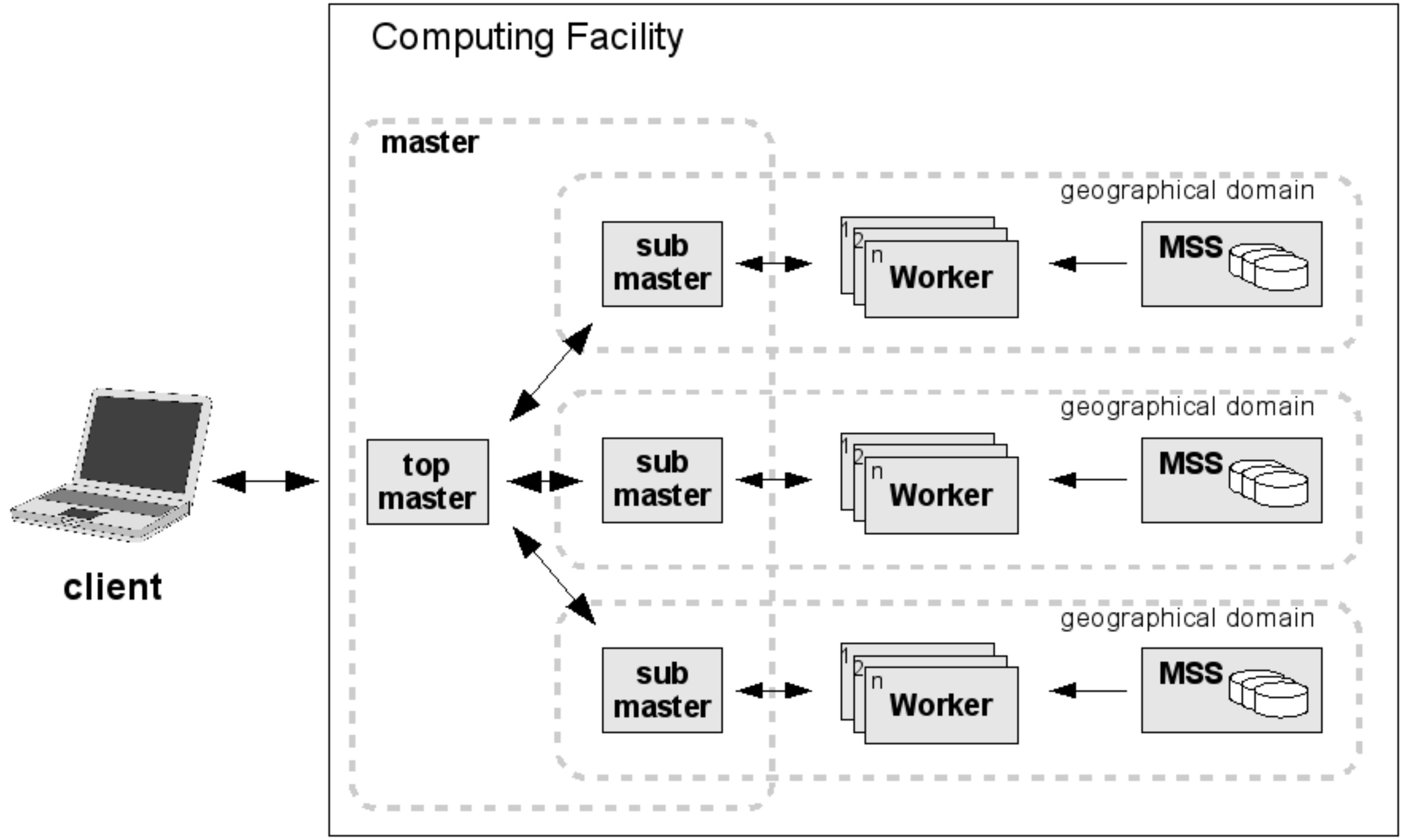
PROOF
in a federated world

Gerardo Ganis, PH-SFT, CERN

PROOF in a nut shell

- Interactive coordination of distributed ROOT sessions running in parallel
- Multi-process parallelism for *ideally parallel* tasks
- Pull architecture (dynamic load-balancing)
- ROOT User Interface

