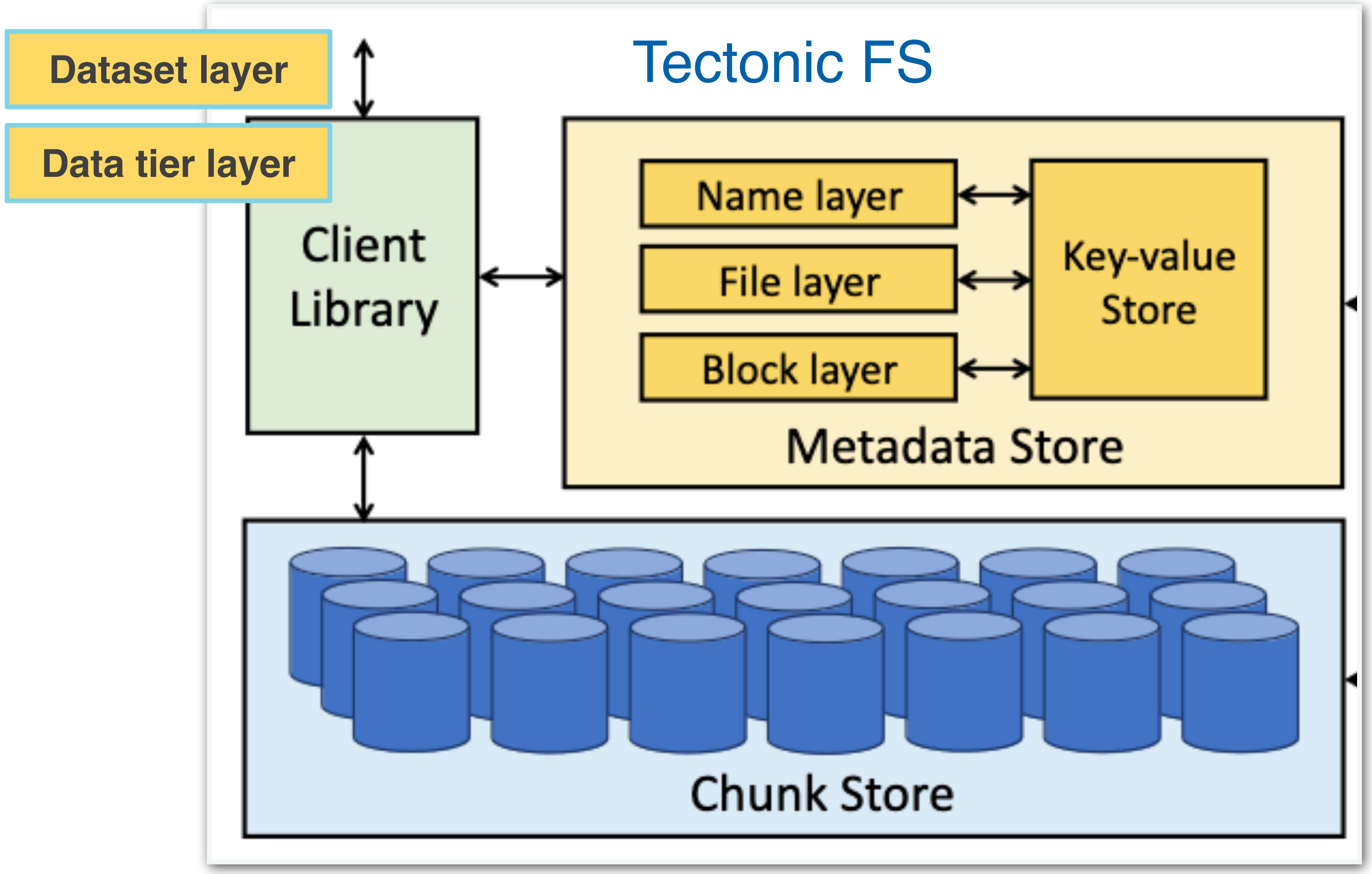
cluster state

CLUSTER

OBJECT NAME → PG ID → [OSD.185, OSD.67]

