Interactive Data Analysis with PROOF

Bleeding Edge Physics with Bleeding Edge Computing

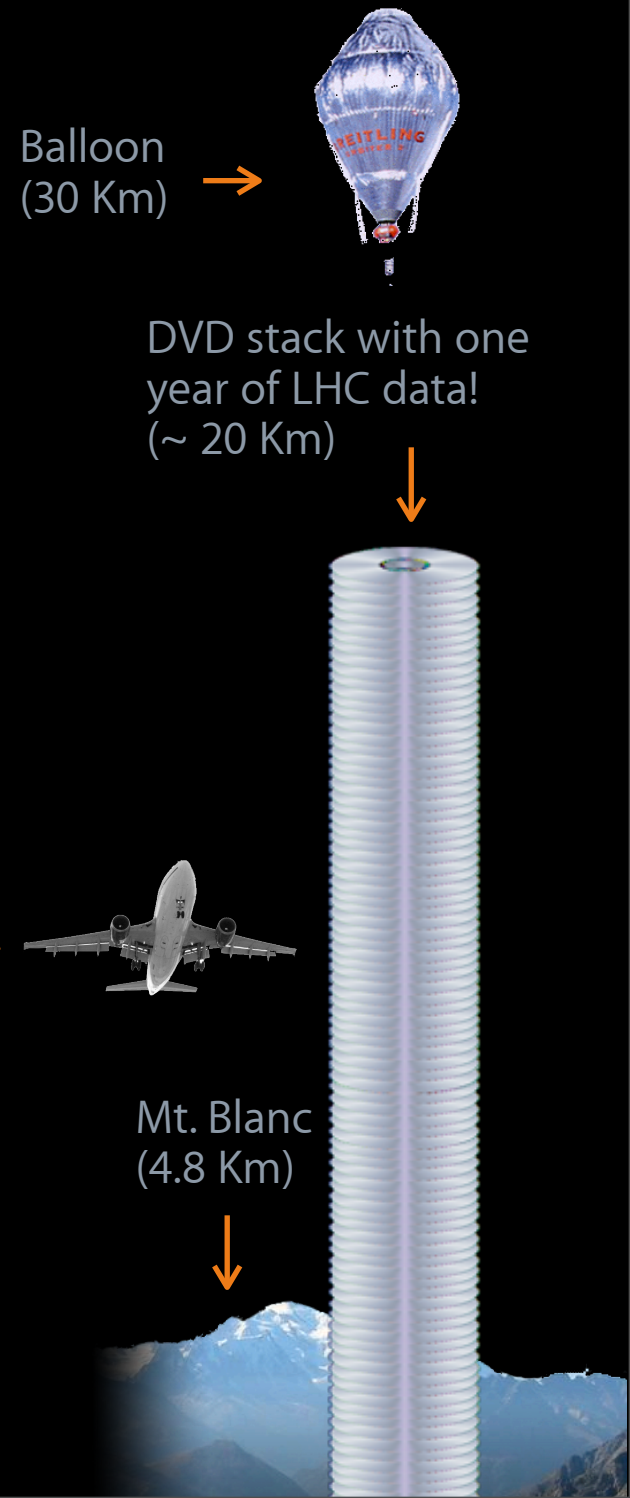
Fons Rademakers, Gerri Ganis, Jan Iwaszkiewicz
CERN

LHC Data Challenge

- The LHC generates:
  - 40 million collisions per second
- Combined the 4 experiments record:
  - After filtering, 100 interesting collision per second
  - From 1 to 12 MB per collision ⇒ from 0.1 to 1.2 GB/s
  - 10^{10} collisions registered every year
  - ~ 10 PetaBytes (10^{16} B) per year
  - LHC data correspond to 20 millions DVD's per year!
  - Computing power equivalent to 100.000 of today's PC
  - Space equivalent to 400.000 large PC disks

LHC Data Challenge

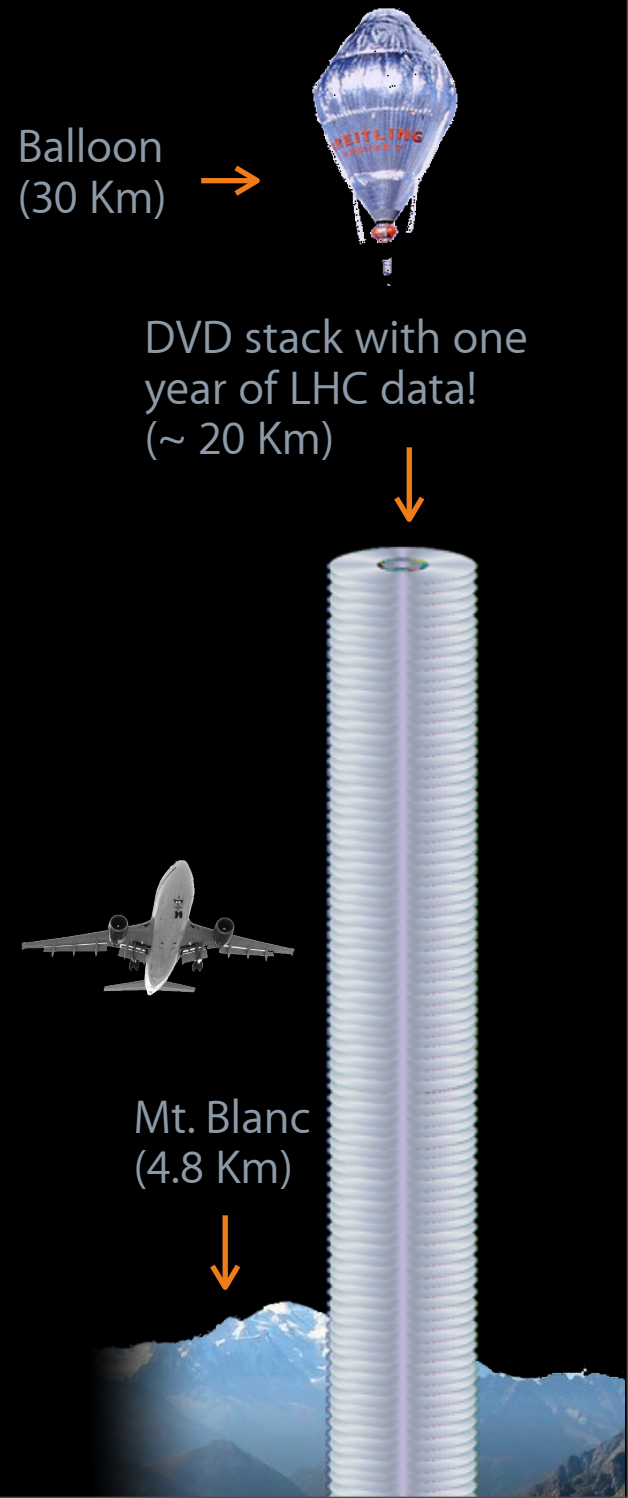
- The LHC generates:
  - 40 million collisions per second
- Combined the 4 experiments record:
  - After filtering, 100 interesting collision per second
  - From 1 to 12 MB per collision ⇒ from 0.1 to 1.2 GB/s
  - 10^{10} collisions registered every year
  - ~ 10 PetaBytes (10^{16} B) per year
  - LHC data correspond to 20 millions DVD's per year!
  - Computing power equivalent to 100.000 of today's PC
  - Space equivalent to 400.000 large PC disks



LHC Data Challenge

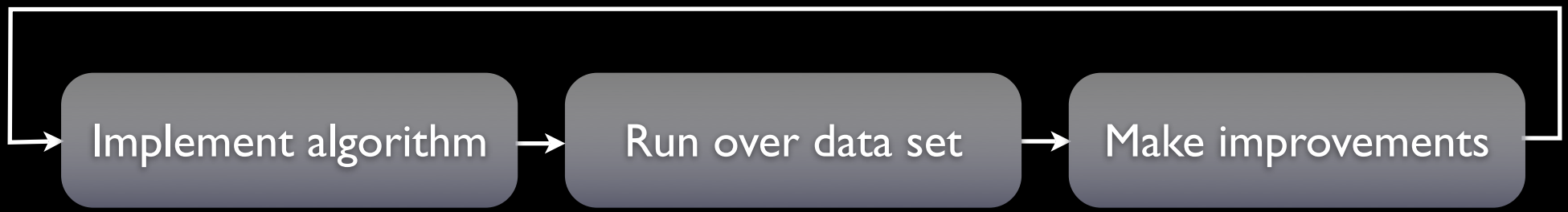
- The LHC generates:
  - 40 million collisions per second
- Combined the 4 experiments record:
  - After filtering, 100 interesting collision per second
  - From 1 to 12 MB per collision ⇒ from 0.1 to 1.2 GB/s
  - 10^{10} collisions registered every year
  - ~ 10 PetaBytes (10^{16} B) per year
  - LHC data correspond to 20 millions DVD's per year!
  - Computing power equivalent to 100.000 of today's PC
  - Space equivalent to 400.000 large PC disks

Using parallelism is the only way to analyze this amount of data in a reasonable amount of time



