PROOF-Condor integration for ATLAS

G. Ganis, J. Iwaszkiewicz, F. Rademakers
CERN / PH-SFT
M. Livny, B. Mellado, Neng Xu, Sau Lan Wu
University Of Wisconsin

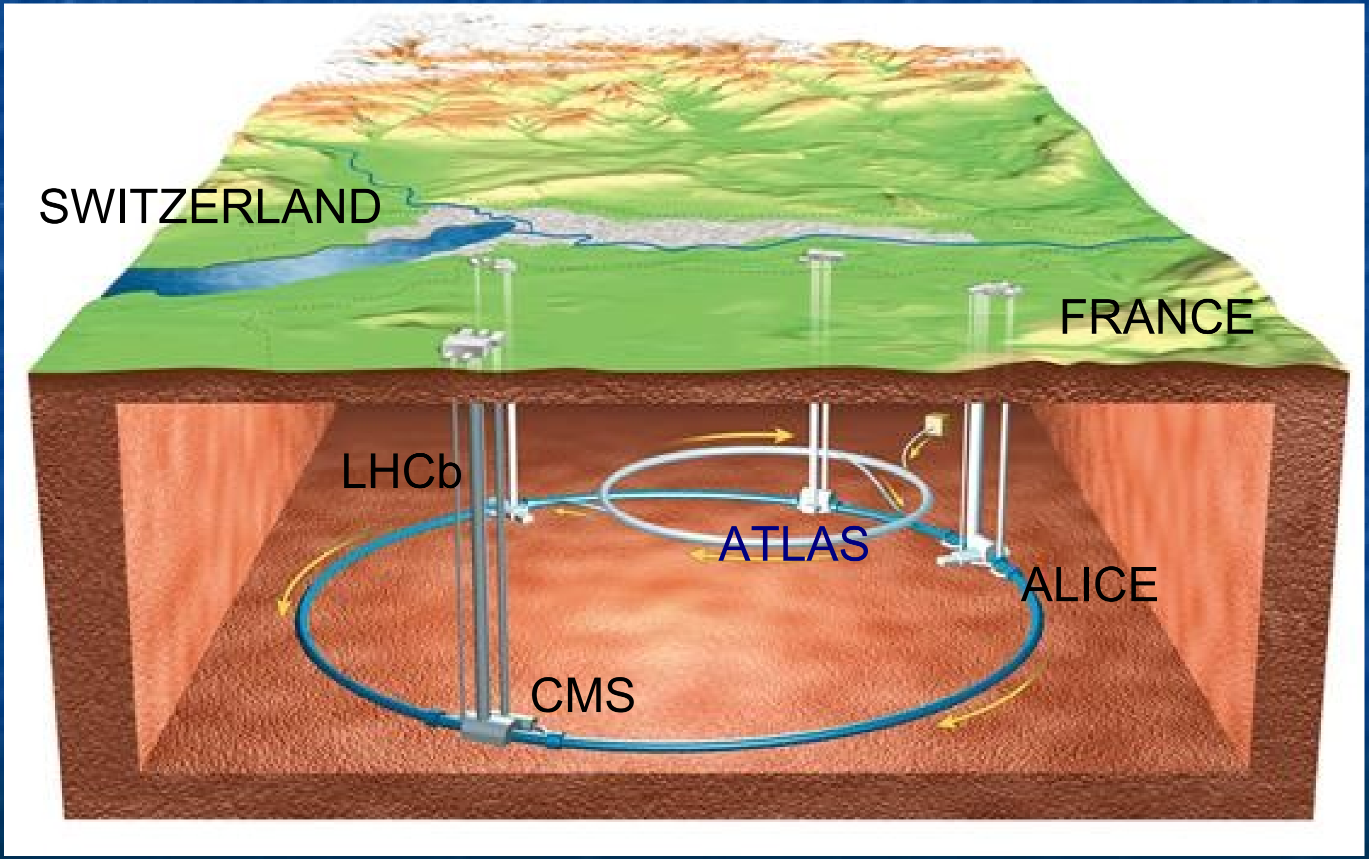
Condor Week, Madison, 29 Apr – 2 May 2008

Outline

- HEP End-User analysis with PROOF
- Why PROOF on Condor
- Proof-Of-Concept using COD
- The ATLAS model
- Summary

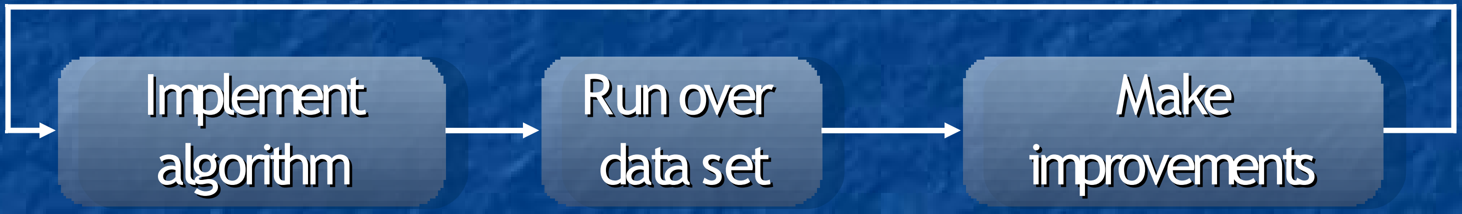
The Large Hadron Collider (LHC)

- p-p collisions at 14 TeV
- start: end 2008 / beg 2009
- 4 experiments



The LHC Data Challenge

- The LHC generates 40·10^6 collisions / s
  - Trigger rate 100 Hz → 10 PB/y for all experiments
- E.g. ATLAS: 3.2 PB / year raw data
  - ~1 PB / year ESD + ~same from simulations
- Analysis at Tier 2 / Tier 3 centers: O(100) cores
- End-user analysis is a contineous refinement cycle