PROOF Architecture



Case addressed by PROOF

- End-User analysis in primis
- Tier 3, Department Analysis Farms, multi-core desktops
- Problematics (may be) different from Tier2's

PROOF and Federation

- Usage of a Federated Data Store
  - (Just) another Mass Storage
- Federated use of Computing Resources
  - Exploiting multi-tier architecture

PROOF and data stores

- Main bottleneck in data analysis: I/O
- PROOF attempts to improve the rate of data processing by
  - Getting closer to data
    - Assumption: data movements are the expensive part
  - Increasing aggregated CPU, network, memory
    - Effective way to increase I/O bandwidth

PROOF engine

- *Packet*: unit of work (file, first-to-last)
- The *packetizer* chooses the best packet to be assigned to a worker at a given time
  - Or the best worker to process a given packet
- *Data-driven*: based on locality, number of workers accessing the same server, ...

Federated data stores

- "Any Data, Any Time, Any Where"
- Data *delocalization*
  - Hide the real location of data
    - E.g. through a mount point
  - Provide a common entry-point but keeping visible the structure behind
    - E.g. standard xroot

Prefetching, caching ...

... essential for performance

- Coarse grained
  - Stage files on local scratch storage
- Fine grained
  - Surgical read-ahead
    - Exploit knowledge about what to process next
  - Cache prefetched chunks locally