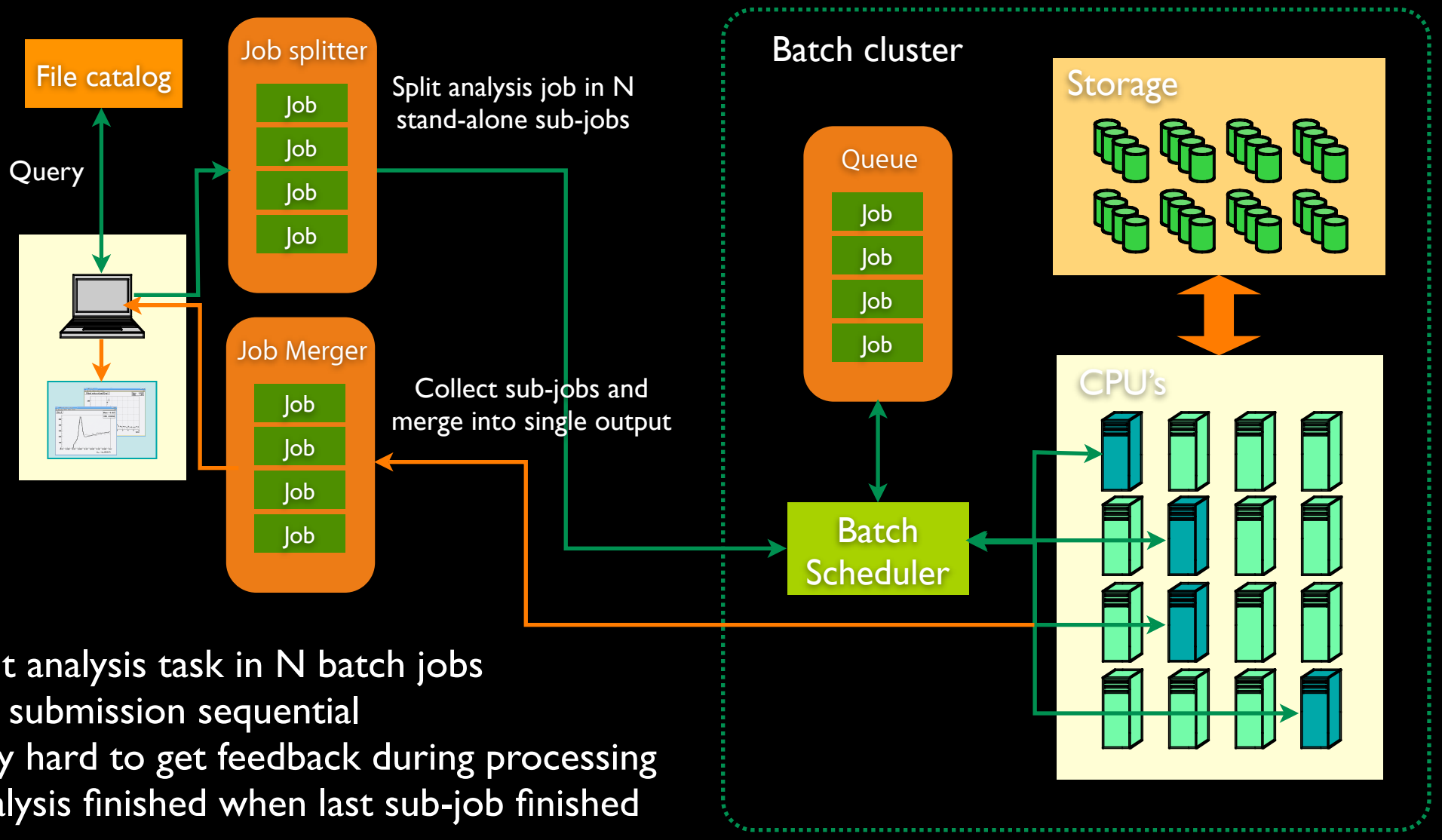
HEP Data Analysis

- Typical HEP analysis needs a continuous algorithm refinement cycle



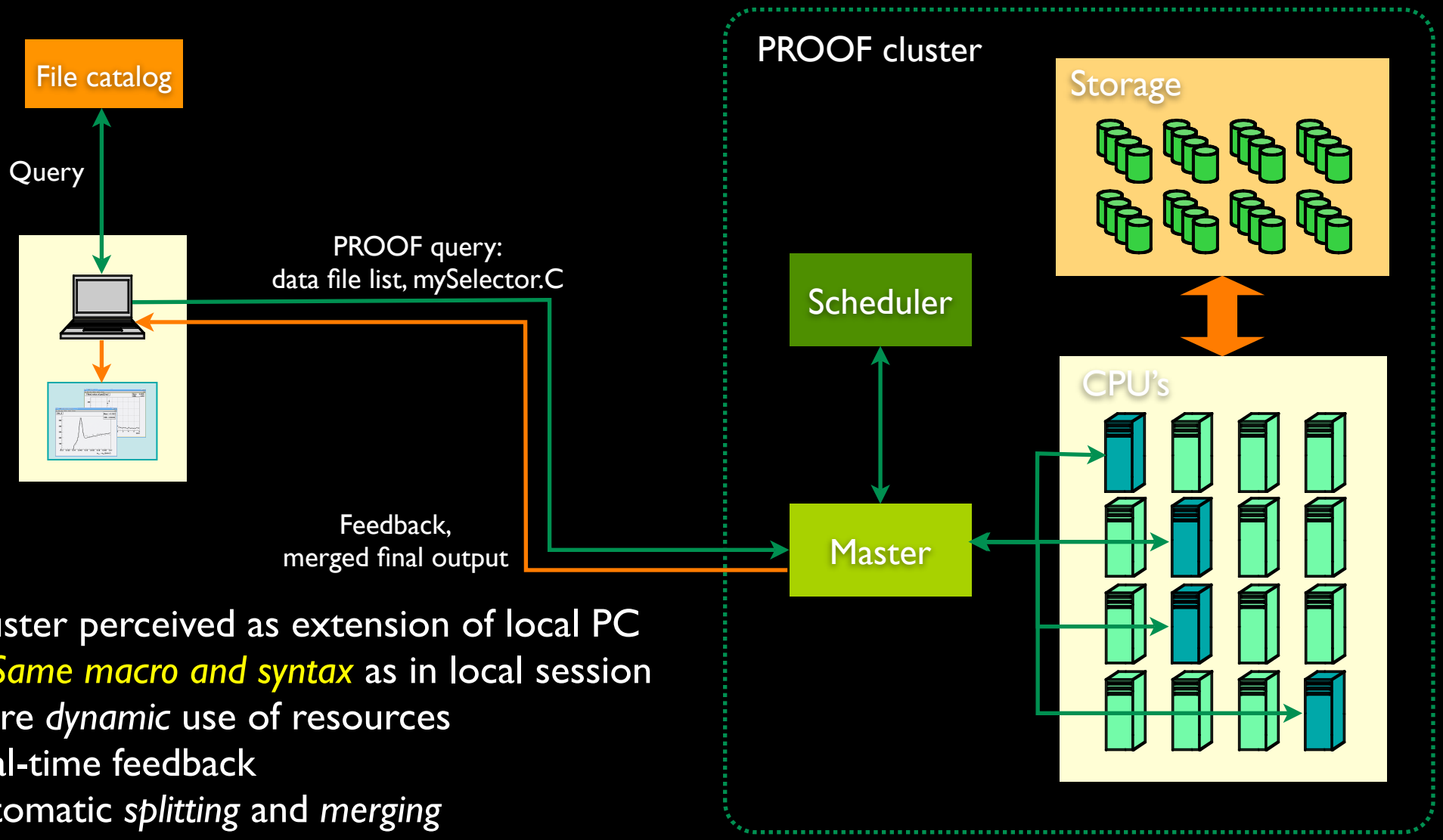
- Ranging from I/O bound to CPU bound
- Need many disks to get the needed I/O rate
- Need many CPUs for processing
- Need a lot of memory to cache as much as possible

The Traditional Batch Approach



- Split analysis task in N batch jobs
- Job submission sequential
- Very hard to get feedback during processing
- Analysis finished when last sub-job finished

The PROOF Approach



- Cluster perceived as extension of local PC
  - *Same macro and syntax* as in local session
- More *dynamic* use of resources
- Real-time feedback
- Automatic *splitting* and *merging*

PROOF Design Goals

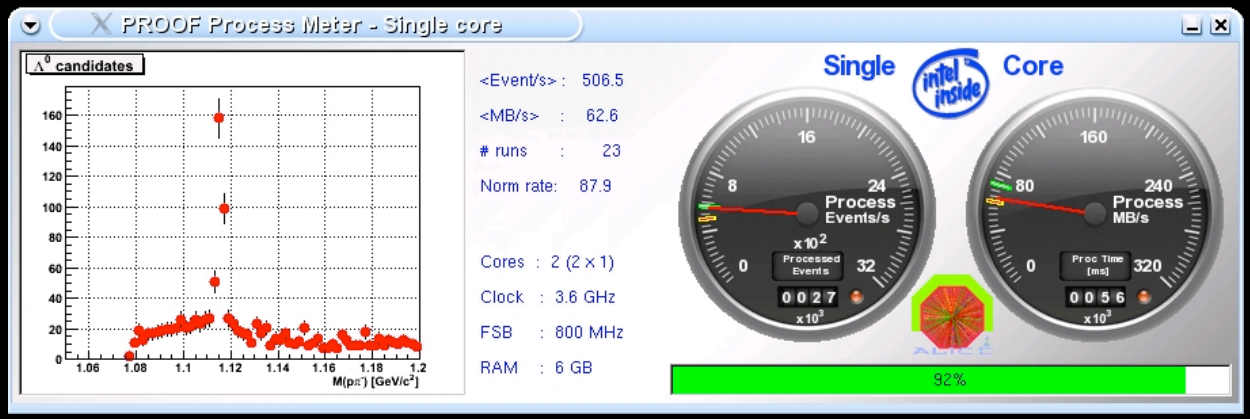
- System for running ROOT queries in parallel on a large number of distributed computers or multi-core machine
- Transparent, scalable and adaptable extension of the local interactive ROOT analysis session
- Support for running long queries (“interactive batch”)

Where to Use PROOF

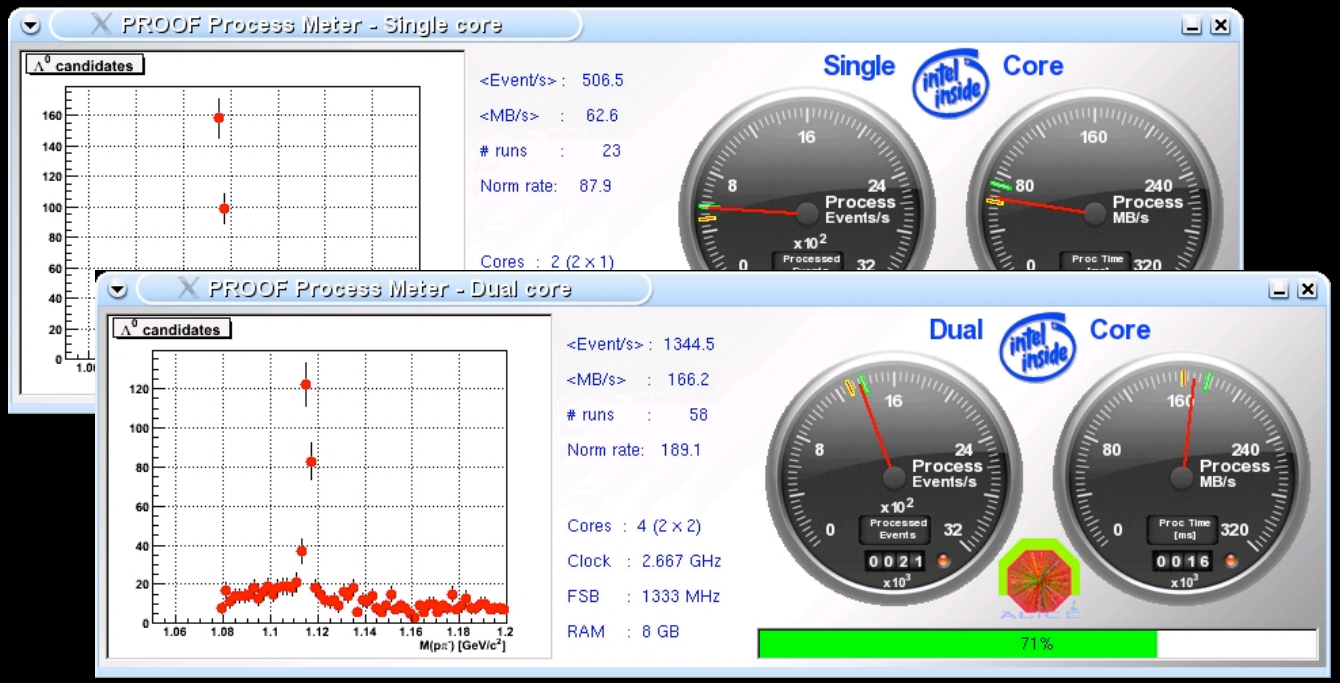
- CERN Analysis Facility (CAF)
- Departmental workgroups (Tier-2’s)
- Multi-core, multi-disk desktops (Tier-3/4’s)