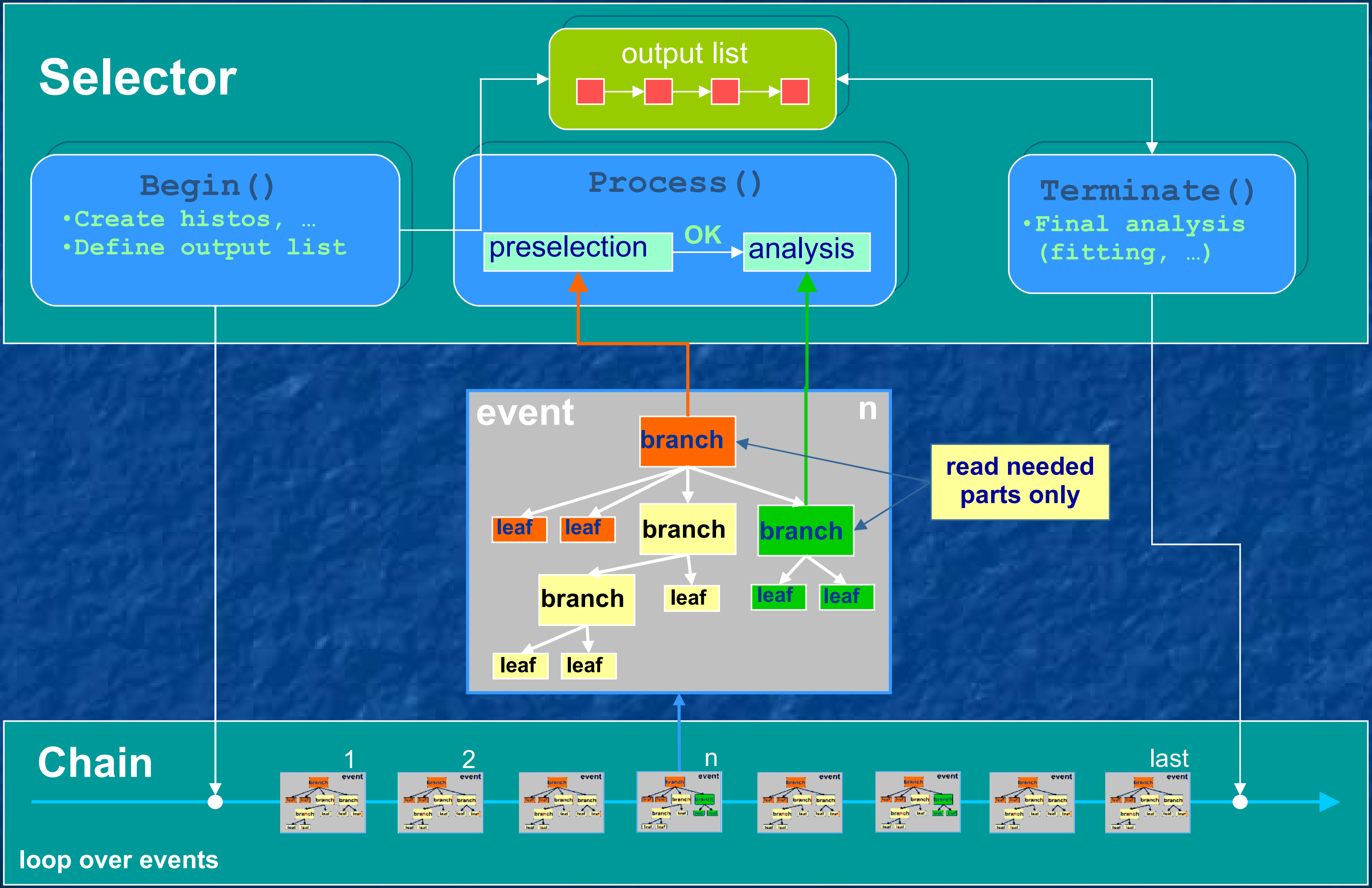


- Reading O(1) PB @ 50 MB/s takes ~230 days!
  - Using parallelism is the only way out