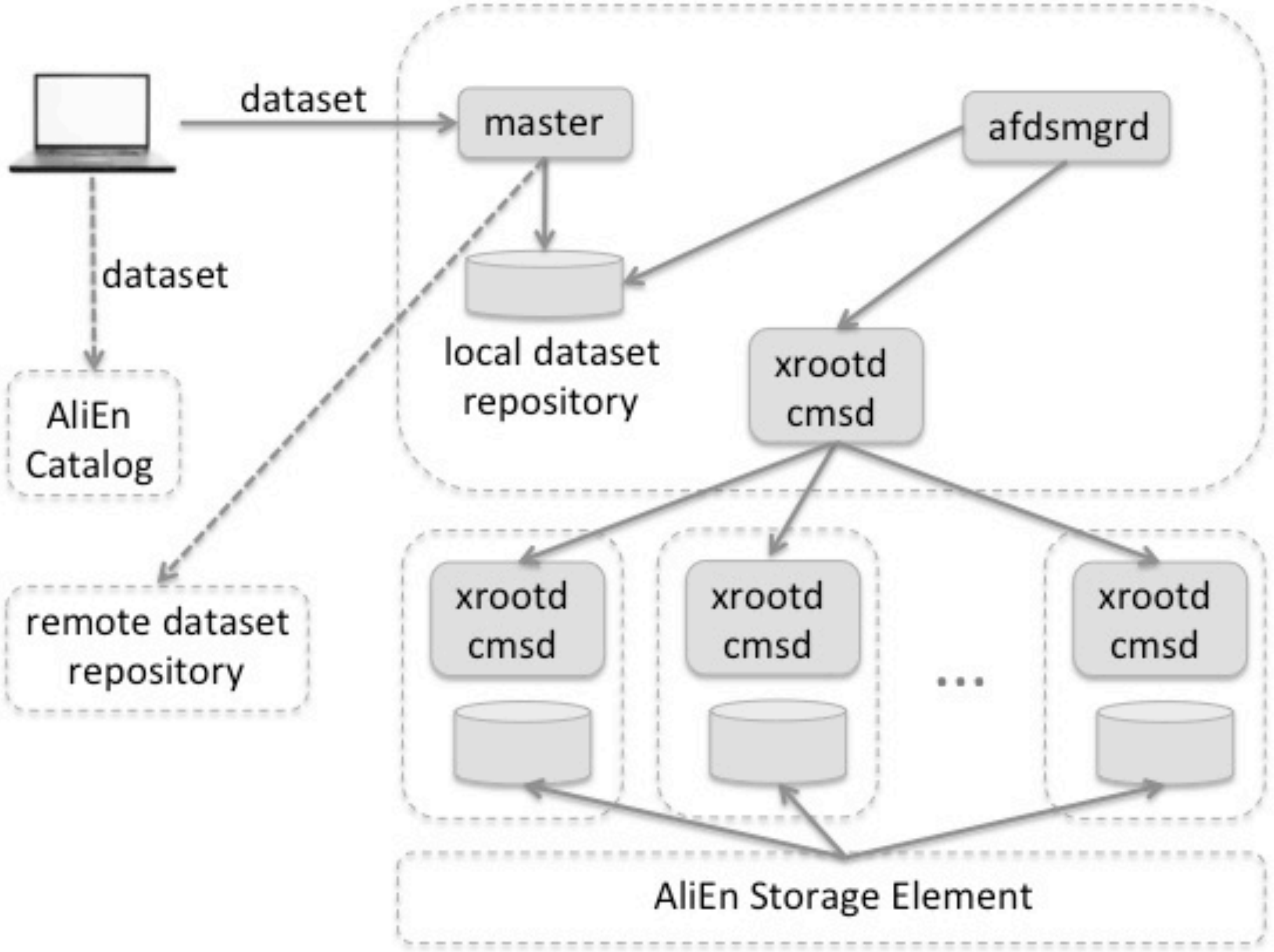
Existing example: AAF

- AAF: ALICE Analysis Facilities
- Cluster of machines with a sizeable amount of local (scratch) storage
- Files copied on demand from AliEn
- Positive experience

Dataset management needed!

- Users ask for a given dataset, i.e. a named collection of AliEn files
- An external daemon watches the request and make sure that the files are available
- When space is needed less used files removed by the storage manager (xrootd)

Datasets at AAF



File chunk caching

- Populate a local cache with the prefetched chunks, used later or by other processes
  - Feature recently introduced in ROOT SEE ROOT TALK
- Alternative way to populate the cache
  - Lower latency, optimized transfers, even distribution
    - May result in improved analysis performance
  - PROOF needs dedicated packetizer