PROOF Scalability on Multi-Core Machines

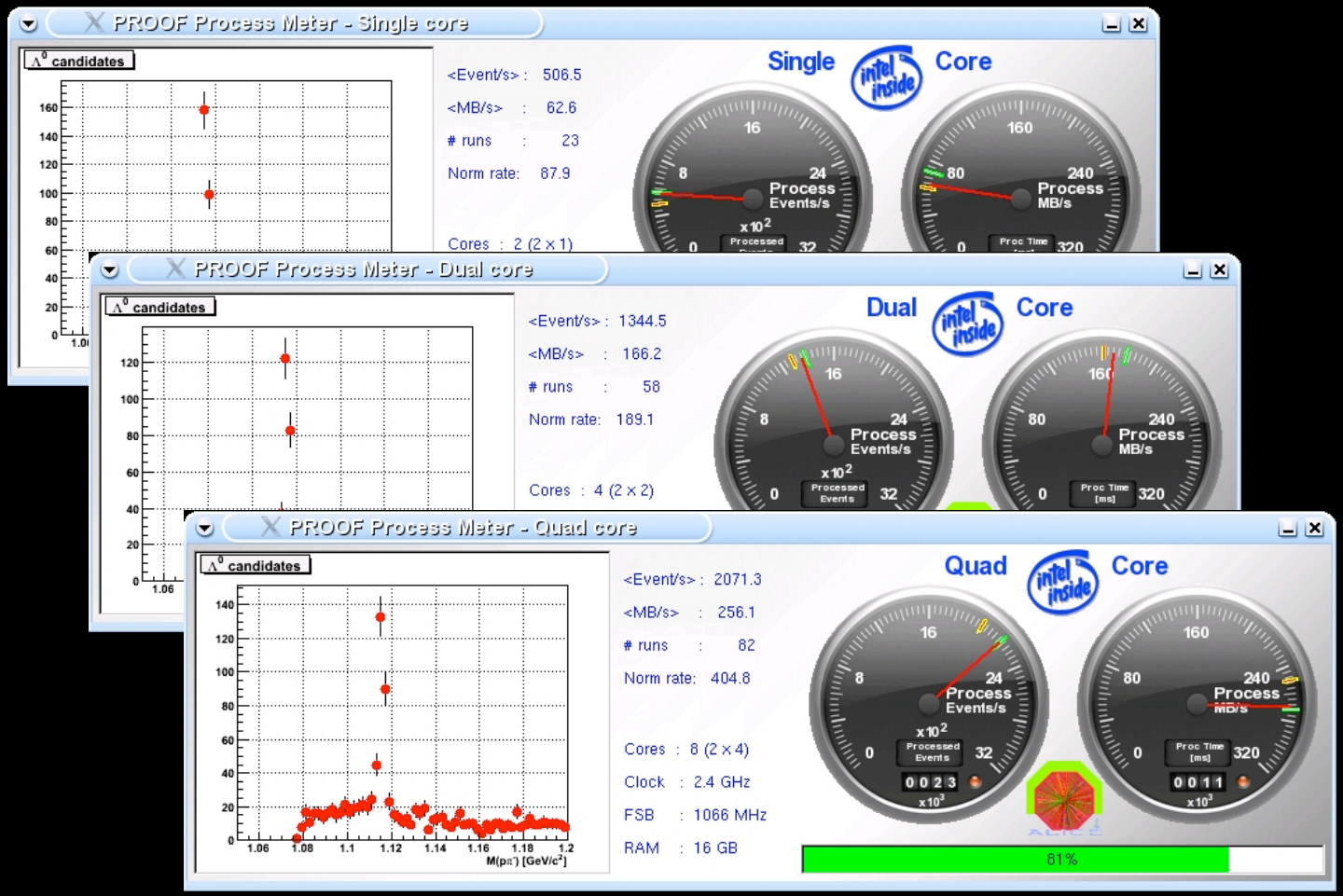
PROOF Scalability on Multi-Core Machines



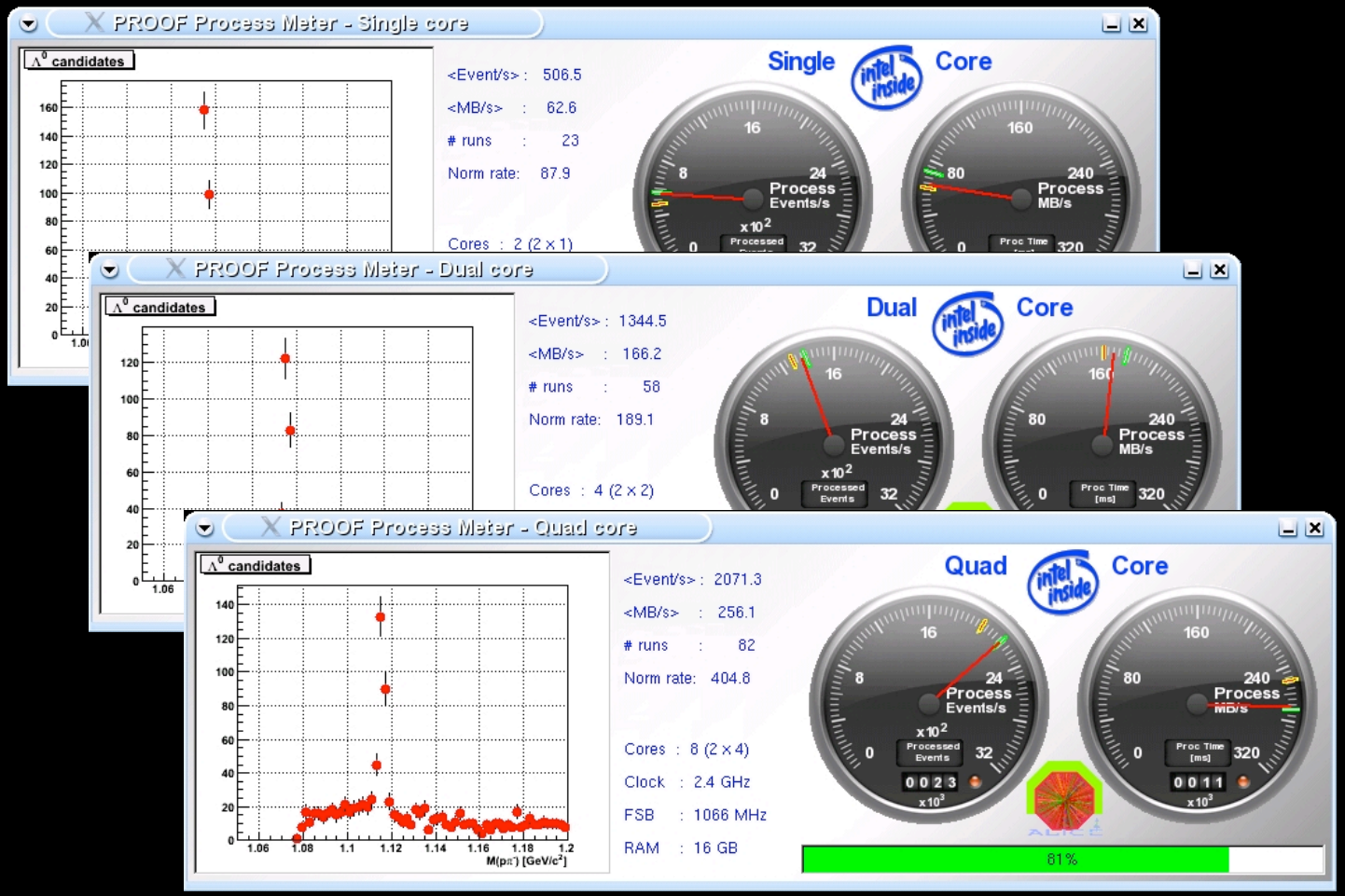
PROOF Scalability on Multi-Core Machines



PROOF Scalability on Multi-Core Machines

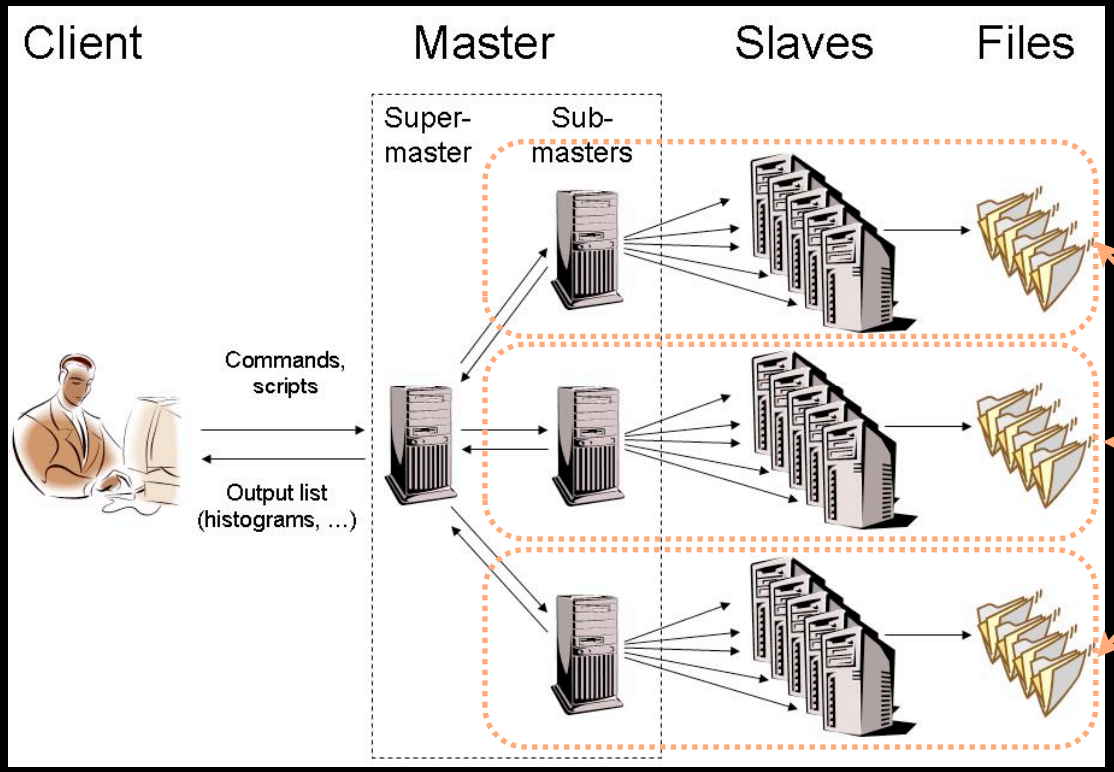


PROOF Scalability on Multi-Core Machines



Running on MacPro with dual Quad Core CPU's.