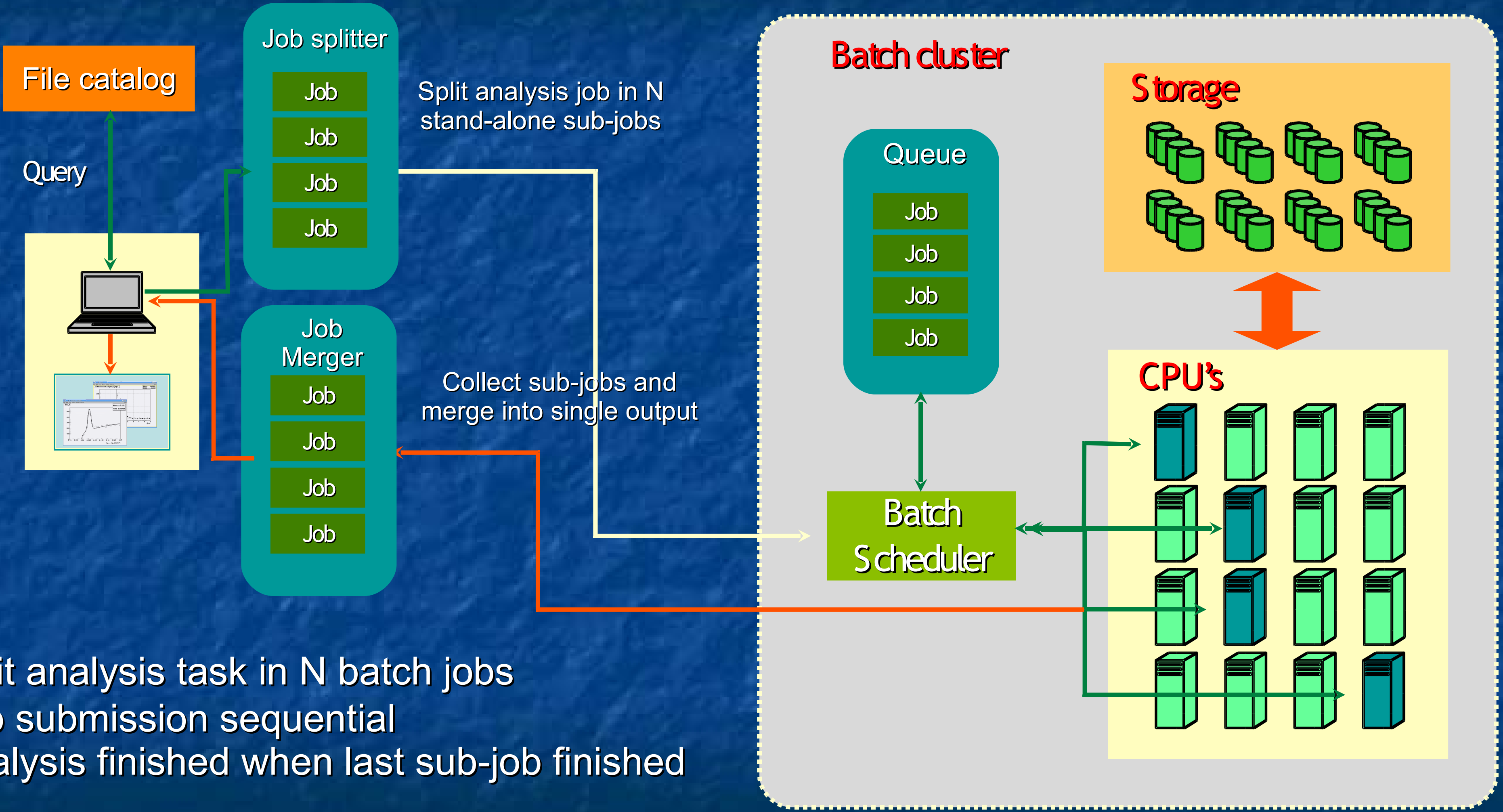
ROOT: the analysis package

- C++ framework providing tools for
  - Storage: optimized for HEP data
  - Visualization: 2D, 3D, event display, ...
  - Statistics, math functions, fitting, ...
  - Abstract interfaces: VirtualMC, ...
- Puts together what was PAW, ZEBRA, CERNLIB and more
- How does ROOT address the problem of data processing?

The ROOT data model: Trees & Selectors



Batch-oriented Approach



- Split analysis task in N batch jobs
- Job submission sequential
- Analysis finished when last sub-job finished

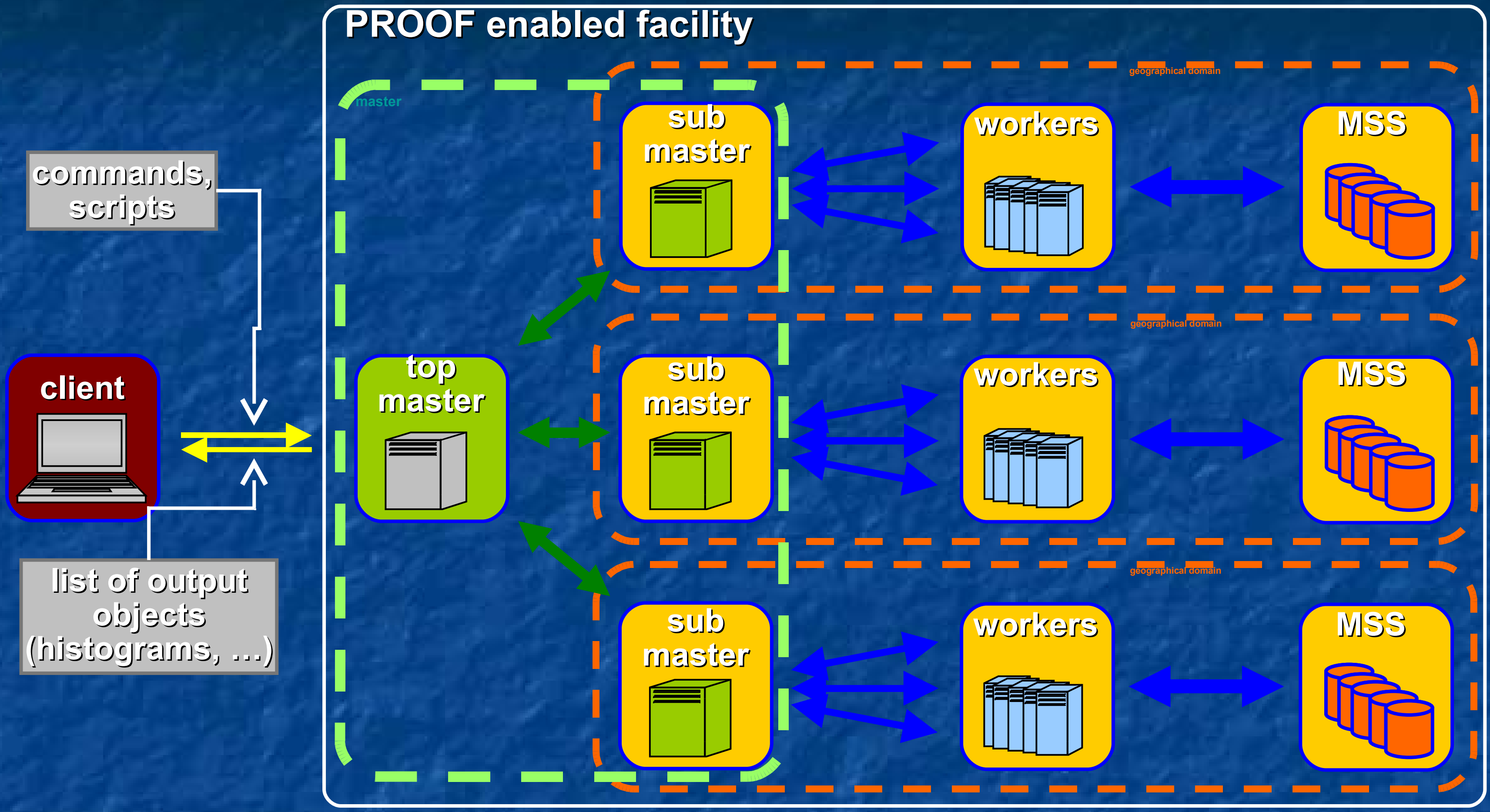
Batch-oriented approach (2)