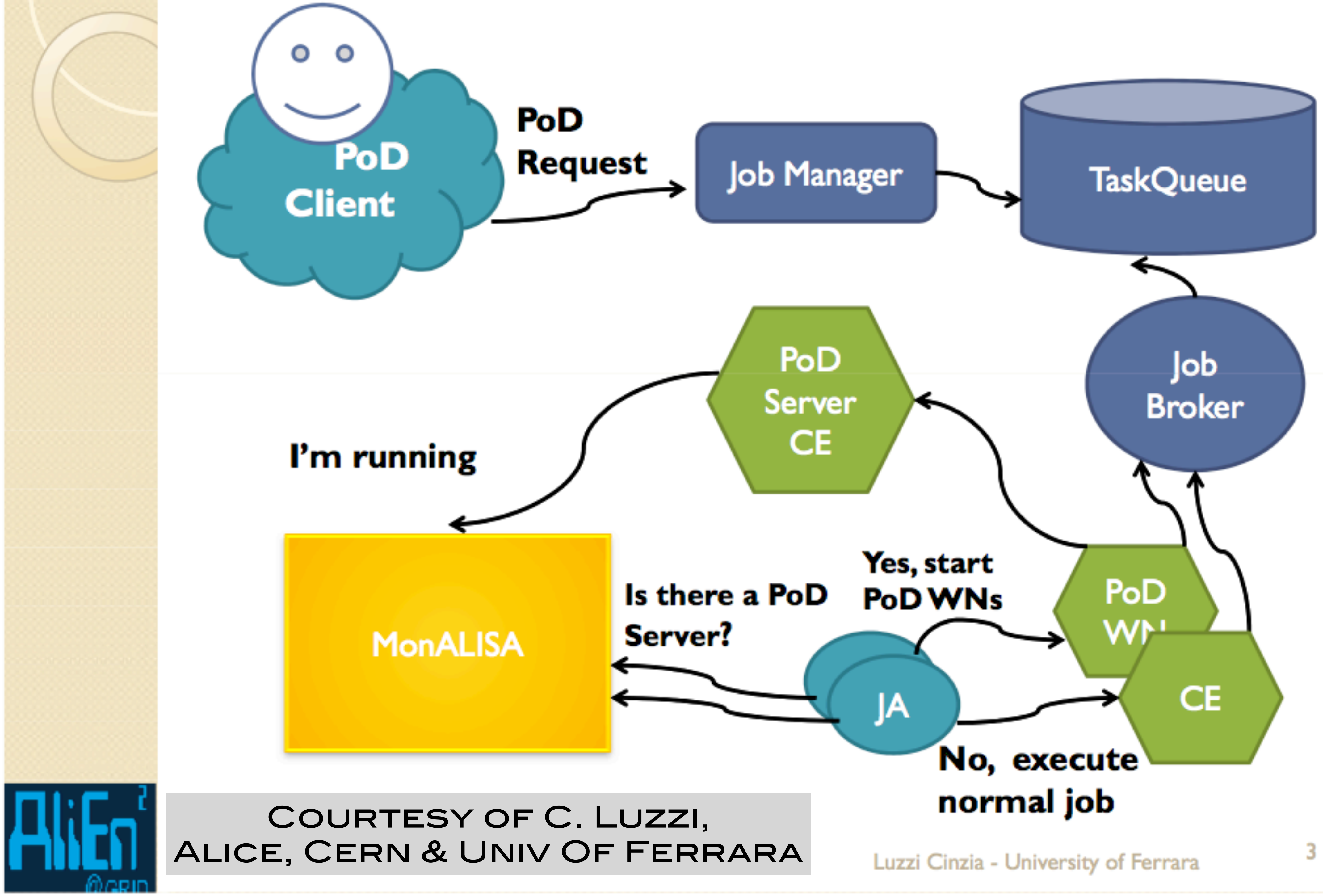
PROOF on ALiEn

- AliEn: data store + computing resources
- Exploit resource management to start workers using Proof-On-Demand(1)
- Eventually use AliEn info about file location as input to the job broker
- In progress