Multi-Tier Architecture



Adapts to wide area *virtual* clusters

Geographically separated domains, heterogeneous machines

Network performance

Less important → VERY important

Optimize for data locality or high bandwidth data server access

Recent Developments

- Dataset management
  - Global and user data sets
  - Disk quotas
  - For more see talk 444 on ALICE CAF developments
- Load balancing
  - New packetizers
- Scheduling
  - User priority handling on worker level
  - Central resource scheduler
  - Abstract interface
  - Selection of workers based on load (CPU, memory, I/O)
- Generic task processing
  - CPU instead of data driven

Load Balancing: the Packetizer

- The packetizer is the heart of the system
- It runs on the master and hands out work to the workers
  - Pull architecture: workers ask for work
  - No complex worker state in the master
- Different packetizers allow for different data access policies
  - All data on disk, allow network access
  - All data on disk, no network access
  - Data on mass storage, go file-by-file
  - Data on Grid, distribute per Storage Element
  - …
- The goal is to have all workers end at the same time

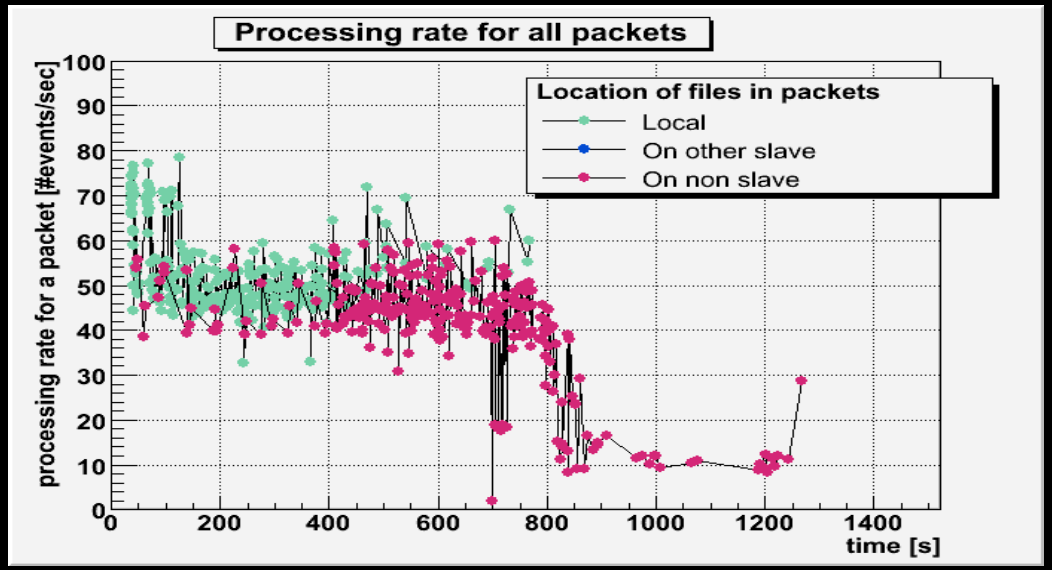
Original Packetizer Strategy

- Each worker processes its local files and processes packets from the remaining remote files (if any)
- Fixed packet size
- Avoid data servers overload by allowing max 4 remote workers to be served concurrently

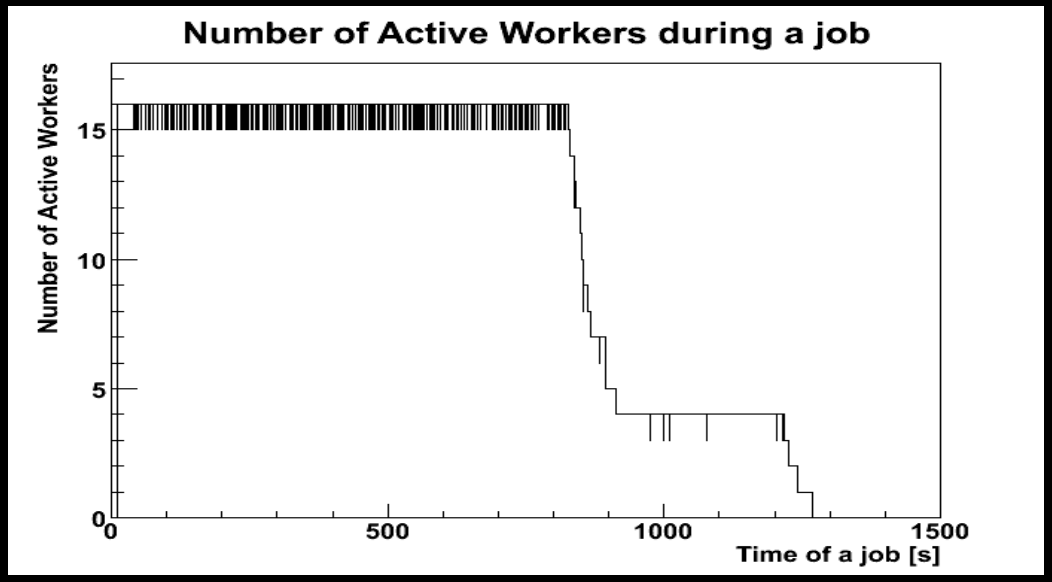
- Works generally fine, but shows tail effects for I/O bound queries, due to a reduction of the effective number of workers when access to non-local files is required

Issues with Original Packetizer Strategy

- Processing rate during a query



- Resource utilization



Where to Improve

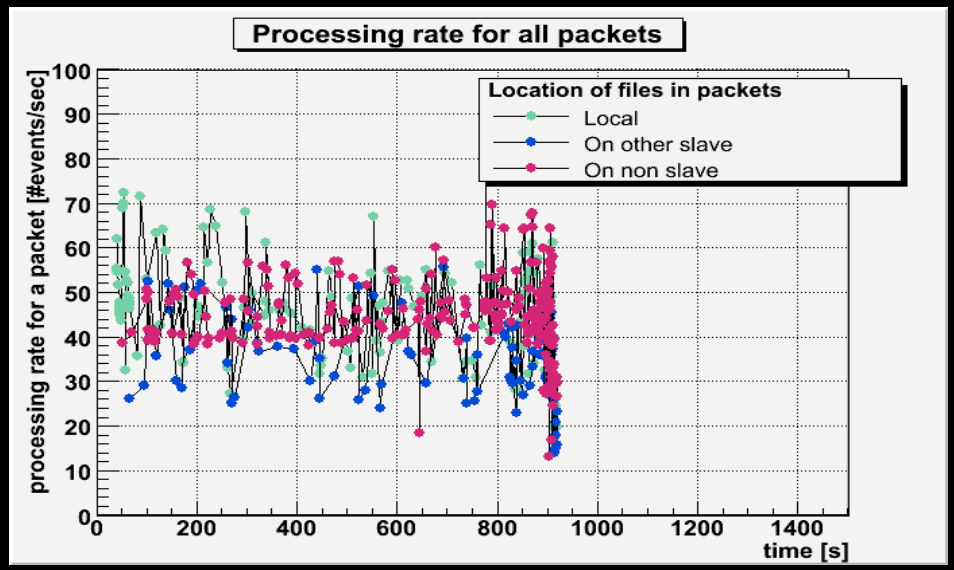
- Focus on I/O bound jobs
  - Limited by disk or network bandwidth
- Predict which data servers can become bottlenecks
- Make sure that other workers help analyzing data from those servers
- Use variable packet sizes (smaller at end of query)

Improved Packetizer Strategy

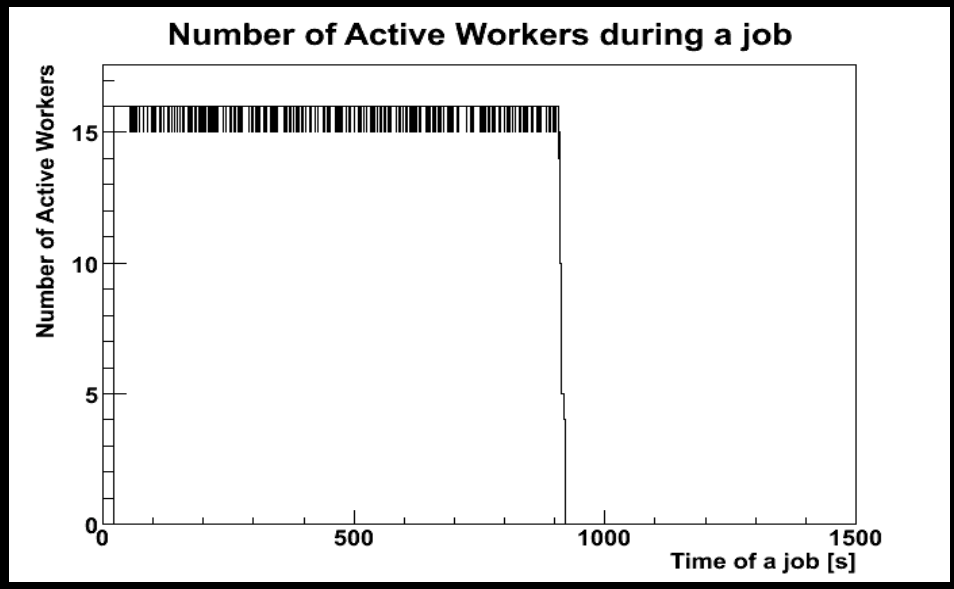
- Predict processing time of local files for each worker
- For the workers that are expected to finish faster, keep assigning remote packets from the beginning of the job
- Assign remote packets from the most heavily loaded file servers
- Variable packet size