- Works well for frozen algorithms
  - Reconstruction, creation of AOD, nano-ESD, ...
- Not very practical for algorithm refinements
  - Work-around is to reduce the data to a (temporary) compact format
  - Refine the algorithm on the compact format
  - Possibly adjust the compact format

PROOF

- Transparent, scalable extension of the ROOT shell
- 3-tier Client-Master-Workers architecture
- Flexible Master tier
  - Adapt to heterogeneous configurations
  - Dilute load of reduction (merging) phase
- PROOF daemon is a plug-in (protocol) to SCALLA (xrootd)
  - Data and PROOF access with the same daemon
  - Local storage pool
  - Coordinator functionality on the master
    - Global view of PROOF activities

PROOF architecture



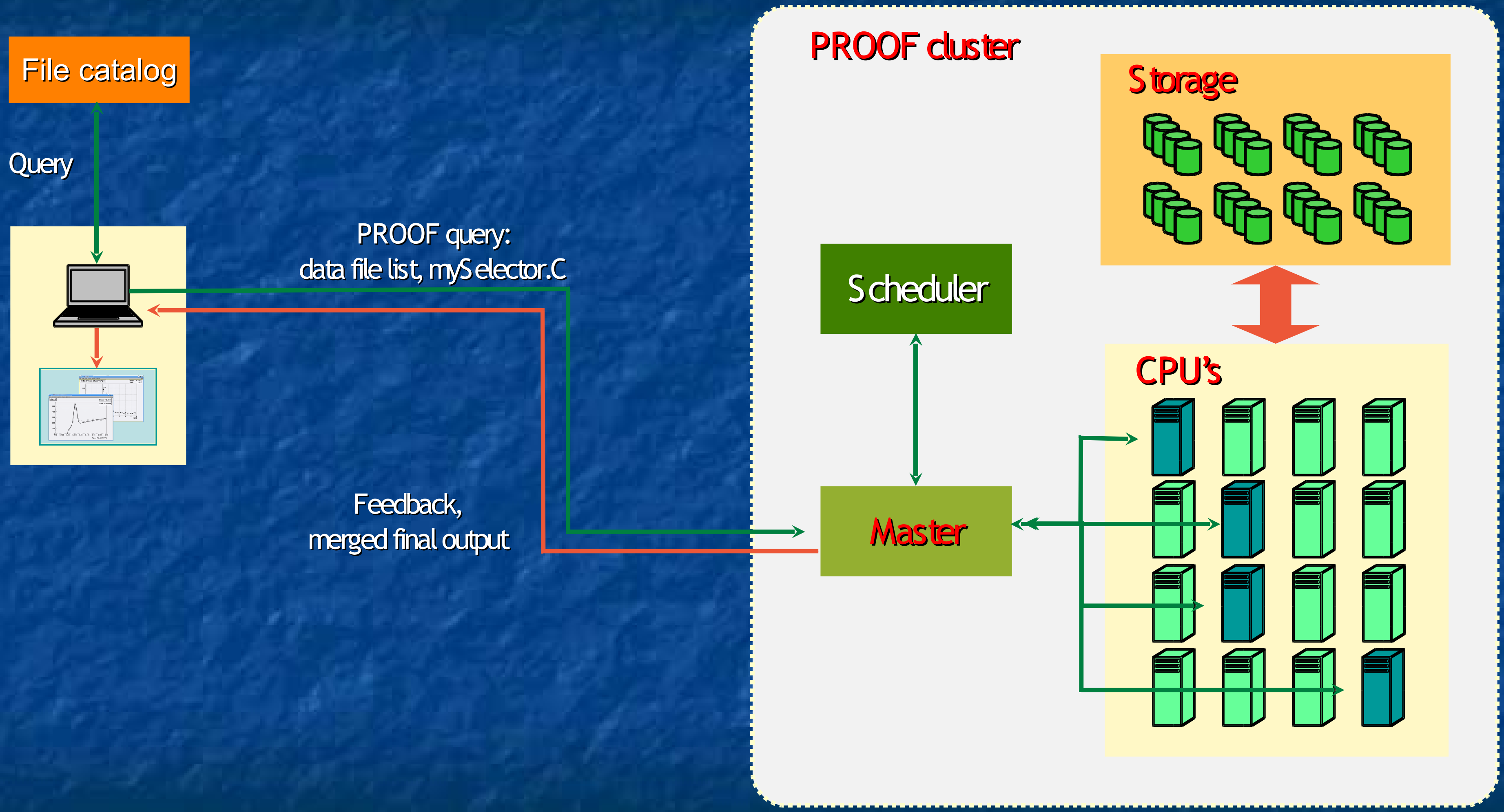
Network performance: Less important → VERY important

Optimize for data locality or high bandwidth data server access

SCALLA (xrootd)