Break-bulk cargo

Higher levels of indirection



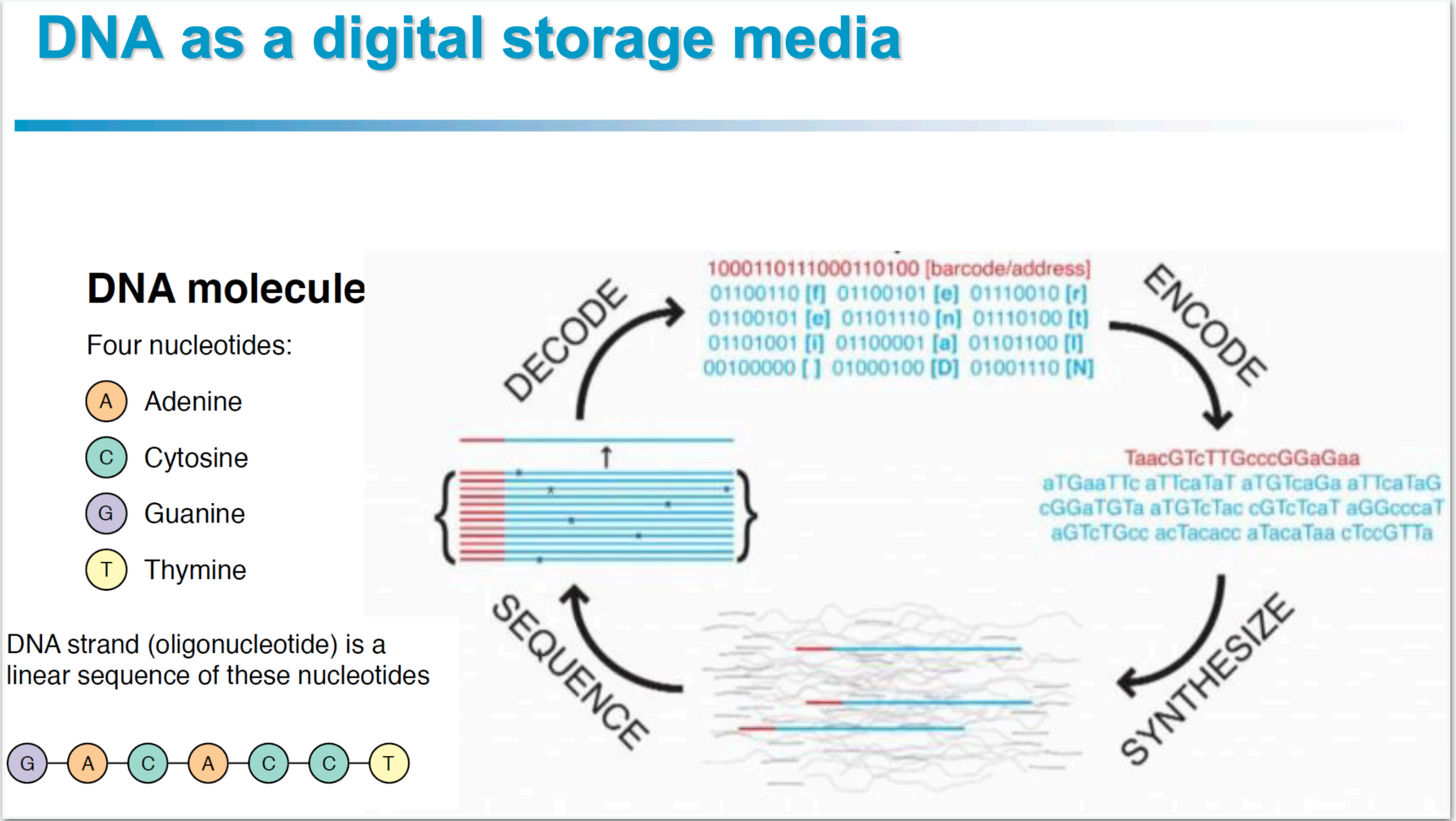
For intermediate data, (ab)using POSIX filesystem as an implicit data catalog.

Bring Rucio to intermediate data? Does Rucio have sufficient indirection layers?

How do we enable a “facility grid” (cross-facility namespace for intermediate data)

A totally different storage technology

Appuswamy



DNA as a digital storage media

DNA molecule

Four nucleotides:

- A Adenine
- C Cytosine
- G Guanine
- T Thymine

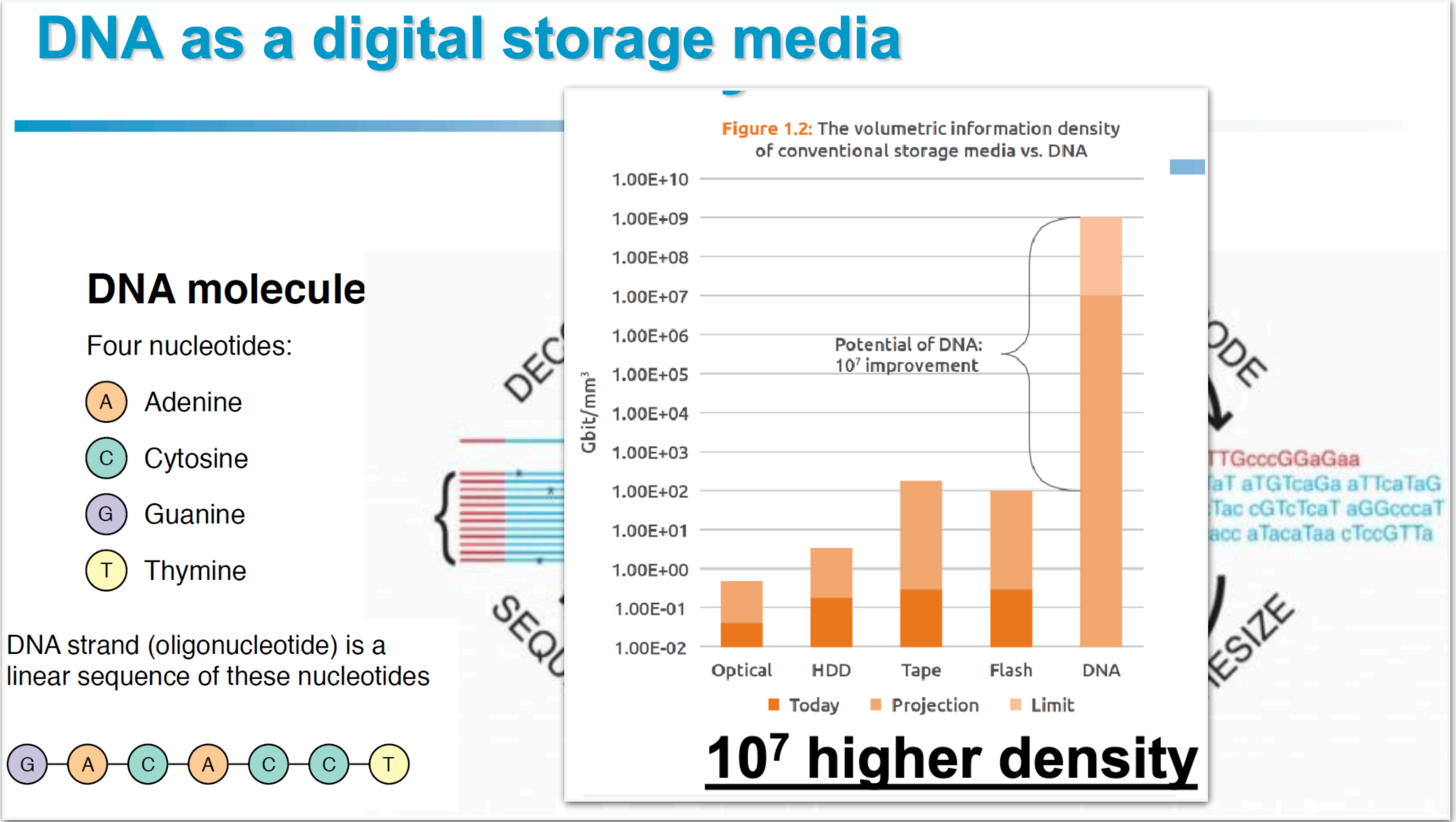
DNA strand (oligonucleotide) is a linear sequence of these nucleotides

10001101111000110100 [barcode/address]
01100110 [f] 01100101 [e] 01110010 [r]
01100101 [e] 01101110 [n] 01110100 [t]
01101001 [i] 01100001 [a] 01101100 [l]
00100000 [ ] 01000100 [D] 01001110 [N]

TaacGTcTTGcccGGaGaa
aTGaaTTc aTTcaTaT aTGTcaGa aTTcaTaG
cGGaTGTa aTGTcTac cGTcTcaT aGGcccaT
aGTcTGcc acTacacc aTacaTaa cTccGTTa

A totally different storage technology

Appuswamy



DNA as a digital storage media

DNA molecule

Four nucleotides:

- A Adenine
- C Cytosine
- G Guanine
- T Thymine

DNA strand (oligonucleotide) is a linear sequence of these nucleotides

G — A — C — A — C — C — T

Figure 1.2: The volumetric information density of conventional storage media vs. DNA

10^7 higher density

Conclusion

- Data management for large HEP experiments is a complex topic
- Physics workflows drive requirements
- Always keep an eye out for new technologies