Improved Packetizer: Results

- Processing rate during a query

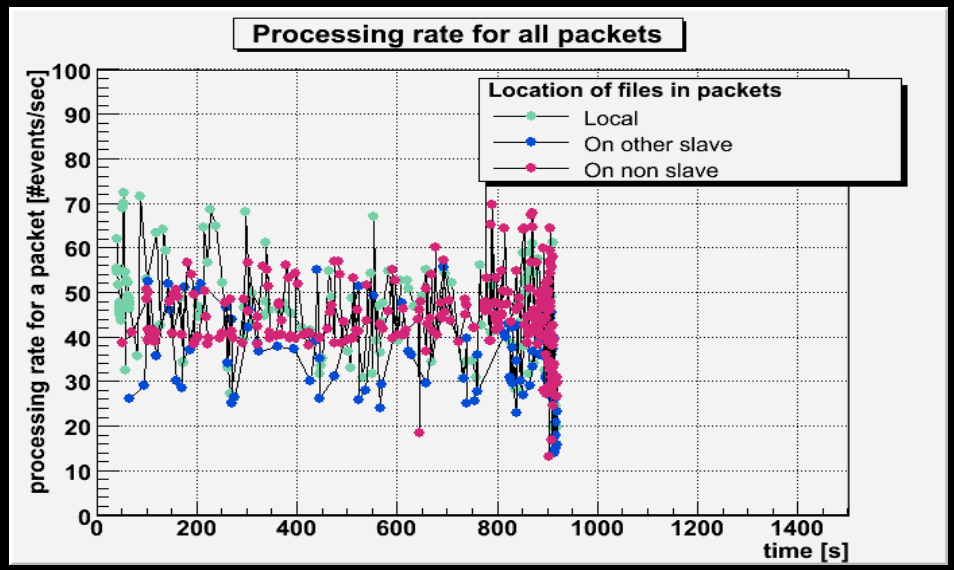


- Resource utilization

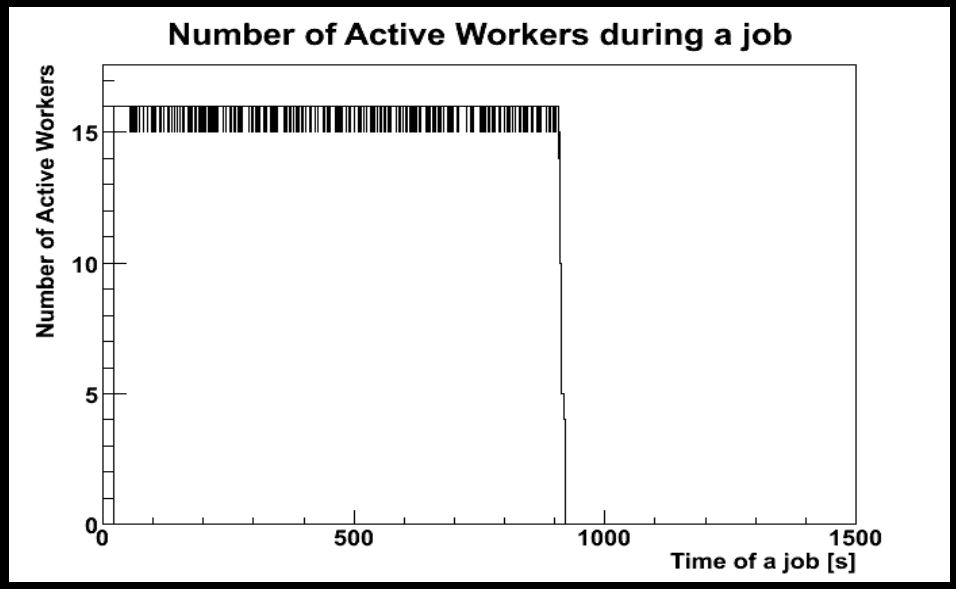


Improved Packetizer: Results

- Processing rate during a query



- Resource utilization



Up to 30% improvement

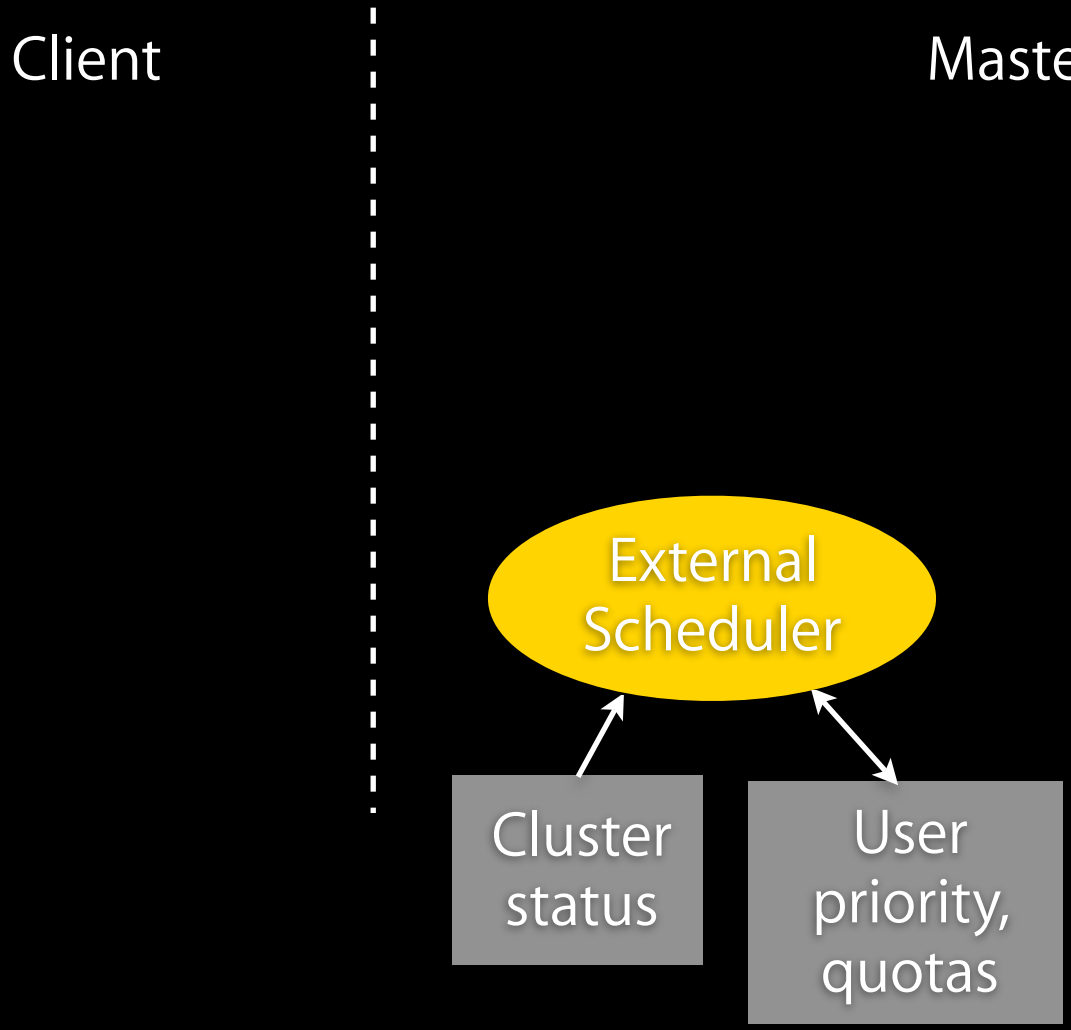
Why Scheduling?

- Controlling resources and how they are used
- Improving efficiency
  - Assigning to a job those nodes that have data which needs to be analyzed
- Implementing different scheduling policies
  - E.g. fair share, group priorities & quotas
- Avoid congestion and cluster grinding to a halt

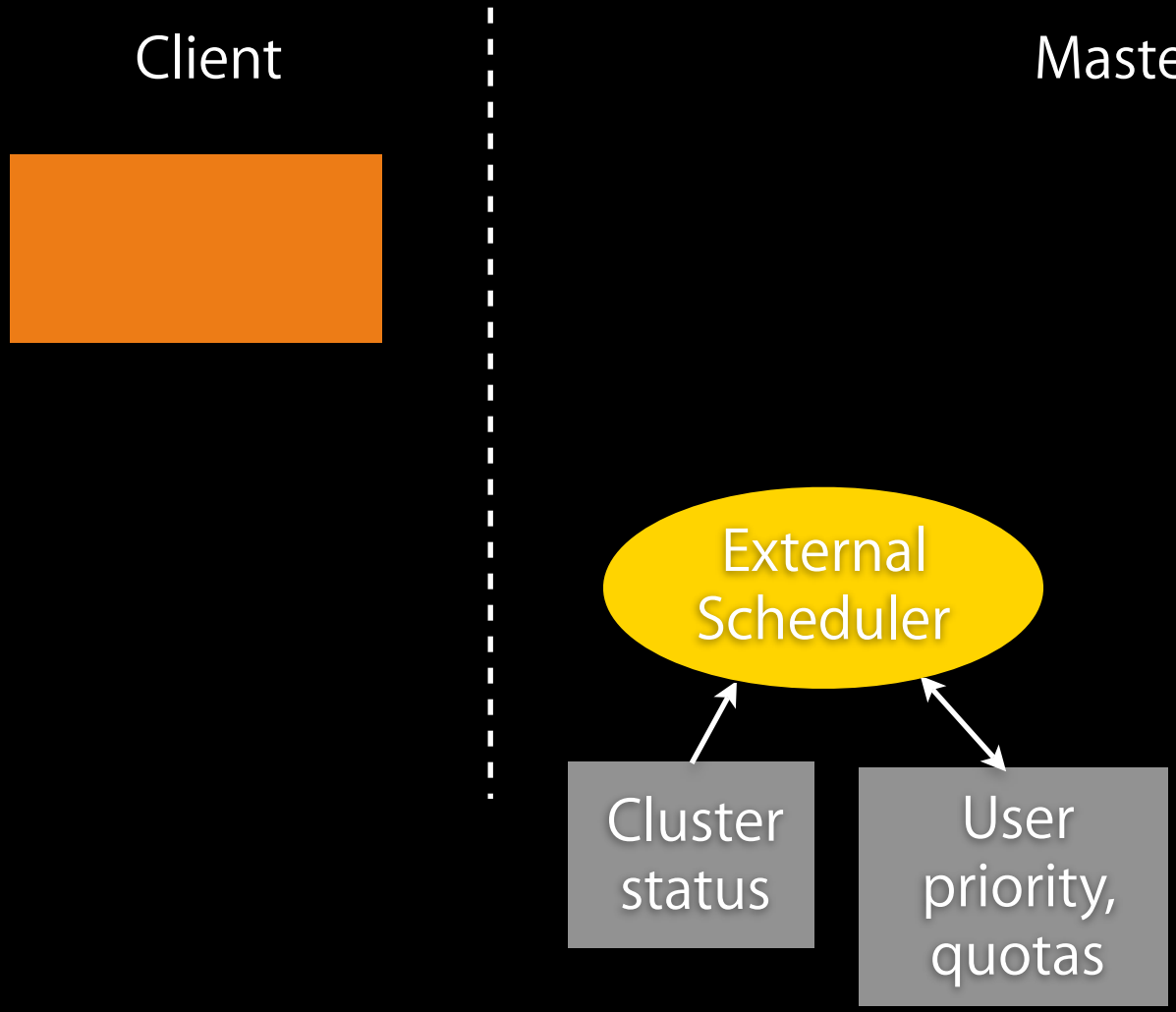
PROOF Specific Requirements

- Interactive system
  - Jobs should be processed as soon as submitted
  - However when max. system throughput is reached some jobs have to be postponed
- I/O bound jobs use more resources at the start and less at the end (file distribution)
- Try to process data locally
- User defines a data set not the number of workers
- Possibility to remove/add workers during a job