- Scalable Cluster Architecture for Low Latency Access
- Two building blocks
  - xrootd: server for low latency high bandwidth data access
  - cmsd (olbd): server to build scalable xrootd clusters
- Multi-thread, extensible architecture
  - Top layer controls threading, memory, protocols
    - Protocol is a generic interface to services
      - Can run any number of protocols at the same time
  - Special focus on fault-tolerance
- Developed by SLAC/INFN for BaBar
- Growing interest by LHC collaborations
- http://xrootd.slac.stanford.edu

The PROOF Approach



PROOF features

- Dynamic use of resources
  - Pull architecture
    - Workers ask for work when idle
- Real-time feedback
  - Set of objects can be send to the client at a tunable frequency
- Package manager
  - Allows to upload/enable code needed by the job
- Cluster perceived as an extension of the local shell
  - Can control many clusters from the same shell
- Automatic splitting and merging

Interest in PROOF

- Started as a joint project MIT (PHOBOS) / CERN (ALICE)
  - Currently CERN (PH-SFT) + contributions from GSI Darmstadt
- Used by PHOBOS for end-user analysis since 2003
- At LHC
  - ALICE: official requirement for analysis facilities
  - ATLAS, CMS: testing analysis models based on SCALLA for data-serving and PROOF
  - LHCb: started some work to adapt pyroot (Python ROOT) to PROOF

Why PROOF and Condor ?

- End-user interactive analysis is cahotic
  - Typically intensive for limited periods of time
  - Average load on a PROOF dedicated pool may be low
- ALICE express-line at their CERN Analysis Facility aims at that
  - ~50 users for ~500 cores
  - Fast response time for prompt quality control analysis
- But in general this is not affordable
- Can we increase the average load, keeping the advantages of PROOF ?

PROOF and Condor

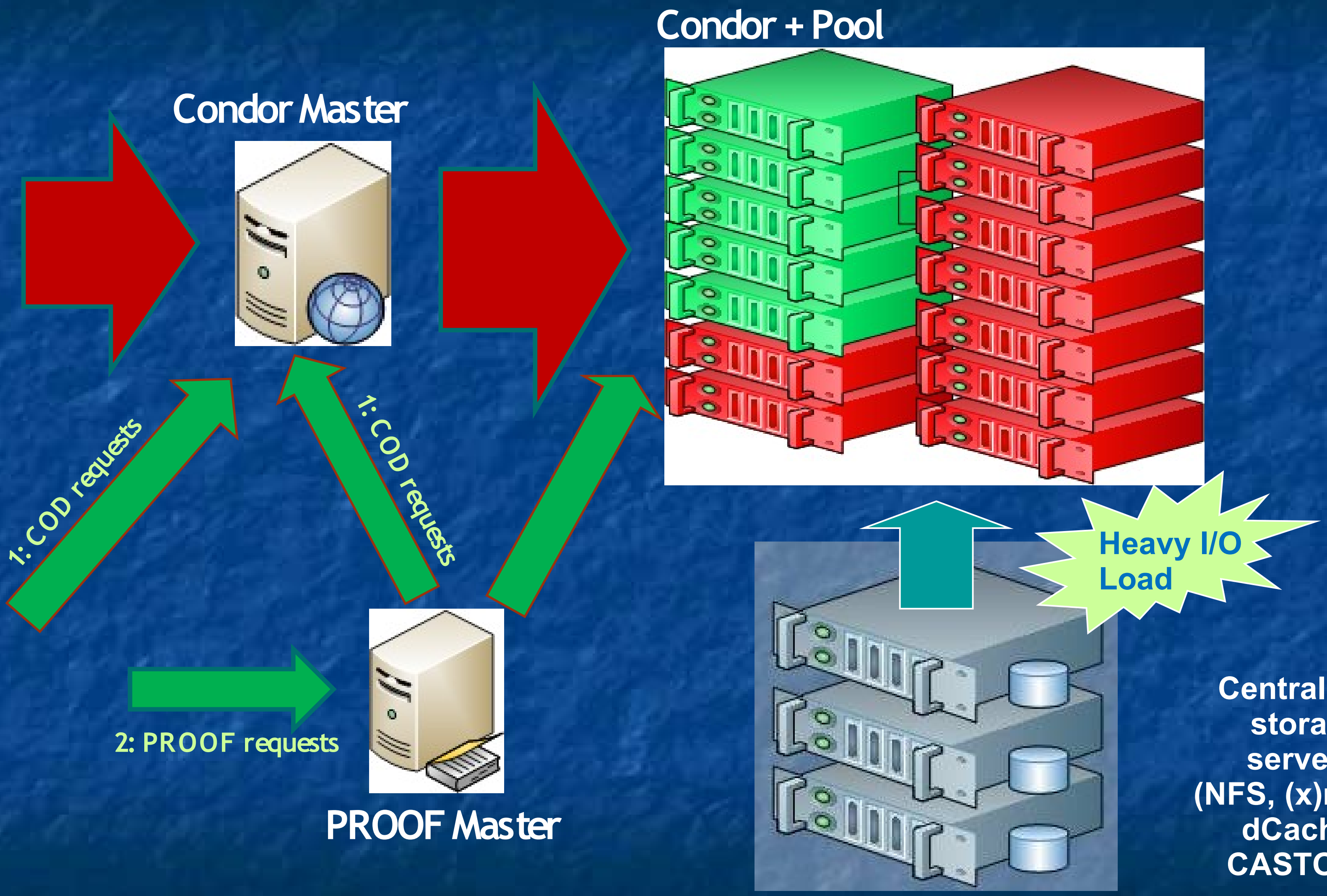
- Use Condor as a tool to share the available resources between batch-like activities and PROOF
- Get the CPUs by suspending / preempting running jobs
  - Needs to free all resources (not only CPU)
  - Simple renicing could be sufficient for CPU-intensive jobs
- Use them interactively
- Resume jobs after the session is finished

The first PROOF + COD model

- Developed for PHOBOS analysis
- Based on Computer-On-Demand (COD)
- COD requests submitted by either directly by users or by their master session
  - 'proofd' daemon started during 'activate'
- User starts a PROOF session on the allocated machines