1) A.Manafov, http://pod.gsi.de/

PoD on AliEn

FOR ILLUSTRATION ONLY
WORK IN PROGRESS

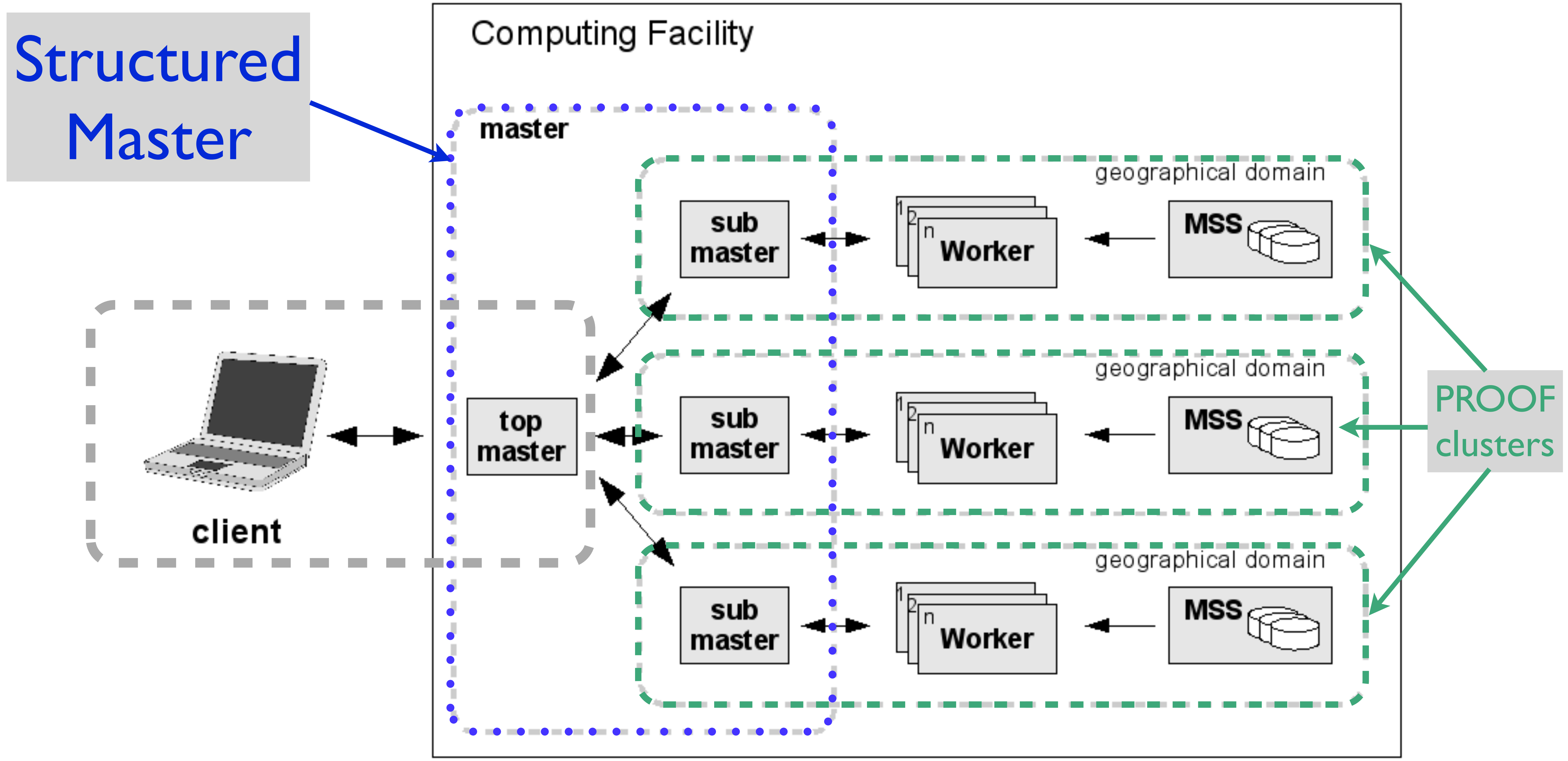


COURTESY OF C. LUZZI,
ALICE, CERN & UNIV OF FERRARA

Federating processing resources

- An old idea
    - Federation of PROOF clusters demo'ed (and used at Phobos) in the early times (2003, ...)
- Exploits multi-tier master architecture
    - Common view via the top master, single entry point into the system

Multi-Tier Architecture



Basic idea

- Driven by the *data locality paradigm* applied to geographically distributed sets of data
  - *Step I*: setup a PROOF cluster around each data set
  - *Step II*: connect together the clusters with a single entry point

Enforcing data locality

- *Data locality was enforced*
  - No access across clusters
- Implications (or simplifications)
  - Static configuration
  - Basic dataset management
  - No load balancing across clusters

Federating existing AFs

- Data locality requirement relaxed
  - AF can read data from a federated mass storage including other AFs
- Federation facilitates unified access to larger set of resources
- (Should) improve overall performance

Federating AFs (cnt'd)

- Global dataset management
  - Knowledge of what exists already on the single cluster for overall optimization
- Load-balancing across clusters
  - Requires super-packetizer
- ...

Federation in PROOF improves scalability

- PROOF packetizer is serial
  - Deviations from linearity above ~200 workers
- Using sub-masters scalability breakdown pushed up (Nx)
- Dynamic sub-clustering of large farms
  - E.g. dynamic clusters on Grids, Clouds, ...

Summary

- PROOF as client of federated data stores
  - Just another mass storage
  - Good prefetching / caching for performance
- Federating computing resources w/ PROOF
  - Unify access to resources
  - Way to improve scalability and to optimize performance

Thanks!

Questions?