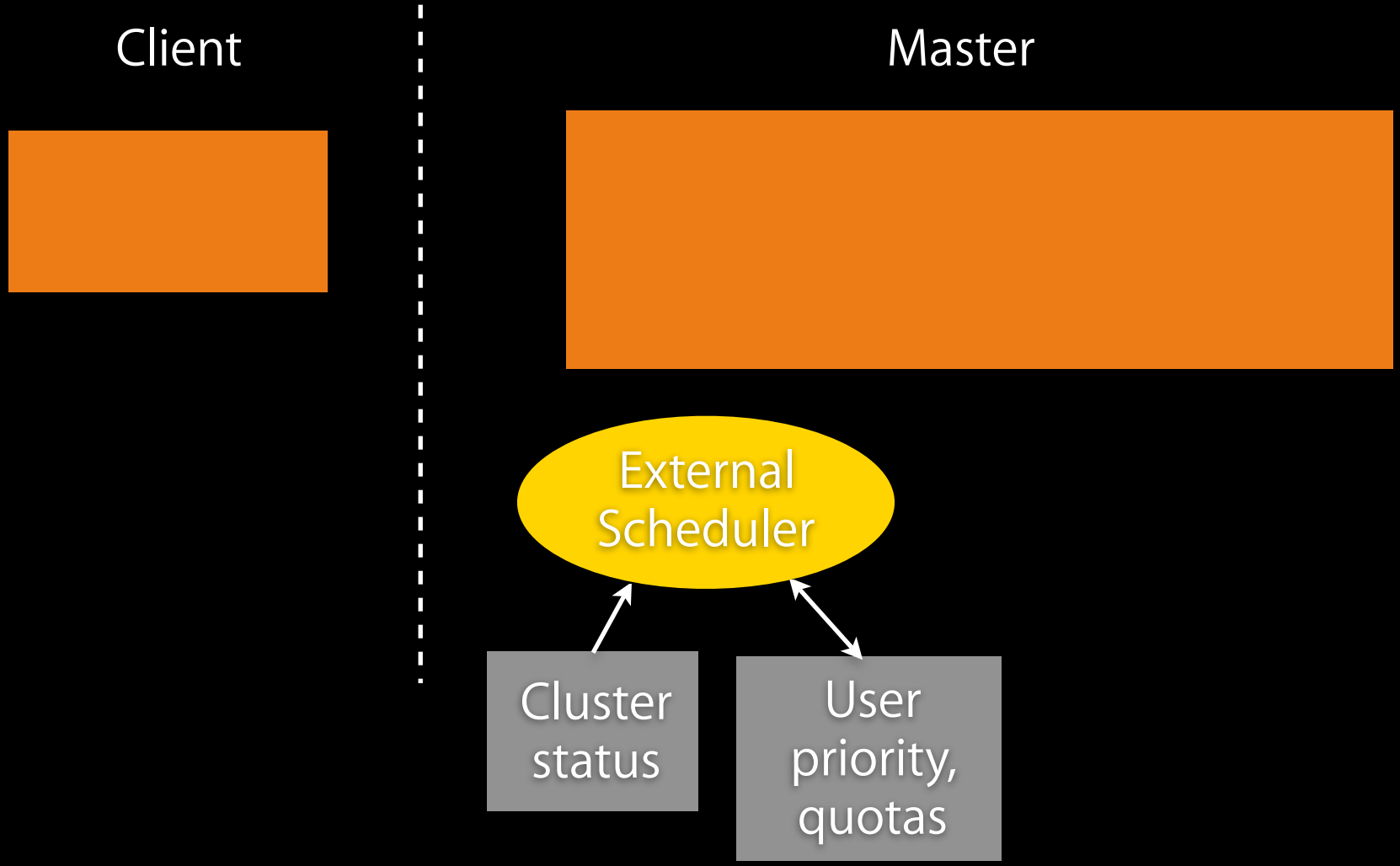
Starting a Query With a Central Scheduler



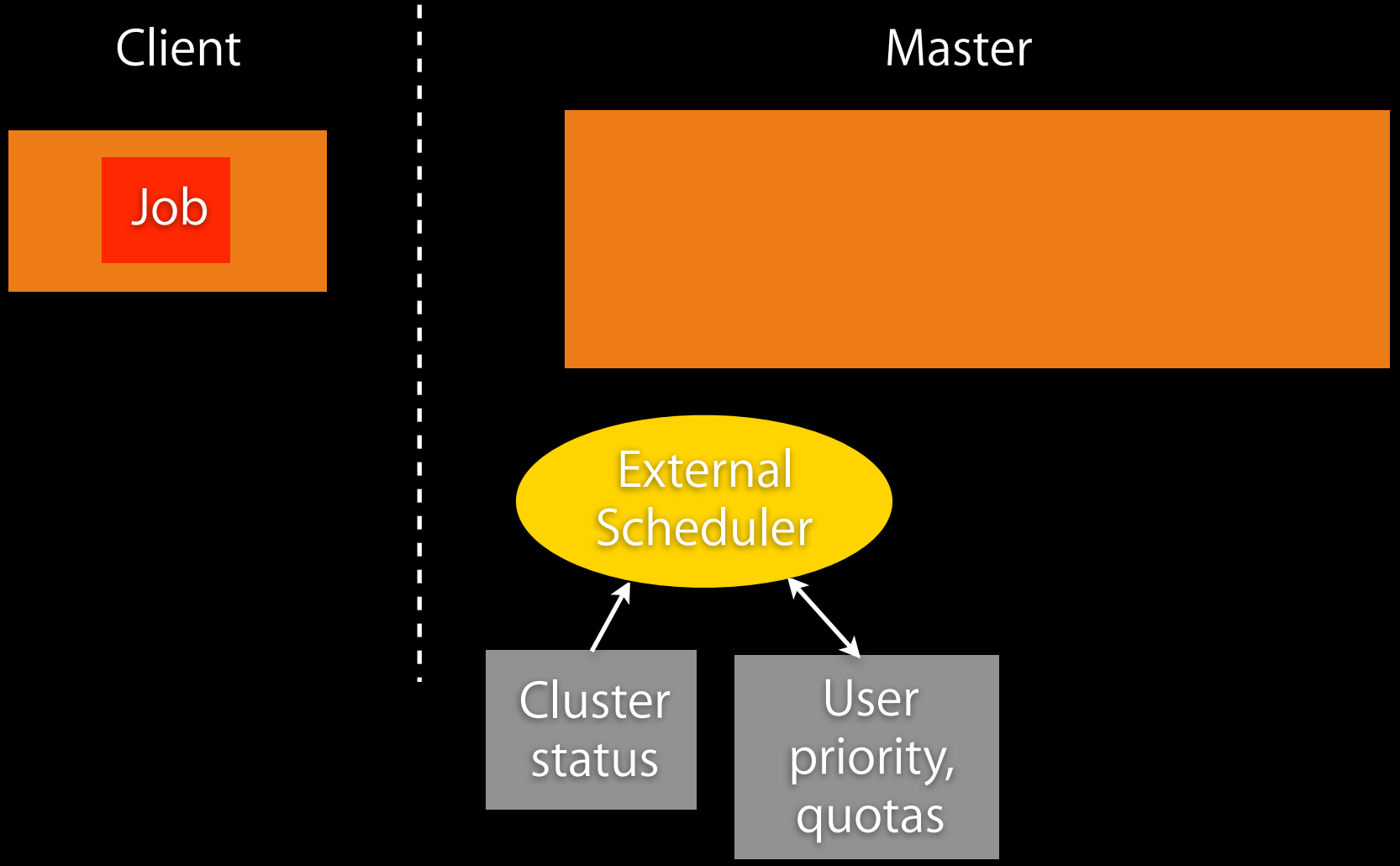
Starting a Query With a Central Scheduler



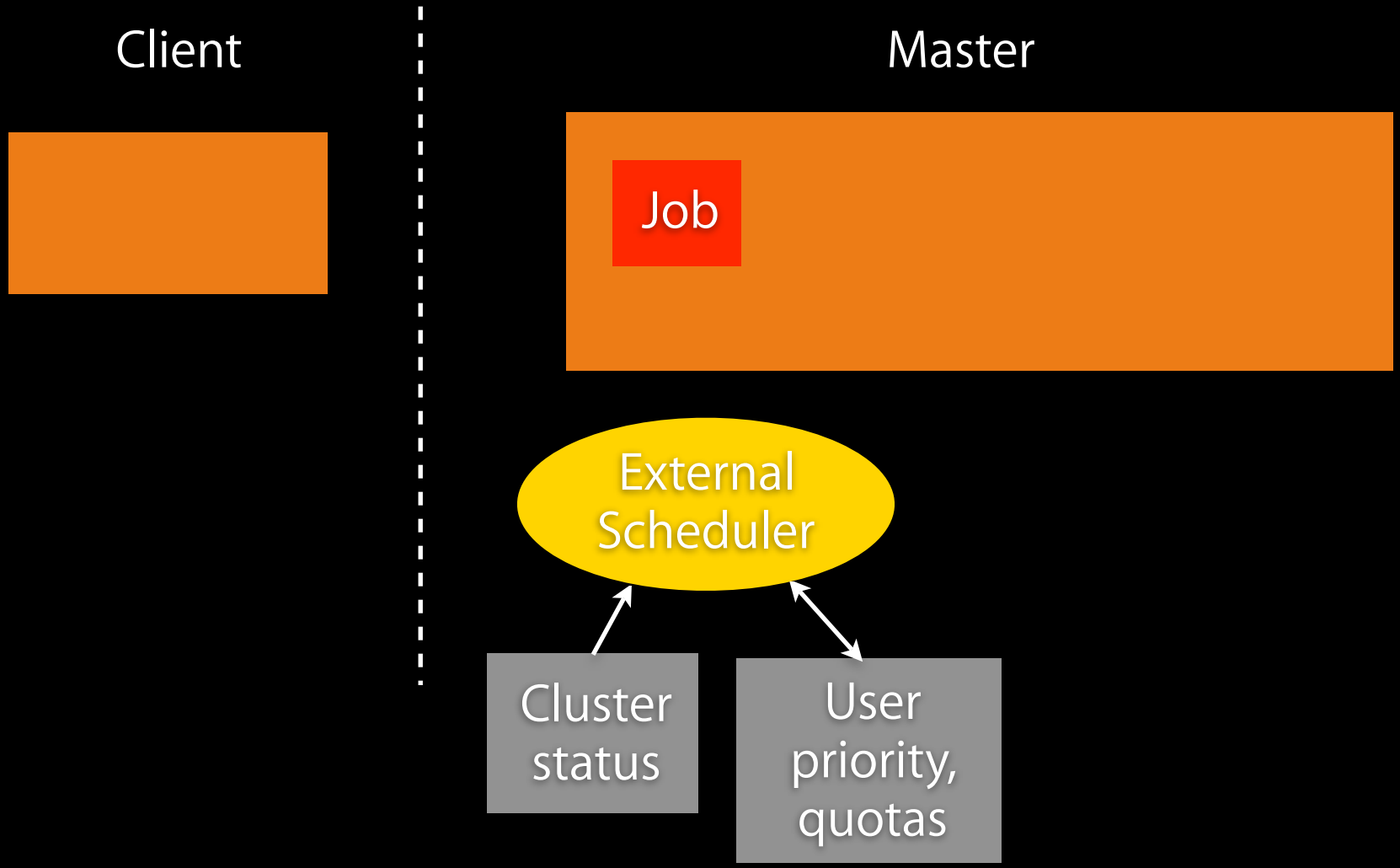
Starting a Query With a Central Scheduler