The first PROOF + COD model (2)

Normal Production Condor jobs



Condor Master

Condor + Pool

1: COD requests

1: COD requests

2: PROOF requests

PROOF Master

Heavy I/O Load

PROOF queries

Centralized storage servers (NFS, (x)rootd, dCache, CASTOR)

The first PROOF + COD model (3)

- Pros
  - It worked: used by PHOBOS to manage their resources at RCF / BNL
  - Successful Proof-Of-Concept
- Cons
  - Release of COD claims under users responsibility
    - '-lease' not used
  - Startup scaling issues with large number of nodes
    - Needs better activate strategy
  - Potential problems with many users
    - COD does not affect Condor priority system
    - Reading data from a central storage system may cause heavy traffic and may be very inefficient

Towards an improved model

Ideas

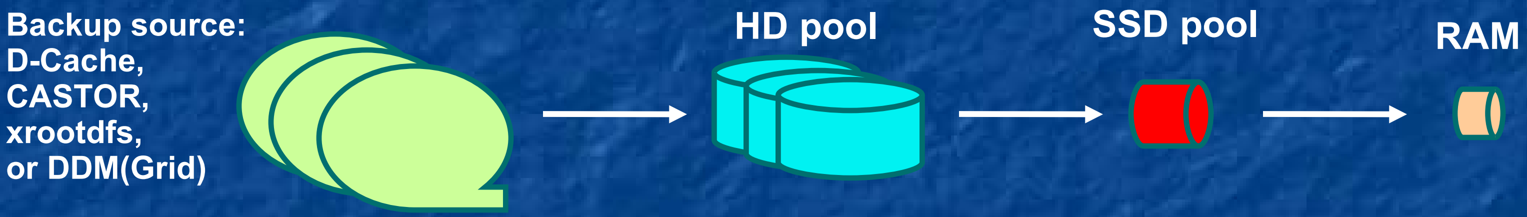
- Use standard suspension / resume instead of COD
  - Get Condor priority system in the game
- Exploit local storage to optimize data access
  - Temporarly upload data files on the pool
  - ALICE experience shows that this is more efficient if the same data are used by many groups
- Exploit global view of the system provided by the SCALLA-based PROOF connection layer
  - Control machines where to start workers taking into account exact location of data files

Some Remarks

- If pre-installation not available all what we need is ROOT either from a shared file system or from a tarball (~100 MB, from the Web or shipped)
  - Installation script available
- Local storage not available or small
  - Use remote access to files
  - Asynchronous read-ahead recently introduced in XROOTD

The ATLAS Wisconsin model

- Designed for a large scale Analysis Facility with
  - PROOF pool
  - > 100 users
  - Limited and structured storage

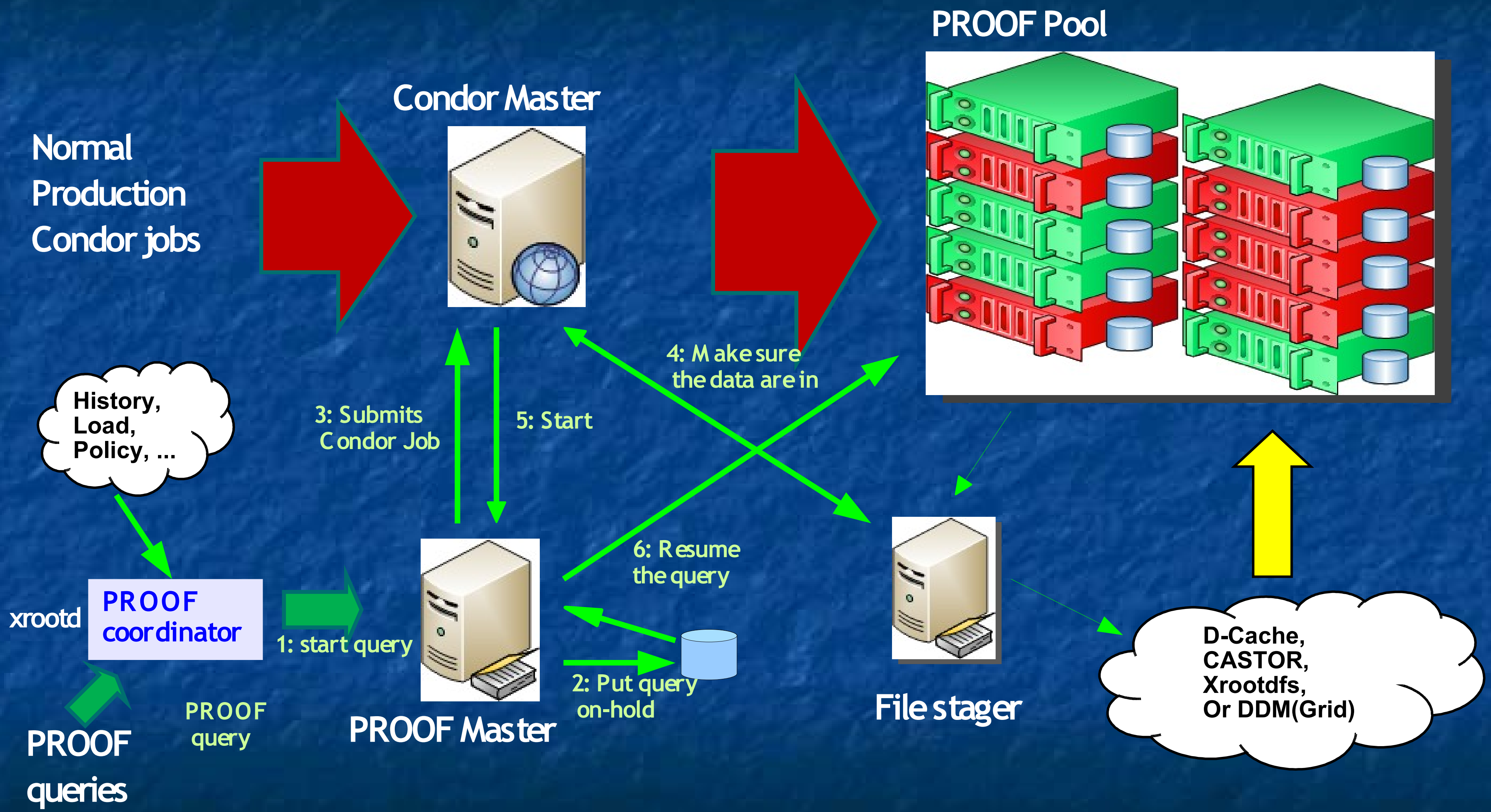


- Issues:
  - Efficient scheduling of large number of users
  - Data staging-in/-out on the pool

The ATLAS Wisconsin model (2)

- Synchronize job running to dataset availability
- Dedicated file staging daemons running under Condor
  - Database with dataset availability info
    - Datasets are named collection of files
- Control the total number of running PROOF sessions using SLOTs
  - Pool of High priority slots for PROOF
  - Pool of slots for background jobs
- PROOF scheduler can enforce external requirements, e.g. experiment specific policies

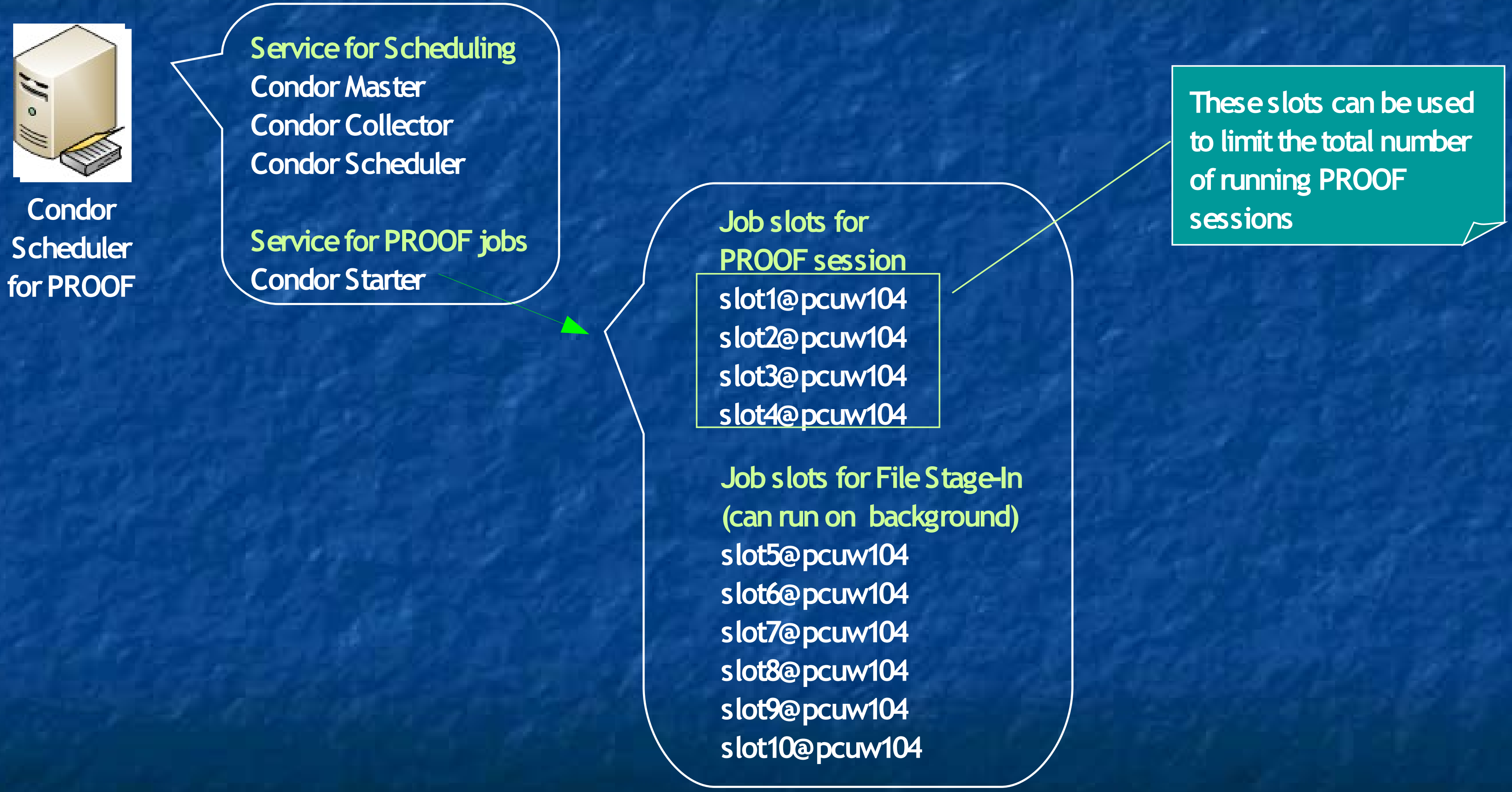
The ATLAS Wisconsin model (3)



The ATLAS Wisconsin model (4)

- Users issue PROOF processing queries in normal way
- PROOF master
  - Puts the query on-hold
  - Creates a Condor job for each query
    - Dataset readiness as requirement ClassAd
  - Submits the job to the Condor scheduler
- Condor scheduler puts the job in Held state
- File stager daemon checks regurarly for new dataset requests
  - Collects the dataset requirements and arranges the movement of files
  - Releases the processing job when their required datasets are ready
- Condor scheduler runs the job on PROOF, resuming the query put on-hold at the previous step