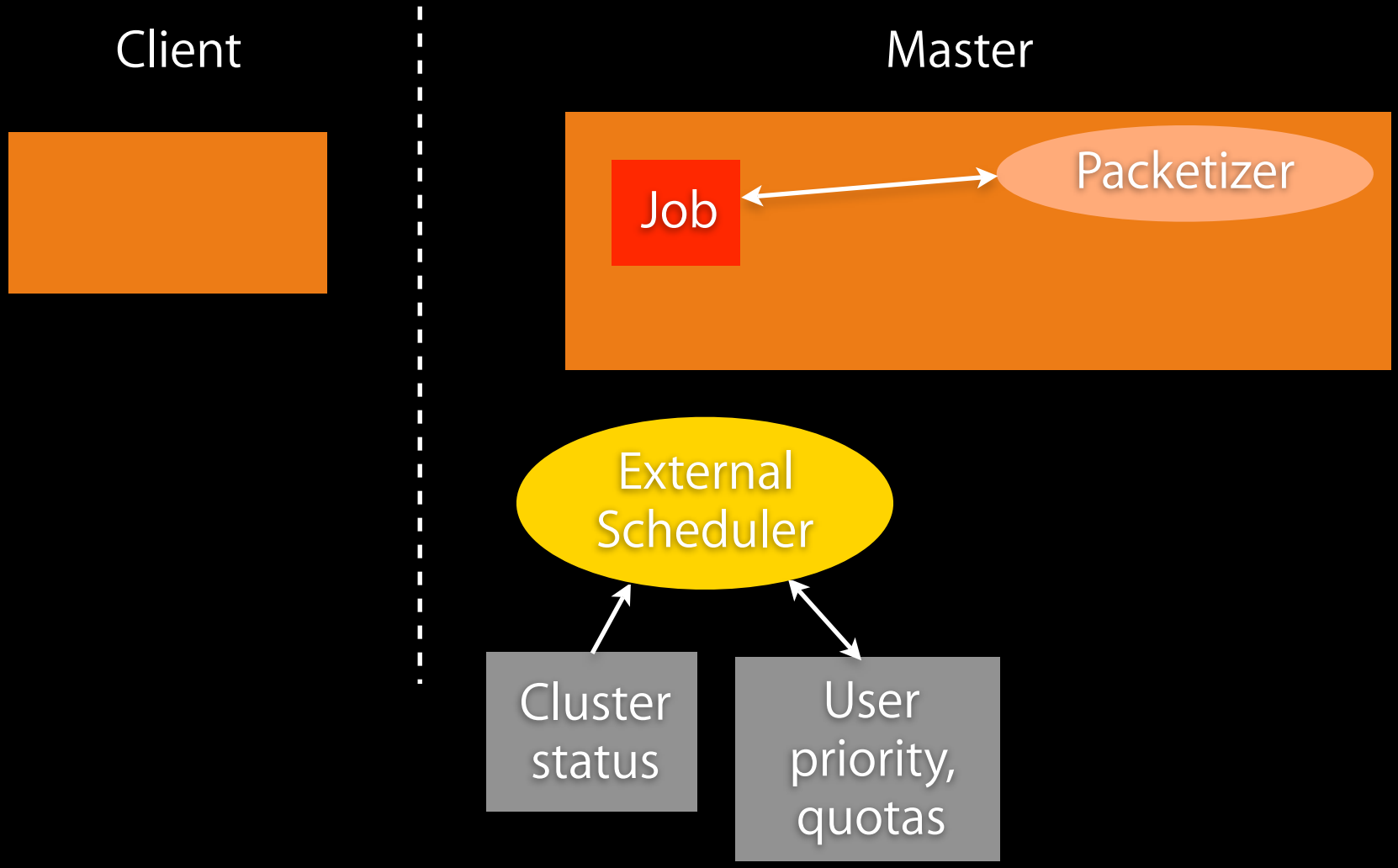
Starting a Query With a Central Scheduler



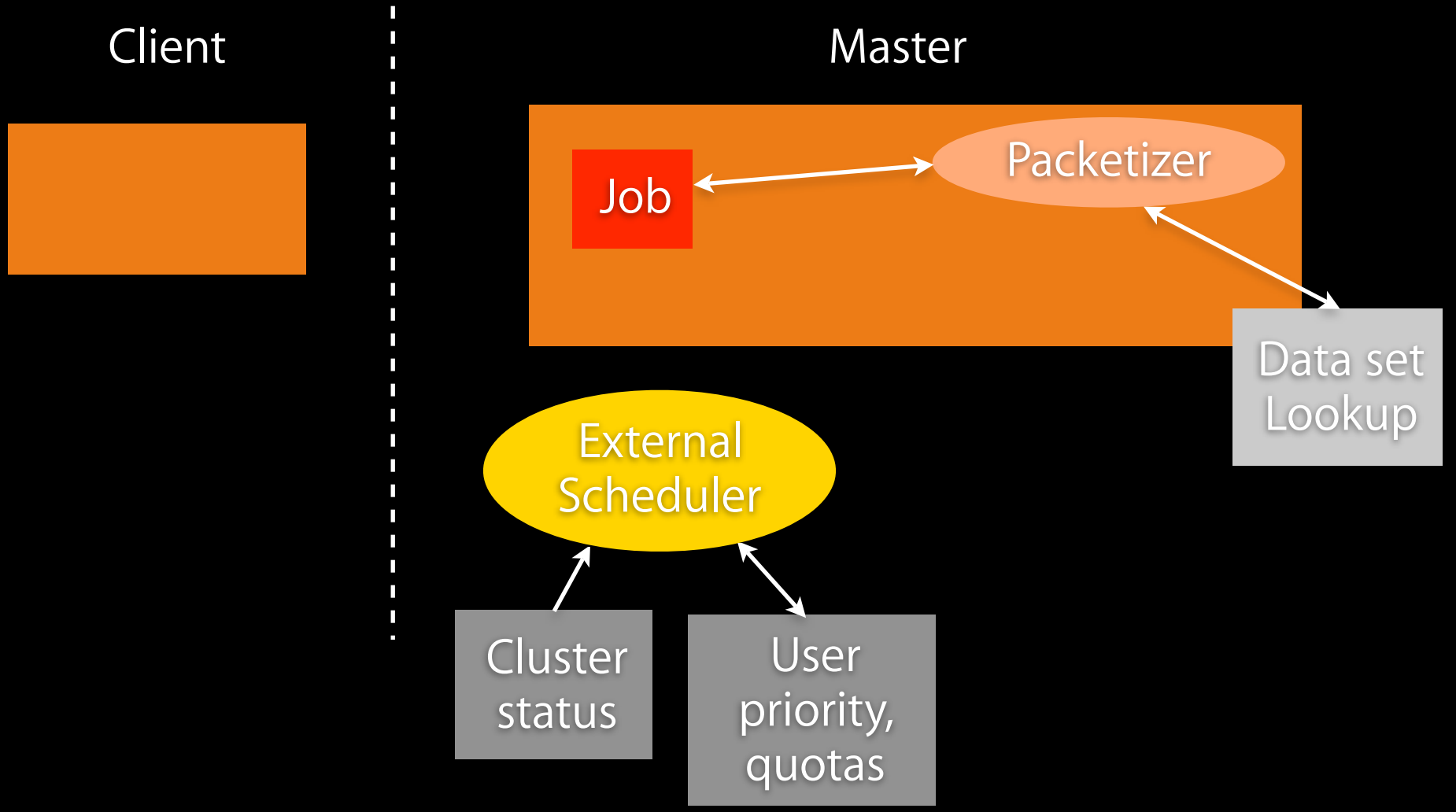
Starting a Query With a Central Scheduler



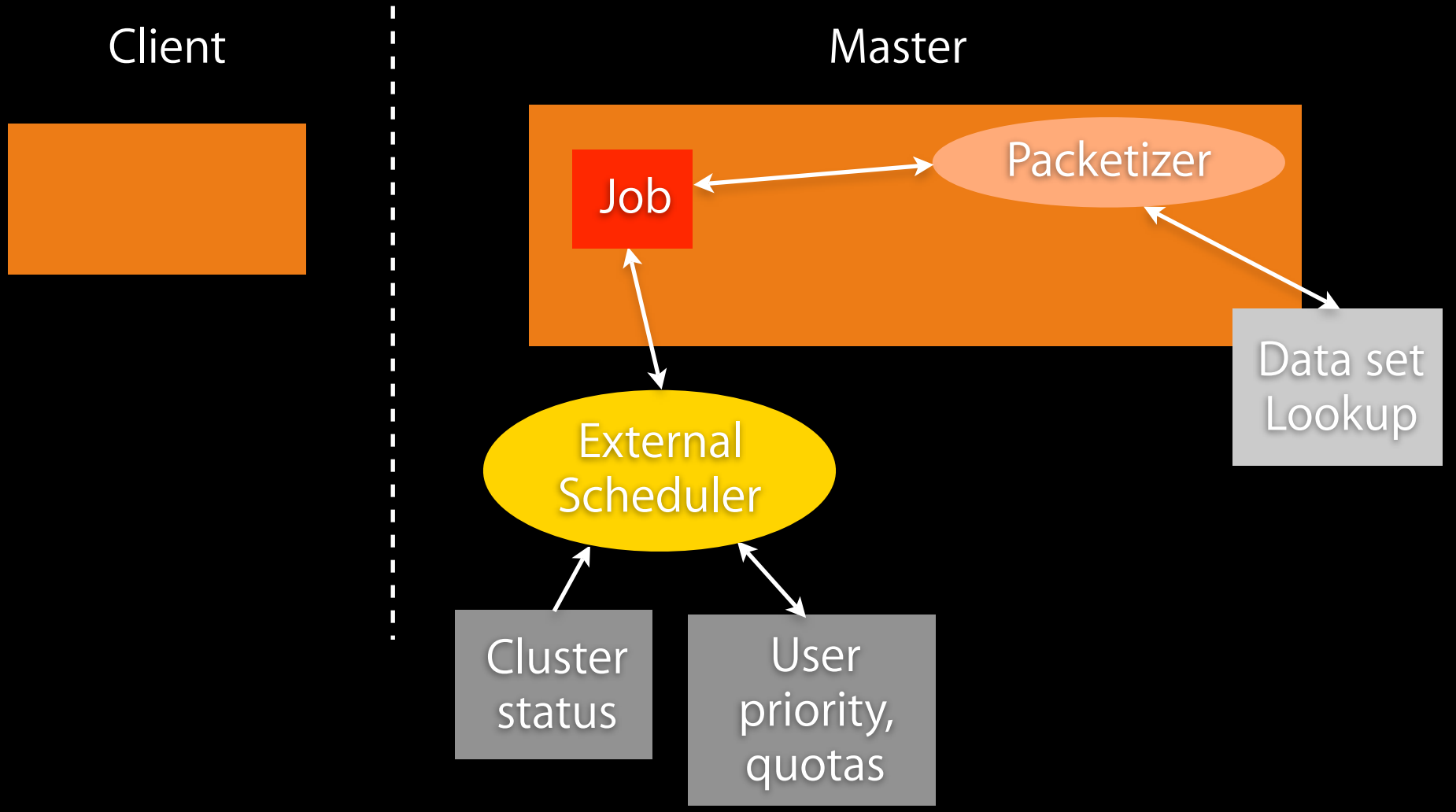
Starting a Query With a Central Scheduler