The ATLAS Wisconsin model (5)



Condor Scheduler for PROOF

Service for Scheduling
Condor Master
Condor Collector
Condor Scheduler

Service for PROOF jobs
Condor Starter

Job slots for PROOF session
slot1@pcuw104
slot2@pcuw104
slot3@pcuw104
slot4@pcuw104

Job slots for File Stage-In (can run on background)
slot5@pcuw104
slot6@pcuw104
slot7@pcuw104
slot8@pcuw104
slot9@pcuw104
slot10@pcuw104

These slots can be used to limit the total number of running PROOF sessions

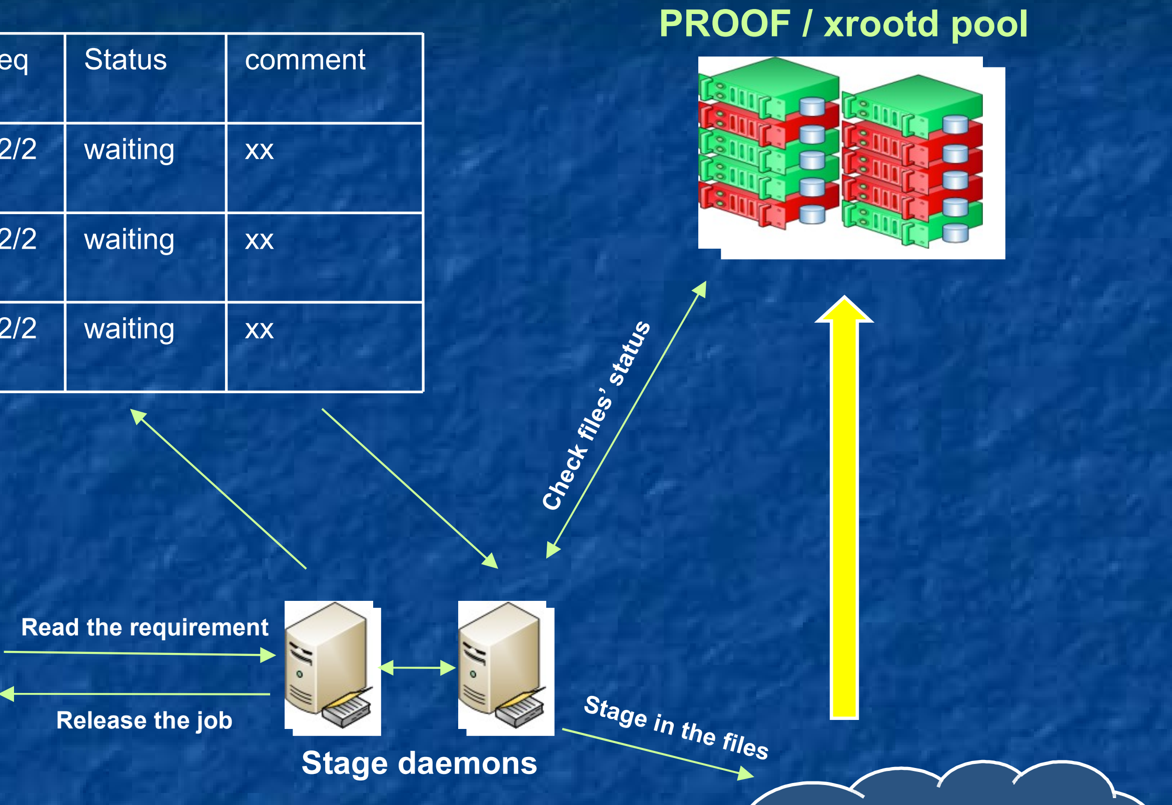
File stager

Database for Datasets

Dataset name	#of req	# of file	Last req date	Status	comment
mc08.017506.PythiaB_bbmu6mu4X.evgen.e306	2	50	2008/2/20	waiting	xx
mc08.017506.PythiaB_bbmu6mu4X.evgen.e306	1	50	2008/2/20	waiting	xx
mc08.017506.PythiaB_bbmu6mu4X.evgen.e306	1	500	2008/2/20	waiting	xx

PROOF batch jobs list

Name	input dataset
Job1	mc08.017506.PythiaB_bbmu6mu4X.evgen.e306 500
Job2	mc08.017506.PythiaB_bbmu6mu6X.evgen.e306 400
Job3	mc08.0175068.PythiaB_bbmu6mu4X.evgen.e306 30
Job4	mc08.017506.PythiaB_bbmu6mu4X.evgen 50
Job5	mc08.017888.PythiaB_bbmu6mu4X.evgen.e306 100
Job6	mc08.017506.PythiaB_bbmu6mu4X.evgen.e306 120



- Condor jobs set to “Held” by default. The Stage Server release them once the dataset is staged into the PROOF / Xrootd pool.
- Dataset stage-in also has priority which depends on the number of requests, number of files, waiting time, etc..

Open Issues

- Condor
  - Full understanding of 'slot' handling
  - Enforce dependency on dataset readiness
    - Job “held” / ”release” ?
    - Direct synchronization using a ClassAd ?
  - Startup performance issues
  - Suspension / preemption
- Dataset management
  - Error handling during file movement
  - Dataset lifetime

Open Issues (2)

- PROOF
  - Main missing ingredient was support for on-hold query submission and preempt / resume
    - Prototype on test
  - Query preemption / restart via signal
    - If Condor needs to preempt only part of the workers doing it via PROOF may allow processing to continue on the reduced set of workers
  - Start PROOF servers via Condor to fully control the session

Summary

- PROOF-Condor integration allows to optimally share a cluster between batch and interactive usage
- Basic model based on COD available since 2003
- Recent PROOF-xrootd integration allow the design of an alternative model
  - Addresses the case of concurrent multi-user analysis of large amounts of HEP data
- Working prototype under development