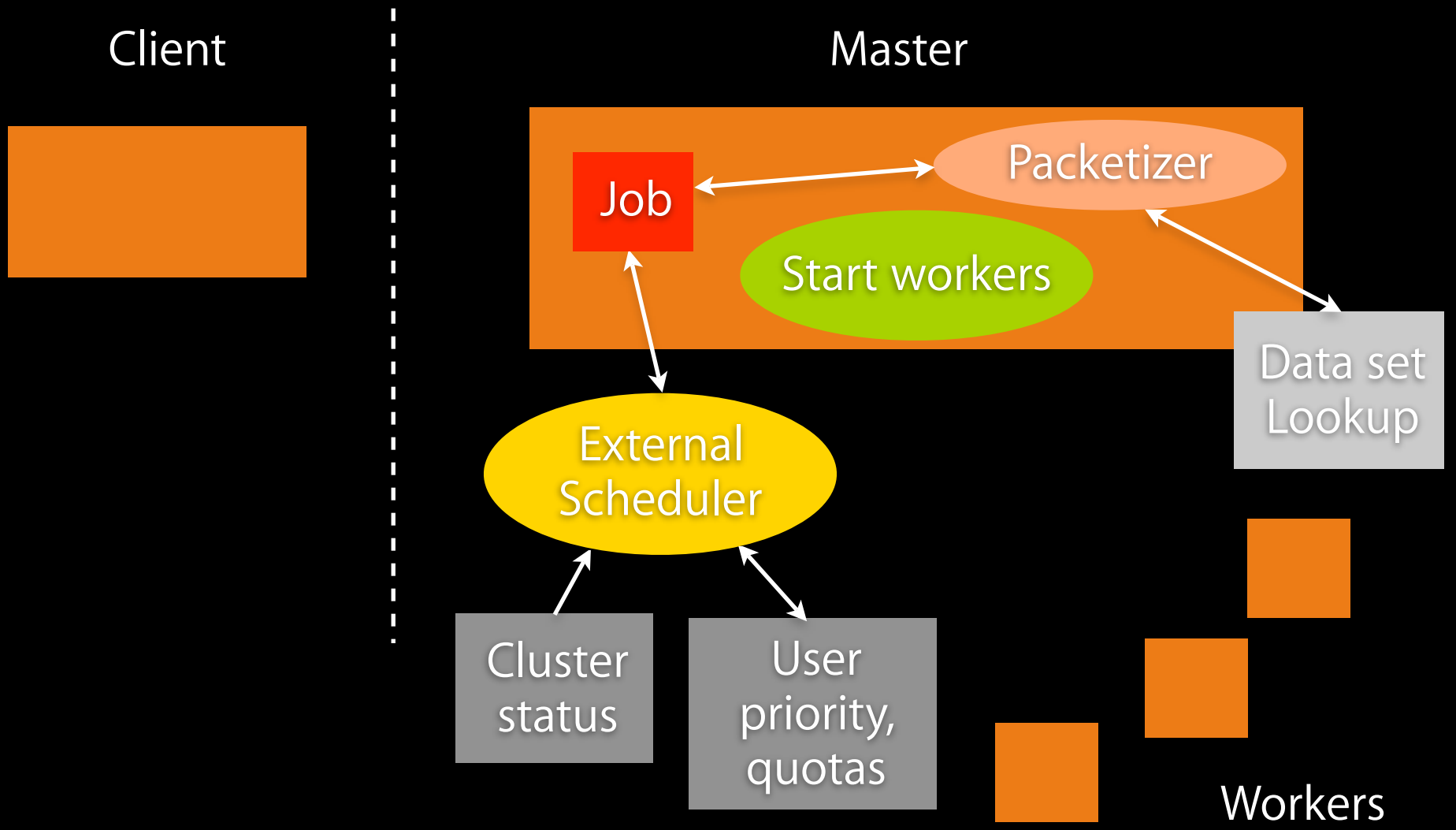
Starting a Query With a Central Scheduler



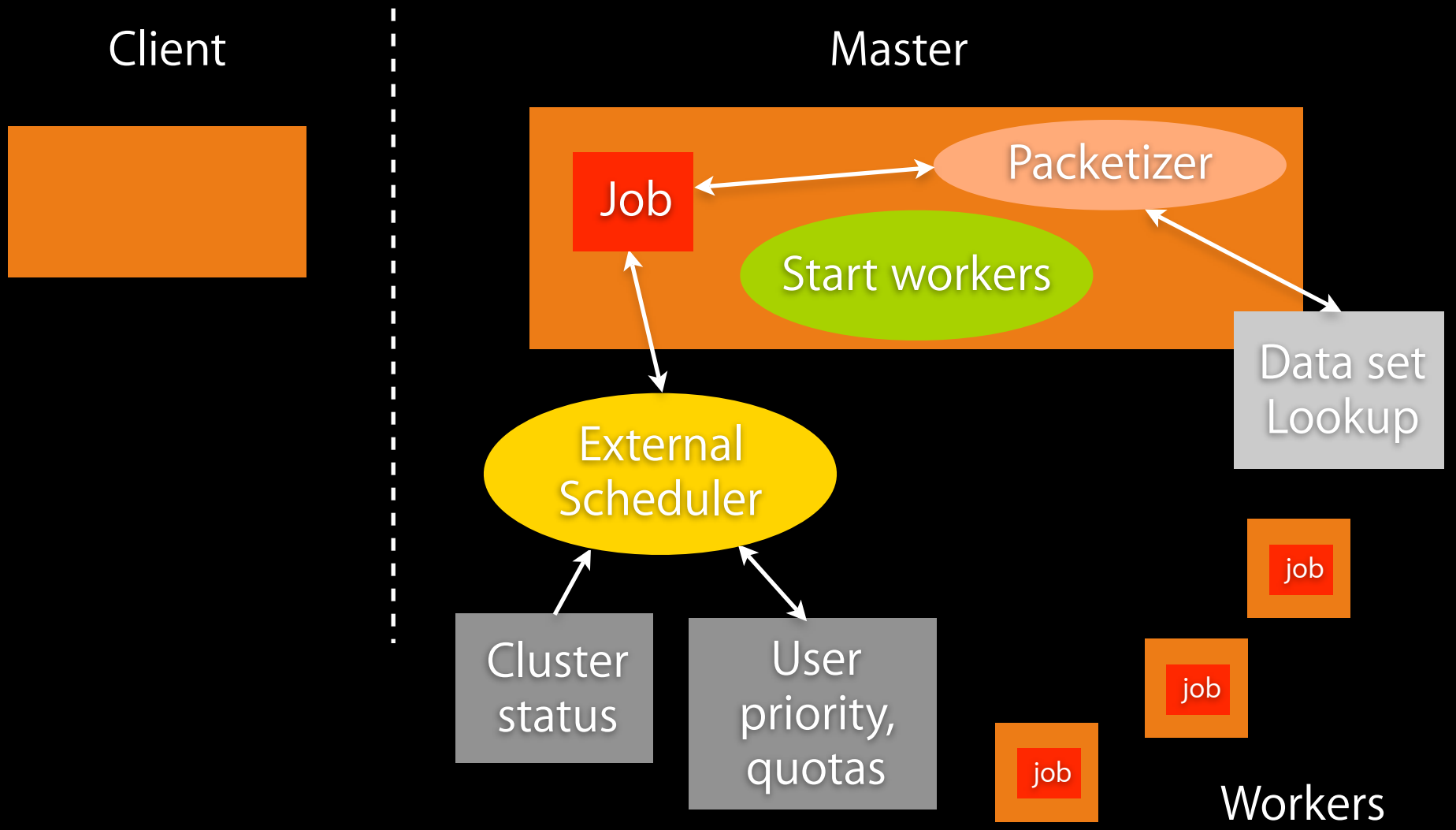
Starting a Query With a Central Scheduler



Starting a Query With a Central Scheduler



Starting a Query With a Central Scheduler



Scheduler Development Plans

- Interface for scheduling "per job”
  - Special functionality will allow to change the set of nodes during a session without loosing user libraries and other settings
- Removing workers during a job
- Integration with a third-party scheduler
  - Maui, LSF

User Priority Based Scheduling

- User priority based worker level scheduling
  - Simple and solid implementation, no global state
  - Group priorities defined in a configuration file
  - Group priorities can also be obtained from a central scheduler via the master
  - Configuration tested currently at the CAF by ALICE
- Scheduling performed on each worker independently
- Lower priority processes slowdown
  - Sleep before next packet request
  - Use Round-Robin Linux process scheduler

Generic Task Processing

- CPU instead of data driven
- Uses the established PROOF infrastructure to distribute jobs (i.e. selectors, input lists, output lists, PAR files, etc.)
  - Monte Carlo, image analysis, etc.
  - Output files in the output list will be automatically merged
- First version will be coming later this year

Growing Interest by LHC Experiments

- The ideal solution for fast AOD analysis, easy to deploy on cluster or a bunch of multi-core machines
  - ALICE CAF
  - ATLAS
    - BNL, Wisconsin
  - CMS
    - FNAL

Conclusions

- The LHC will generate data on a scale not seen anywhere before
- LHC experiments will critically depend on parallel solutions to analyze their enormous amounts of data
- Grids will very likely not provide the needed stability and reliability we need for repeatable high statistics analysis